Gonzalo Remedi et al.: Remote Control of VLBI Operations, IVS 2006 General Meeting Proceedings, p.127–131
http://ivscc.gsfc.nasa.gov/publications/gm2006/remedi

Remote Control of VLBI Operations

Gonzalo Remedi [1], *Sergio Sobarzo* [2], *Hayo Hase* [3]

[1)] *Universidad del Bío Bío - TIGO*
[2)] *Universidad de Concepción - TIGO*
[3)] *Bundesamt für Kartographie und Geodäsie - TIGO*

Contact author: Gonzalo Remedi, e-mail: gonzalo.remedi@tigo.cl

Abstract

The Geodetic Observatory TIGO developed an easy-to-use software which enables remote control and monitoring of VLBI operations. This software is a Java-applet and can be executed from any web browser. Several tests have been successful. The remote control operation of VLBI opens new observation strategies involving more observation without additional staff by making use of operators in different time zones operating more than one radiotelescope.

1. Introduction

The VLBI network consists of many geographically distant stations. Remote control of a VLBI station can provide a distributed operation supporting other stations in different time zones. Having a system that is able to automatically check the general status of the ongoing experiment can be helpful to respond to warnings or errors quicker than using only the log file. Remote trouble shooting is also considered to a certain extent. In the present document a remote control VLBI software with all its implementation issues is explained. The developments are described in detail in Remedi [1].

2. Objectives

The main idea of a VLBI remote control software is to give the ability to control a radiotelescope station from anywhere and from any platform without the need of installation of any kind of software on the client computer.

The location of some remote VLBI stations implies that the Internet connection has a limited bandwidth. In addition computers are not always fast enough to run heavy programs. These conditions suggest a second objective which is the use of minimal network and computer resources.

The third objective is the development of a program of easy and very intuitive use in order to facilitate the manipulation by people with less experience and to have a fast diagnosis of the current experiment with just one look at the main client screen.

3. Implementation

3.1. Servers

At VLBI stations, the so-called VLBI Field System Software (FS) is widely used for VLBI operations. Our software is based on additional servers, which interact with FS variables. These

servers retrieve actual values from the shared memory area, before they pack and send them through the network as it is shown in Figure 1.

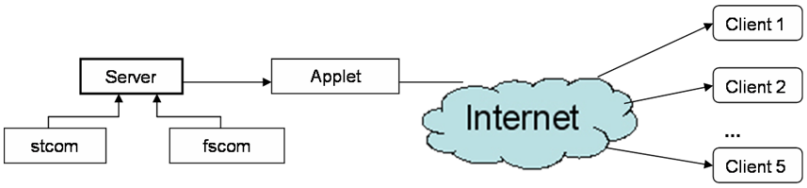

Figure 1. Communication flow diagram between servers and clients on the Internet [2].

Currently there are 4 servers. Each of these starts together with the FS and waits for incoming connections from the remote clients. The value of the maximum simultaneous connection can be modified according to the processor and network capabilities (see also [3]).

3.1.1. Monitoring Server

The monitoring server takes a selection of FS parameters containing the most relevant status information from the FS and station shared memory and put them into a formatted TCP frame which is sent to the client computer. A frame of formatted data is sent to the connected clients each second. This guarantees that the network link utilization is efficient, and as minimal as possible, and that the remote client receives permanently 1 second samples of the status.

3.1.2. Log Server

The log server only sends the changes of the log file every second. It continuously scans for changes in the size of the log file and then transmits the most recent changes to the client using minimum network resources.

3.1.3. Control Server

The control server is always waiting for incoming frames sent from the client with FS commands inside. This program receives the command and then sends it to the FS main program where the command is processed and executed.

3.1.4. Chat Server

The chat server resides in the web page server and is written in Java. It takes incoming messages from the chat applets connected and distributes them to the other connected clients. This allows for discussions between remote (and local) operators. It is needed to pass the control from one client-operator to the other.

3.2. Clients

The clients were implemented in Java as an applet. A screen shot is shown in Figure 2. Each server corresponds to one client, which means we have actually also four clients: monitoring,

control, log and chat.

The main advantage of Java is that you can execute the application on any platform, regardless whether it is Windows, Linux, Mac or any other OS (Figure 3). A typical Java applet works in the following way:

1. A web server offers a Java code. (the Java codes have the extension *.class).
2. Somebody on the Internet, with a Java compatible web browser, makes a connection to the server.
3. The server sends the HTML document and the Java code (*.class).
4. Both (the HTML document and the Java code) arrive at the remote computer of the user that made the connection, and then the Java Virtual Machine inside the browser transforms the Java code into a code that is understood by the operating system of the local machine and executes the program inside the web page.
5. If the user makes a connection to another URL or quits the browser, the client program terminates its execution at the remote computer.

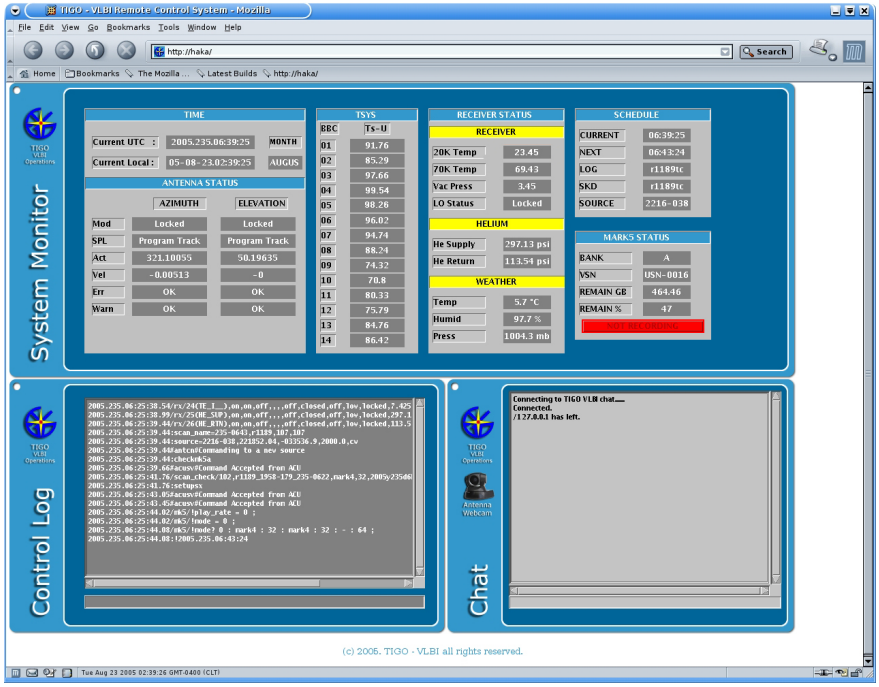

Figure 2. View of the Remote Control applet in a browser.

3.2.1. Monitoring Client

The function of this client is to show the received data of the Monitoring Server. It is an applet. It consists of four panels, each one with specific data of the shared memory.

The first panel is used to monitor the antenna status, UTC and local time.

The second panel shows the system temperatures derived from the SQL-detectors of all base-band converters.

The third panel shows the receiver status with temperatures of the cold stages, helium pressure, LO-status and the meteorological parameters temperature, pressure and humidity.

The fourth panel shows the status of the running experiment with schedule and source name, current and next scan.

The last panel shows the status of the Mark 5 recording system.

3.2.2. Log Client

The log client is in charge of receiving data sent by the log server. In fact, what it does is receive a package of data sent by the server and then display it in a text area. Hence it displays all the log entries, as they are seen on the local screen of the FS computer. The given information is useful to identify errors in the execution of commands.

3.2.3. Control Client

The control client is in charge of sending the commands entered by the remote operator to the control server.

The server accepts commands in SNAP language. The main process in the FS determines the validity of a SNAP command. The remote operator is informed about the response of the FS, by the usual log-entry wich can be seen in the log client panel.

3.2.4. Chat Client

The chat client allows the communication between the users who are connected to the application (applet).

For security reasons the web page and the applets must reside on another computer than the FS computer. But Java doesn't allow the execution of an applet if this tries to connect to another machine which is different to the applet host machine. This problem was solved by a redirection from the client via the applet host computer to the FS-computer by using IP tables.

It implies also a modification to Java security policy. The configuration file in the web page host machine needs to enable all the network permissions: accept, connect, listen and resolve for each of the ports used by the remote connection, for example:

```
permission java.net.SocketPermission "IP_of_the_remote_machine:port1", "accept, connect, listen, resolve";
permission java.net.SocketPermission "IP_of_the_remote_machine:port2", "accept, connect, listen, resolve";
permission java.net.SocketPermission "IP_of_the_remote_machine:port3", "accept, connect, listen, resolve";
permission java.net.SocketPermission "IP_of_the_remote_machine:port4", "accept, connect, listen, resolve";
```

These modifications are only possible if we have the necessary safety measures to avoid an undesired access. In this case the main filter is the firewall where the connections to the application are authorized only to specified IP addresses. A further access restriction can be realized by username and password.

4. Future Work

It is possible to implement the servers in the FS official distribution. The first step is to define a standard set of variables for monitoring and include them into the FS shared memory area, as

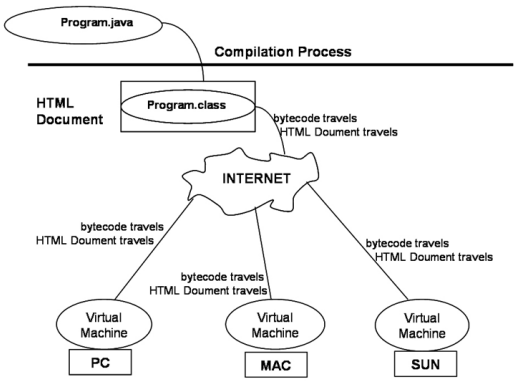

Figure 3. Each platform has its own virtual machine that allows the execution of a program in Java. The program is executed locally on each computer.

it can be done following our example. The second step is that every station must fill the correct values in the shared memory modifying the station software.

Other improvements that can be added are the addition of new key variables to the monitoring scheme and the use of sound alerts in the applets. Also automatic log comment generation can be implemented.

Since the monitoring can not handle all the possible errors of a station, it is possible to extract the last error from the log file and show it permanently.

Visual and acustic guidance of the remote operator can be provided by adding webcam or videoconferencing capabilities, which are currently under development.

4.1. Conclusions

With this application it is possible to share the operators load between co-observing VLBI stations and to avoid night shifts when control can be passed to an operator in a daylight time zone.

The advantages of the open source software tools were used for the development of this application. This avoided costly development tools like Microsoft technologies.

The appliaction is fully functional now. It is possible to integrate other features depending on the specific needs of the VLBI station and the operator's experiences.

References

[1] Remedi, G. A., Sistema de Control Remoto para el Monitoreo de un Radiotelescopio, Computer Engineer Thesis, 2005.

[2] Stevens, W. R., UNIX Network Programming, Volume 1, Second Edition: Networking APIs: Sockets and XTI, Prentice Hall, 1998.

[3] Stevens, W. R., UNIX Network Programming, Volume 2, Second Edition: Interprocess Communications, Prentice Hall, 1999.