Design of a Satellite Tracking Station for Remote Operation and Multi-User Observation

Mark Wilkinson
Graduate Student
Dept. of Electrical and Computer Engr.
Utah State University
Logan, Utah 84322
mark@tower-ravens.org

Dr. Charles Swenson
Assistant Professor
Dept. of Electrical and Computer Engr.
Utah State University
Logan, Utah 84322
Charles.Swenson@ece.usu.edu

Abstract . With the advent of LEO constellations, the operation of multiple ground stations becomes an important factor in maintaining several satellites. With a single ground station, the need to perform anomaly resolution on a low-orbiting satellite becomes difficult with short view times and long durations between revisit. The staffing of several ground stations or specialized remote-operation equipment can be cost prohibitive.

This paper discusses the design of a satellite ground station at Utah State University (USU) that can be remotely operated over the internet. The design of this ground station outlines software drivers for controlling hardware through a computer interface. A server daemon design is given that provides connections for multiple users. This server allows an internet connection to the hardware drivers. The design presented illustrates a daemon process running on the Sun Solaris 2 operating system. A graphical user interface (GUI) design was done in the Java programming language. With an embedded TCP/IP client, compatibility is maintained between the C server and this Java client. The end-result is a ground station that can be operated via telepresence through a Java applet.

Such a design will allow the operation of multiple low-orbiting satellites, the utilization of several uplink points, all from any internet node. Without a requirement for special equipment, the operation of an experimental satellite or an entire satellite constellation becomes more cost effective.

Introduction

This paper is a summary of work performed for the completion of a Masters of Science Degree in Electrical Engineering. The author of this paper completed this work in absentia while working professionally as an engineer.

The Air Force Office of Scientific Research (AFOSR), the Defense Advance Research Projects Agency, the National Aeronautics and Space Administration (NASA), and industry partners are sponsoring the launch of 10 university nanosatellites in late 2001.[7] Three of the universities participating in this program, Utah State University, Virginia Polytechnic Institute and State University (Virginia Tech), and University of Washington, have formed a team to fly their satellites as a small constellation. The three satellites are to fly in a leader follower (a.k.a. string-of-pearls) configuration and make measurements of the ionosphere (Figure 1). The project contains barely enough funding for spacecraft hardware. Most labor on all three satellites is being donated by professionals, faculty, and students.

This constellation has been given the mission name ION-F for Ionospheric Observation Nanosatellite Formation. There are four objectives of the mission. First, demonstrate the control of a nanosatellite formation using passive and active means. Second, make the first multi-satellite measurements of the ionosphere. Third, demonstrate new technologies for satellite intercommunication and relative position determination. Fourth, provide hands-on experience for graduate and undergraduate students in space technology.

The preliminary design includes a combination GPS receiver and intersatellite crosslink being developed by the Applied Physics Lab of Johns Hopkins University for NASA. This unit provides absolute and relative location of the satellites in the constellation for navigation though the global positioning system (GPS). Data can be cross-linked through the system allowing one spacecraft in the constellation to act as master to control the constellation dynamics. Alternatively, data can be exchanged such that the satellites can act more independently and use distributed control to maintain the constellation.

All three of the satellites in ION-F will be controlled by two ground stations located at Utah State University and Virginia Tech. These two ground stations have wide geographic separation and provide sufficient access time for the mission. It was desirable to

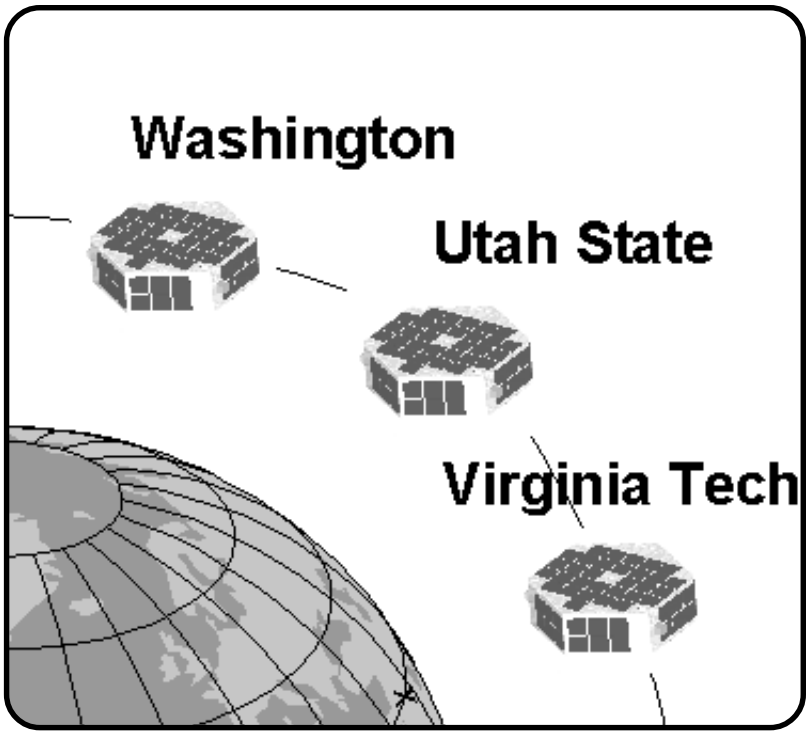

Figure 1. ION-F Constellation

develop technology that would enable Washington University to remotely use Utah State University's ground station. Additionally, the USU ground station is dedicated solely to the ION-F program. Because the Earth-space link may require a unique design that Virginia Tech may not be able to use, the advantages of having a single common ground station became somewhat apparent. Of course, a single ground station would only be feasible if all three universities could remotely access and simultaneously use it.

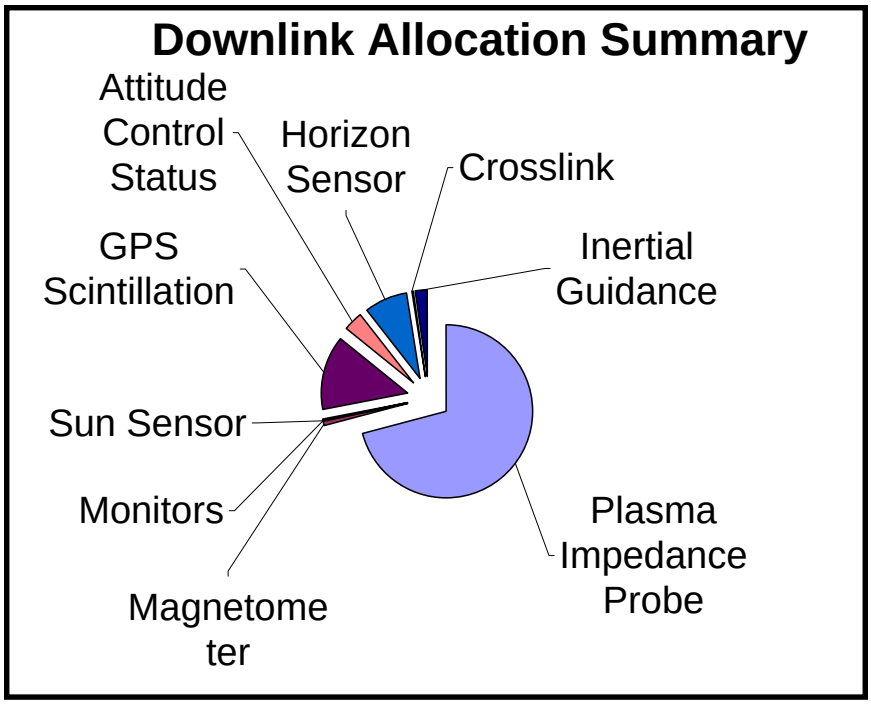

Figure 2. Composition of satellite telemetry and data.

The system developed and prototyped to satisfy these desires is applicable to many challenges of constellation control.

The system, as it stands now, only gives the user the ability to control the dish track. The telemetry has been designed, and will be developed as hardware is selected. The Ion-F Satellites will have full uplink and downlink capability to the ground stations at Utah State University and Virginia Tech. The system will be built around commercial aerospace parts and will leverage off of the Utah State University Space Dynamics Lab's experience in telemetry systems. The system will be all digital using simple frequency shift keying for modulation. A 10 kbit uplink is proposed at around 450 MHz and a 100 kbit downlink is proposed within S-band. Each of the three satellites will use the same telemetry formats and share the same frequency allocations.

Passes over the two ground stations will be divided into turns between the three satellites. For the first few months of the mission all three satellites, from the perspective of the ground station, will be in the sky at the same time. A single dish can only track one of the satellites. In addition, only one satellite will be allowed to use the downlink frequency at a time to prevent interference. Thus an entire overpass will be assigned to one of the three satellites.

Simulations for computing the times and duration of overpasses show that an average collection data rate of about 800 bits/s can be supported on each of the ION-F satellites. Given the two ground stations and a 100 kbit downlink, the on board storage requirements of the satellites under this scenario are about 20 Mbytes. The breakdown of telemetry and data is shown in Figure 2.

Despite the fact that the telemetry link is not yet operational, the potential value of the ground station design has been demonstrated. The system was designed to permit the support of a short satellite pass where predetermined commands are uploaded, and telemetry and data are downloaded to a repository for later collection and analysis. With this methodology, the bandwidth requirements between the remote operator location and the ground station position are very low. Specifically, the bandwidth demands for the station telemetry can be met with a data transfer rate of 1000 bits per second. To put this in perspective, the data communications requirement between the operator and the station can be met with a 1200 baud modem. Real-time reporting of satellite telemetry will increase the bandwidth demand, but it will not be significant for a small satellite.

Additionally, because the requirements of this particular system are very relaxed, and because the system is automated to a certain degree, short lags inherent in using the internet are tolerable. This system, in Utah State University's application, will be fully operational using a standard low-bandwidth internet connection.

There are several applications of this system to industry, and many of the shortcomings of this prototype system can be easily overcome. Because the complexity of this system exists primarily in software, it is very inexpensive to obtain. The ability to operate this system over the public internet reduces its cost to a mere pittance. Of course, using the public internet to operate this system puts several segments of the link out of the control of the satellite operators. In a more critical satellite system, this can be intolerable.

Private dedicated communications links can be used instead of the internet. While this will certainly increase monthly operations costs, the cost will be offset by the reduction in manpower.

In even this prototype design, this system has the capability to support a significant constellation. The communications links for several satellites from many ground stations can be established from a single operator position or multiple operator positions. From widely dispersed geographical locations, one operator can control and several observers can monitor multiple satellite passes. Significant additions would need to be made, however, to provide continuous real-time monitoring. Such a system would also require a much broader bandwidth.

Figure 3. Utah State University Ground Station Pedestal and Dish

The Design

The satellite ground system consists of a Parallax 12' parabolic reflector mounted to a Bendix GMD-1 pedestal. The dish and pedestal are shown in Figure 3. The pedestal contains the control electronics. Aside from these control electronics, there is a UNIX workstation that communicates with the tracking station users over an internet connection.

This system operates by utilizing several different computer programs operating in harmony over a network system. The most expensive component of the design is, of course, the reflector and pointing control of the ground station. The other major component of the system is a UNIX workstation that manages communications between the ground station and the multiple users. While it is conceivable that a personal class computer could possibly handle this task, the reliability and multi-user restrictions make it undesirable.

Using a UNIX workstation, particularly a high class workstation such as a Sun, Hewlett-

Packard, or Silicon Graphics machine, may create some cost concerns. These concerns are unwarranted. The system used to prototype the Utah State University ground station was a low-end Sun SPARC workstation that was purchased and refurbished for under $1000.00. Additionally, unlike heritage systems, this design requires only a single workstation. It is used as the intermediary between the ground station and the operators. The computers used by operators can be simple personal class computers, including Macintosh or Wintel systems as well as UNIX workstations. Operators can even work on laptop computers through a modem connection.

There are three major computing components to this system. First, there is the microcontroller embedded in the station pedestal. Second, there is a workstation between the network and the embedded microcontroller. Third, there is the operator's computer, of which there may be several. In the completed system, there will also be a radio transceiver connected to the UNIX workstation. Figure 4 illustrates these components.

In our prototype design the first component, the embedded controller, is a Tattletale 8 microcontroller. It is based on the Motorolla 68332 microcontroller. This embedded controller receives position information about the satellite dish through two quadrature, relative position, optical encoders. One reads azimuth; the other reads elevation. Position control of the dish is accomplished through pulse-width modulation of two DC motors; again, one for azimuth and the other for elevation. An ephemeris table, which includes time, desired azimuth, and desired elevation,

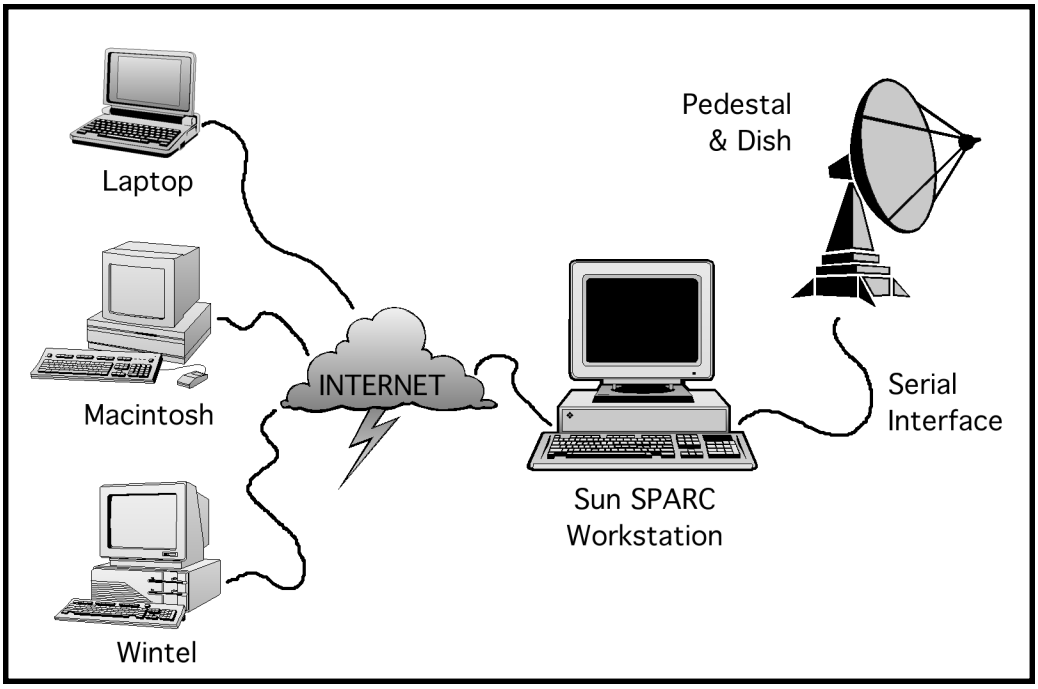

Figure 4. Operators, a workstation, and the pedestal are the three primary components of the ground station.

is received from the workstation through full-duplex, asynchronous, serial communications using the RS-232 standard. The microcontroller also transmits position and status information to the workstation through this same link.

The second component of our prototype design uses a low-end Sun SPARC workstation. While the workstation used in this design operates on the Solaris 2 system, this code can be ported to many UNIX platforms. There are two segments of the workstation code. Segment A is a driver daemon; segment B is a server daemon.

Daemon processes are discussed in more detail later in this paper. Suffice it to say both of these processes operate as part of the UNIX operating system. User interaction is not necessary to start nor maintain them.

The driver daemon is the communications point between the dish-embedded microcontroller and the server daemon. The driver daemon passes commands and ephemeris to the microcontroller, it collects telemetry from the microcontroller, and it determines user privileges for each operator. Communication to the microcontroller is through an RS-232 serial interface using a custom protocol. Communication to the server daemon is through UNIX SVR4 IPC (System V, Release 4, Interprocess Communications) using a custom protocol.

The server daemon is the gateway for operators desiring connection with the ground station. Every time a user requests connection with the system, the server daemon spawns a duplicate process to handle the connection request. It then remains idle, waiting for additional connection requests. The spawned process handles the user's connection. There is one daemon process always running, and there are as many spawned processes as there are users, which self-terminate when the user disconnects. The spawned process handles login requests to the ground station tracking system, and it passes command and telemetry between the driver daemon and the operator's display.

The third system component is the operator's computer. As mentioned, this can be almost any type of platform, including UNIX, Macintosh, and Wintel. The operator must be able to operate under the Java environment and must download the ground station user display from the ground station computer. While the other programs in this system were written in combinations of assembly and C, the user display was written in the Java programming language. Not only does this give the flexibility to run on multiple platforms, but it removed the computationally intensive task of maintaining the user display from the UNIX workstation. Instead, the user display resides on the operator's computer. Because the display provides only a view of the information arriving from the workstation, as well as buttons for commanding the workstation, there is no concern about the performance limitations of Java. The user display communicates with the workstation server daemon through the abstract socket interface over the internet. It uses a custom protocol built on top of the TCP/IP stack.

Microcontroller Design

During the development of this design, the conclusion was reached that the control law should be implemented on a microcontroller embedded in the dish pedestal. The Tattletale Model 8 microcontroller made by Onset Corporation was selected to provide this control. The Tattletale 8 is based on the Motorolla 68332 microcontroller with additional peripheral electronics. Motor drivers

and power isolation circuits were added to the system which was installed in the pedestal.

The microcontroller provides only three functions: First, it provides communications to the workstation. Second, it determines the azimuth and elevation of the satellite dish from two quadrature, relative, optical encoders. Third, it determines the desired position of the dish and controls two DC motors with pulse-width modulation to establish this position.

Communications between the microcontroller and the workstation occur over an RS-232 connection.[4] This provides asynchronous, full-duplex communications at 9600 bps. The microcontroller transmits status and position to the workstation; the workstation transmits commands and ephemeris tables to the microcontroller.

Ephemeris received from the workstation includes system time, azimuth position, and elevation position. Data points are generally separated by one second, but this interval can vary. Using three ephemeris points, the microcontroller determines intermediate points using a cubic spline.

Accurate time is obtained by the microcontroller from the workstation. Currently the workstation receives its time via the NIST (National Institute of Standards–Time) service. This service periodically updates the time on the workstation from a national standard. The closest location of this standard to the USU ground station in located in Denver, Colorado. While network latencies between Denver, Colorado and Logan, Utah are considered when updating the time, accuracy may be improved by adding a global position system (GPS) receiver to the workstation. Accurate time would then be obtained from the GPS constellation.

Workstation Daemons

The term daemon is applied to any UNIX process that runs in the background. As opposed to a program that runs in the foreground, a daemon process has no user interaction through standard channels. A daemon process is invisible to the user. Typically, these processes are always running, and wait for some event to occur that they service.

In the case of the USU Satellite Ground Station, both daemon processes are started with the operating system when the computer is powered up. When started, each daemon process immediately duplicates itself in memory with the fork function. The original parent process creates a child process. The parent process then terminates itself. The operating system will detect that the child process has been orphaned. In the server daemon, this child becomes the parent to each user connection.

When a child process is orphaned, the operating system adopts the child process with its initial system process (init). This has the effect of making the daemon process a part of the operating system. The child process also performs several other functions that protect it from outside computer resources; it is buried even deeper into the operating system.[1]

This child process, the daemon, is completely invisible to the user. For a system administrator, it is very difficult to identify and maintain a daemon process; it is almost impossible to diagnose a malfunctioning daemon. To circumvent this problem, the daemon process creates a self-identifying lock file in a known directory. Using the syslog service in the UNIX operating system, messages are created by the daemon process and stored in the same directory. Both notices and error messages are recorded in this file. These system message

logs will enable the system administrator to diagnose erroneous or fatal behavior of the daemon.

A good example of a UNIX daemon is a printer driver. The daemon does nothing until a print request is made. When a request is made, it services the request by gathering the data from the user's application program, processing the data, and then passing the data to the printer in a new format. The user does not need to execute additional commands between issuing the print command and collecting the paper from the printer, but between these user actions, the printer daemon executes a large number of very complicated commands.

There are two daemons in the ground station system. Figure 5 illustrates the communication paths between the two daemons, the dish tracking electronics, and the users.

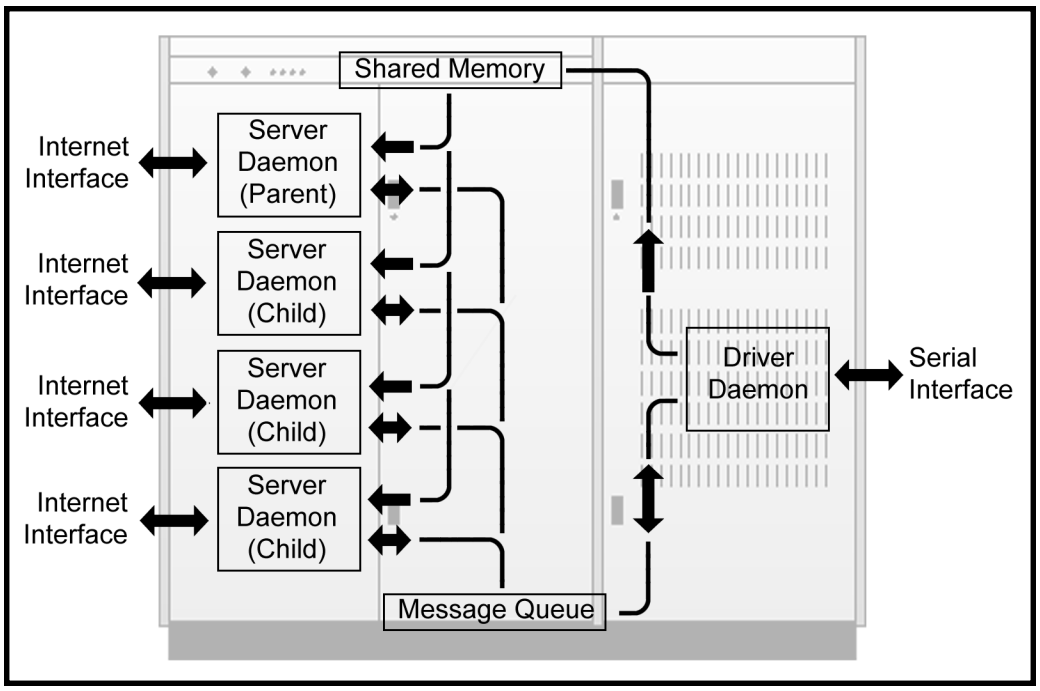

Figure 5. The server daemon with three users and the driver daemon with their communication paths—TCP/IP sockets, IPC shared memory, IPC message queue, and serial connections.

Driver Daemon Details

The driver daemon was created to service information between the ground station hardware and the server daemon processes created when users log in. While the user interface restricts commands, the driver validates the permission level of each command. Unauthorized commands will be rejected if the user does not have the proper access level.

When a user connects to the ground station system, they are immediately given permission to observe the system. They are unable to issue commands. *Observer* is the lowest permission level. *Operator* is the next level of access. An operator is given the ability to add missions to the ground station. A mission is a command to track a satellite at a given date and time. The operator provides the ephemeris through their user interface.

The final permission level is *technician.* In addition to having all operator permissions, the technician has the ability to shutdown the system at any time, move the dish to its lock-down position, calibrate the dish pointing angle, change the station's geospatial coordinates, and manually command the slewing of the dish.

On a side note, any person has the ability, through a restricted shell login, to prevent or allow movement commands to the dish. This is needed when the dish position is locked down because of wind conditions or disuse. The restricted shell login will only be accepted from the main terminal located in the locked ground station lab adjacent to the dish.

Efforts to force commands to the ground station system through hacking the ground station user interface will be detected by the driver. As mentioned, the user interface restricts commands, but permissions are validated by the driver. Hacking the user interface cannot circumvent security.

The driver reports a constant set of information to all users. It also accepts individual messages from each user and transmits specific messages to individual users. This capability is accomplished through the UNIX Interprocess Communications (IPC) service.[2]

For the constant set of information that is sent to all users, the driver uses a shared memory location. Several times every second, this information is updated. It includes the azimuth and elevation of the dish as well as system time. Because the shared memory is handled by the operating system, each of the server processes servicing users is able to obtain this information without querying the driver. This helps protect the driver from being overloaded by user requests.

The second service is called a message queue. It is a location in system memory where messages are transmitted by one process and retrieved by another. Message identifiers prevent the messages from being inadvertently retrieved by the wrong process. Server processes receive this identifier when the user attaches to the system. Unlike the shared memory, the driver must service every message that is sent. The speed of the computer hosting the driver will restrict the maximum number of users. The maximum number of users allowed is set by the system administrator when the driver is installed into the system.

Server Daemon Details

While the server daemon has all the functions necessary to service a user connection, it does not do this. Instead, the server daemon simply waits for connections. When a connection request is made, the server spawns out a child. The server then goes back to simply waiting for connections. This new child process, which is almost identical to the original parent, then services the user. When the user completes their session, the child process then terminates itself.

The process of servicing internet connection requests can be accomplished by the UNIX Internet Supervisor daemon (inetd). With inetd, a daemon process is much simpler to develop. Unfortunately, there is a performance loss.[9] Because the UNIX computer used in the ground station is dedicated solely to one purpose, the server daemon was developed to run independent of the UNIX Internet Supervisor. This gives it better performance.

The server daemon can spawn as many children as it needs to service an equal number of users. Only the driver daemon has the ability

to limit the number of users by commanding the user's display program to close itself.

As discussed in the previous section, the server uses the UNIX Interprocess Communications (IPC) library for communicating with the driver. Common information such as azimuth, elevation, and time is placed in a system shared memory location by the driver. The server retrieves this information several times a second and transmits it along to the user display.

When the user display needs to transmit information to the driver daemon, the server passes information through a message queue, another IPC service. A login command or an ephemeris table are examples of messages that would be sent from the user display to the driver daemon. The driver sends information back to the user display through the message queue also. Such a message can cause the user to automatically log out, or it can activate controls on the display that were previously restricted. Once again, the server is the intermediary. The server polls the message queue and transmits any messages along to the user display.

Communication between the server daemon and the user display is quite different. It relies on a standard internet connection. The ground station uses a custom protocol built on the Transmission Control Protocol, Internet Protocol (TCP/IP) stack that is registered in the UNIX machine's services database.[8] This protocol is used to send information between the server daemon and the user interface. This is the main reason the server daemon spawns a separate process for each user: The communication interface between the server daemon and user display is connection-oriented. This is in contrast to User Datagram Protocol, Internet Protocol (UDP/IP) which is connectionless.[4]

In providing the interface between the driver daemon and the user's display, the server provides another very important function. Because the core system—meaning the driver daemon and the server daemon—consists of C programs residing on a UNIX machine, and because the user display is a Java program residing on another machine, there needs to be a set of functions that convert Java standardized datatypes to nonstandard UNIX datatypes.[5,10] The server provides these functions.

The UNIX datatypes used in this application follow the standard established under Sun Solaris 2. Single byte UNIX character types and string elements are converted to Java standard two-byte unicode characters. Java booleans and byte integers are converted to UNIX signed short integers, Java short integers are converted to UNIX C integers, and Java integers are converted to UNIX long integers. Fortunately, Java float and double types follow the same format as UNIX float and double types. Unfortunately there are some incongruities. Java does not have unsigned integers types, and UNIX C does not have a standard equivalent of the Java long integer. Special handling was required for these complexities.

User Display Details

The user display was written to conform to Java 1.1. Because of its relative young age and limited support by applications and operating systems, Java 1.2 was not used. The Abstract Windowing Toolkit (AWT) was used to construct the display, not the native looking Swing toolkit.[3] The display is shown in Figure 6.

The user display contains several regions. There is a mission time frame, a mission management frame, a station position frame, and a station pointing frame. A fifth frame has been reserved on the display to support the addition of a satellite status and transceiver interface frame.

This mission time frame gives the year, day of year, hour, minute, and seconds in military format Greenwich Mean Time (GMT), the standard used in aeronautics and astronautics. When the user display is started, an offset is determined to obtain the difference between the ground station system time and the user's

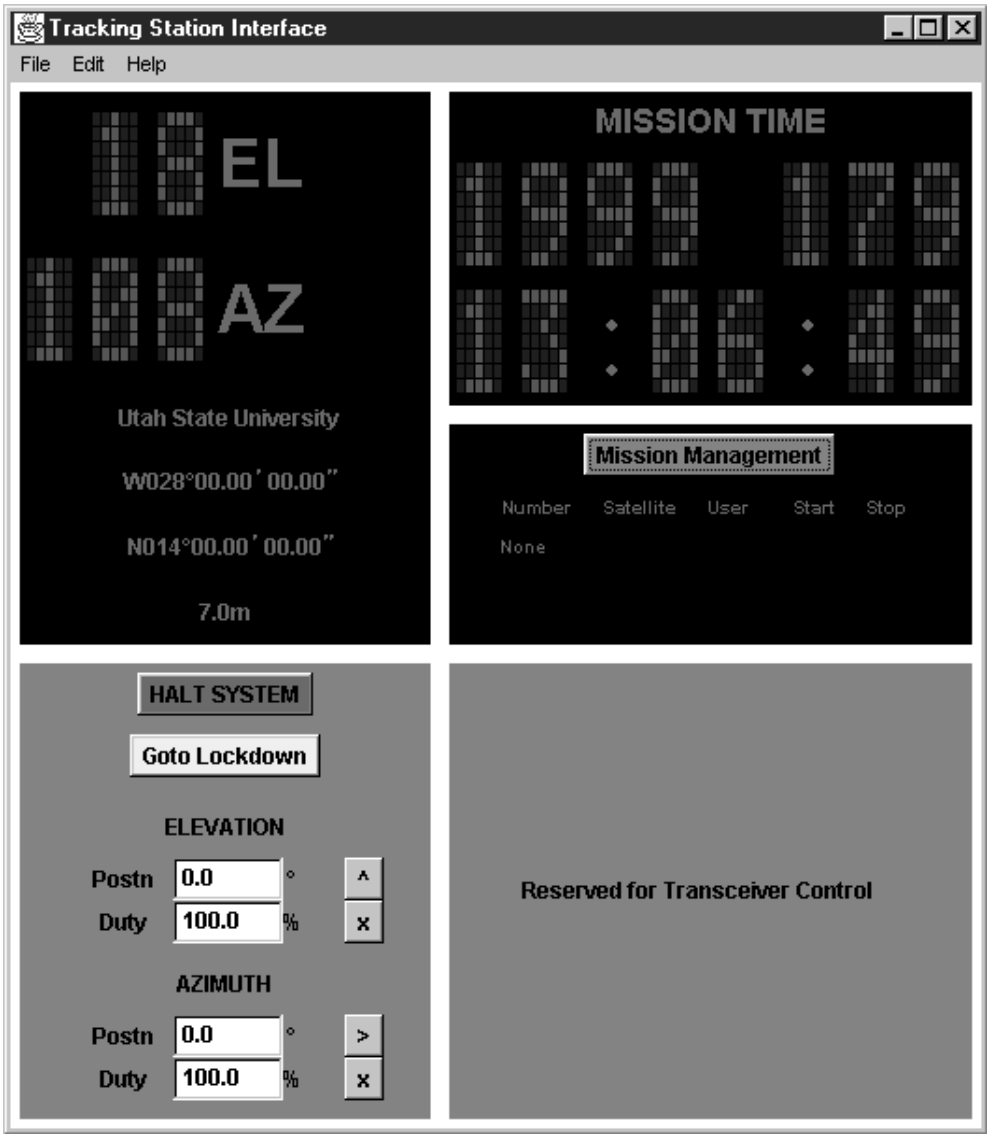

Figure 6. The User Display.

system time. The mission clock then uses the time service on the local machine rather than taxing the resources of the ground station computer.

The mission management frame serves two purposes. First, it tabulates the missions currently stored. This information is publicly available. Second, it provides an interface for adding or deleting missions. This action requires the user to have at least operator permissions. Currently missions can only be added by downloading an ASCII format ephemeris table. The capability exists, however, to interface the user display directly to Satellite Tool Kit (STK).

STK is a commercial off-the-shelf (COTS) product. The core product is an excellent tool for generating satellite ephemeris. Additionally, is available for free download and use from Analytical Graphics, Inc. Additional modules can be purchased that expand the capability of STK. One of these modules, STK-Connect, enables another program to communicate with STK through a TCP/IP interface. The user's display could be further developed to include this capability.

The position of the dish is reported in the station position frame. This includes not only the azimuth and elevation pointing of the dish, but the geospatial coordinates of the dish-latitude, longitude, and altitude. This information, including the geospatial coordinates, is provided to the user display from the driver daemon. Therefore, a user display can be adjusted to connect to any number of ground stations. The display will properly report the position of the ground station to which it is connected. For those with proper privileges, manual commanding of the dish position can be performed. The desired position and a maximum motor duty cycle are specified.

As mentioned in the previous section, the server daemon provides the service of unit translation between the Solaris implementation of UNIX datatypes and the Java standard for datatypes. Nevertheless, special handling is required to get data from the network connection to the user display. Data arrives to the user display in a stream of data. This stream of data is reliably delivered to the user by the TCP/IP service. It is, however, just a stream of bytes. The user display identifies the message stream from its first few bytes. Based on this identification, the stream is then assembled into meaningful datatypes—integers, floats, strings, etc.

The converse is true when sending information through the network connection. The information to be sent is dissolved into individual bytes and reassembled as a stream of bytes. It is given an identifier and then transmitted over the network. Provisions have been added in the protocol to implement error-checking, ensuring data was not corrupted during transmission. This has not been necessary because of the reliability of TCP/IP and has not been implemented.

Future Developments

Currently, there is only one ground station that uses this system. It is the station at Utah State University. If other locations were to adopt this software into their ground stations, the user display would be able to select the ground station, from a list of stations, to which it would connect. This is a very powerful feature that can be easily added. For a more complex satellite system where more ground stations are needed, all ground stations can be controlled from a single terminal. One user window can swap between multiple ground stations or multiple user windows can be opened to monitor and control each of the ground stations.

After the satellite design has been finalized and a transceiver for the satellite has been selected, capability to communicate with the ground station transceiver will be added to the user interface; space has been set-aside in the current interface for this addition. This will compete the design objective of being able to operate a satellite from a remote location.

In its present implementation, ephemeris can only be provided to the system in the form of an ASCII table. As mentioned earlier, it is anticipated that an interface to Satellite Tool Kit will be created using the STK-Connect module and a TCP/IP interface. This would greatly simplify ground station operation and would reduce operating costs, as operators would not require as much training.

Because of recent construction, the physical location of the satellite dish will need to be moved. Currently there is a restriction that the workstation must be located within 50 feet of the dish to accommodate the RS-232 communications between the workstation and the embedded microcontroller. This requirement can be circumvented by adding a modular RS-232 to ethernet converter. This may need to be done to accommodate a new location for the dish. While the workstation and microcontroller will need to be collocated on the same subnet, this will allow a much greater physical separation.

There is some desire to demonstrate a degree of pointing accuracy that is not necessary for this project. This may be pursued in the future.

The current time base for the system, which is acquired using the NIST service over the internet, may not be sufficient. If it is necessary to improve the accuracy of the time base, a GPS receiver may be added. Time would then be obtained with higher accuracy from the GPS satellite constellation.

Source Code Licensing

The source code for this system is available from its author, Mark Wilkinson. The code is not in the public domain; it must be licensed. Licenses are free but require a license agreement with the author. The license agreement basically states that any improvements to the system must be given to the author so that they may, at the author's discretion, be incorporated into the baseline for further distribution. The license agreement also places some provisions on its use in commercial systems, and it prohibits redistribution of the code. Contact the author for a copy of the source code and the license agreement.

The author can most reliably be contacted through his email address at "mark@tower-ravens.org". It is anticipated this email address will be valid throughout his lifetime.

Summary

This ground station will be used to operate the ION-F satellite constellation being developed by Utah State University, Virginia Tech, and the University of Washington. These satellites will, among other objectives, demonstrate formation flying. The ground station will demonstrate the capability to monitor and control several satellites with one station and several users.

The station design incorporates a core UNIX program that operates as a system daemon process. It provides communication between multiple users and a the control electronics in the pedestal of the satellite tracking system. The user display was programed in Java with an internet socket; it can connect from any internet node, using many types of computers, to the ground station.

While this design utilizes the public internet, a dedicated connection can be established for more demanding applications.

The ground station architecture provides several possible advantages. Several unmanned ground stations can be operated by a one or more users in a remote location. A ground station can even be operated over a modem connection from a laptop. With this capability, the cost of operating a large, low Earth orbit constellation can be greatly reduced. In addition to reducing manpower requirements at remote stations, the hardware costs are much less, and bandwidth costs can be very economical by using the internet.

A free license for this software can be obtained from the author. The operation of ION-F in late 2001 will demonstrate the capabilities of this ground station. Until this time, further developments will be made, and the transceiver interface will be integrated.

References

1. Comer, Douglas E. and David L. Stevens. Internetworking with TCP/IP, Volume III: Client-Server Programming and Applications. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

2. Curry, David A. UNIX Systems Programming for SVR4. O'Reilly & Associates, Inc., Sebastopol, California,1996.

3. Flanagan, David. Java in a Nutshell, 2nd ed. O'Reilly & Associates, Inc., Sebastopol, California,1997.

4. Halsall, Fred. Data Communications, Computer Networks, and Open Systems, 3rd ed. Addison-Wesley Publishing Company, Workingham, England,1992.

5. Horstmann, Cay S. and Gary Cornell. Core Java 1.1, Volume l: Fundamentals. Sun Microsystems Press, A Prentice Hall Title, Mountain View, California,1997.

6. Liu, Cricket, Jerry Peek, Russ Jones, Bryan Buus, and Adrian Nye. Managing INTERNET Information Services. O'Reilly & Associates, Inc., Sebastopol, California,1994.

7. Martin, Maurice, Howard Schlossberg, Joe Mitola, et. al. "University Nanosatellite Program." IAF Symposium, Redondo Beach, California, April 19-21,1999.

8. Solaris 2.6, TCP/IP and Data Communications Administration Guide. Sun Microsystems, Mountain View, California,1997.

9. Stevens, W. Richard. UNIX Network Programming. Prentice Hall, Englewood Cliffs, New Jersey, 1990.

10. Sun Workshop, C User's Guide, C Compiler 4.2. Sunsoft, Inc., Mountain View, California, 1996.

Notices