

A concept for remote control of VLBI-telescopes and first experiences at Wettzell

A. Neidhardt, M. Ettl, R. Zeitlhöfler: FESG, TUM, Geodätisches Observatorium Wettzell, Sackendorfer Str. 25, D-93444 Bad Kötzing, Germany

C. Plötz, M. Mühlbauer, R. Dassing, H. Hase: BKG, Geodätisches Observatorium Wettzell, Sackendorfer Str. 25, D-93444 Bad Kötzing, Germany

S. Sobarzo, C. Herrera: UdeC, Camino Einstein Km 2,5., Casilla 4036, Correo 3, Concepción, Chile

W. Alef, H. Rottmann: MPIfR, Auf dem Hügel 69, 53121 Bonn, Germany

E. Himwich: NASA/GSFC, Greenbelt, MD 20771, USA

Abstract. The requirements for VLBI systems are increasing: higher observation density schedules, real-time access for changing schedules, more automation of observations and remote control of complete sites are being planned. To support these changes new additional software components are required. The addition of (semi-) autonomous, remote accessible control features, which are becoming a reality now will provide needed support by offering reliable, safe, and modular structures from the high-level controlling layers down to the basic equipment interaction elements. An extension to the current NASA Field System (FS) with remotely accessible, autonomous process cells is being developed at the Wettzell Geodetic Observatory. It uses the specially designed middleware generator “idl2rpc.pl”, developed at Wettzell, to generate the remote C++ interfaces for communication issues. A new modern graphical user interface in combination with an initial programmatic interface to the FS, both developed as extensions, demonstrate the capability for controlling radio telescopes remotely. The first successful remote control tests, with operators present, during regular experiments with the telescopes at O’Higgins, Concepción and Wettzell have demonstrated that this approach works well in the global communication network.

Keywords. VLBI, field system, automation, remote control, e-control, idl2rpc, middleware

1 Introduction

Personnel staff from the Geodetic Observatory Wettzell operate not only the 20 meter radio telescope at Wettzell. In cooperation with other

institutes they also run the 9 meter radio telescope at the German Antarctic Receiving Station (GARS) O’Higgins, Antarctica and the 6 meter radio telescope of the Transportable Integrated Geodetic Observatory (TIGO) Concepción, Chile for geodetic VLBI experiments. Because of the remote locations especially in case of the telescope in the Antarctica and because of the increasing requirements for such observations, e.g. relating to the number of experiments, a first concept was developed, to control sites remotely. This is also an appropriate starting point for the control modes of the new VLBI2010 TWIN radio telescope at Wettzell.

The current equipment at VLBI radio telescopes is controlled by the NASA Field System (FS) software package, which is a very stable, well known and well supported package. But the current realisation of the FS has some deficits according to that new possibilities of remote control and remote attendance. Already existing tools to forward mouse, keyboard and video signals are also suboptimal, because they don’t allow monitoring of the internet connection itself. This is especially important for safety issues at a site to facilitate safety actions when no responsible observer is connected anymore. Therefore it is necessary to extend the given controlling mechanism by a Ethernet based, safe and stable remote communication. Since most of the new devices controlled by the field system are also connected via ethernet mechanisms the new concept also includes ideas to standardize such individual communication needs. Together it offers the appropriate elements for remote control as a new VLBI observation mode, possibly named “e-control”.

2 The layers of e-control

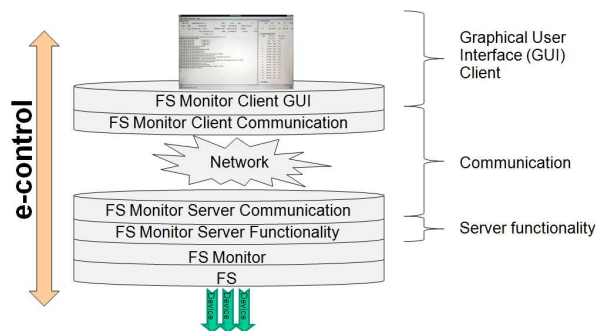


Figure 1. The complete e-control stack.

The given remote control design realises a classic client-server-model on the basis of a communication with Transmission Control Protocol over Internet Protocol (TCP/IP) or User Datagram Protocol over Internet Protocol (UDP/IP). In such a client-server-model a service requesting client starts the communication and sends an order request via message communication to a service offering server. At the server side the order is processed and an answer message is returned to the client (Singhal et al., 1994). The client-server-based complete stack for e-control from user interaction via remote procedure call communication to the FS interaction is shown in Fig. 1.

One of the main drivers in the given design is a consequent separation of control, communication and presentation logic. The complete arithmetic and workflow control logic reside in the server, defined as device control code. It is an autonomous working process which interacts with the remote controlled device (here at this level, the FS). The communication code independently connects the server to the outer world for requesting clients. And the clients are only used to realize an user interface with a presentation of the server processed elements.

2.1 The communication middleware with remote procedure calls

In classic communication networks each client server interaction is programmed individually during the software development process. This proprietary approach makes it more difficult to

set up a remote control and to include or adapt new properties of remote usable devices. Another more modern attempt reduces the efforts of communication programming by defining a standardized way for the transmission of so called remote procedure calls (RPC). RPCs are comparable to local calls of procedures (functions) in a structured program but realized as control and data flows over a communication network to allow a standardized interaction between a requesting client and a service offering server (Singhal et al., 1994). The client just calls a procedure or function without the concrete knowledge of the processing location and an additional RPC communication layer realizes the transfer between the client and the remote processing server. The derived answer follows the same way vice versa to the client, so that the procedure call seems as local (see Fig. 2).

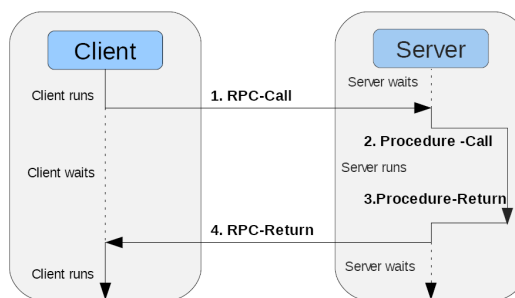


Figure 2. Sketch of the remote procedure call client-server-model (Saxonia Systems, 2007).

To reduce the programming effort the complete communication layer within this model is created by a special RPC generator which reads an interface definition file and produces all of the necessary modules. They can directly be used in the application code. So the application programmer has only to think about his application specific code and not about the lower-level data transfer parts. Modern networking techniques know several realizations of the RPC because this communication method is also the basis for the distribution platforms of modern web services. These distribution platforms are also known as distributed systems. A distributed system consists of several independent computers (processors), which are connected together to solve a collective task in a cooperative way. During the processing time they don't share mem-

ory, clocks or other hardware and just communicate information while transferring messages via a computer network (Singhal et al., 1994 and Puder et al., 2001).

For the planned remote control of the FS, basic and compact realizations are more flexible and reliable than huge, sophisticated, additional communication packages, which are also in most cases commercially offered. Several basic RPC-realizations are well known over years. In the given context the Open Network Computing Remote Procedure Call (ONC RPC) is a preferred communication technique because it is available in each Linux operating system as standard and has been well tested since the year 1988. To generate the actual communication code units of RPC in the programming language C, the generator “rpcgen” is used which is also part of the Linux operating system. The generated code also includes the platform independent conversion of procedure parameter data using the External Data Representation (XDR; Stevens, 1992).

To update these interface paradigms for modern object oriented implementations an additional C++ layer over the C coded communication was added and to include more sophisticated communication control mechanisms a new, very straight forward code generator “idl2rpc.pl” is used. This code generator is based on a script written in Perl language containing C++/C code templates. The templates are filled using remote function definitions written in a specific Interface Definition Language (IDL) as a high level description of a remote interface. This definition looks similar to C function headers, so that it is easily understandable. The generator converts the IDL description into standard RPC equivalent code already supported by the Linux operating system. Several C++ adaptor classes for the C written RPC communication are created as well as necessary modules for threads to use in parallel tasks and with semaphores to protect critical sections. In only a few steps the complete communication is generated (see Fig. 3). For the application developer only a few files are important to edit while the rest automatically realize the communication.

With this approach the application programmer doesn’t have to consider communication concerns and can concentrate on the actual application tasks. But he should use the possibilities on server side like threaded periodic loop activ-

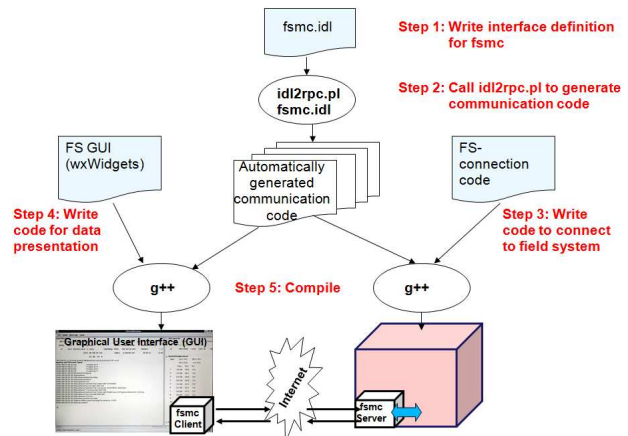


Figure 3. The generation process using the code generator “idl2rpc.pl”.

ities, etc., to realize independent servers which always keep stable states. The servers can be located wherever they are needed to realize the structures of a distributed system. For reliability each server contains something like a watchdog process which always restarts it after an unexpected crash. In addition to that an automatic safety device (ASD) is implemented which expects periodic requests from responsible clients. If this “alive”-signal is not detected, the server can activate a specific routine which leads the system into a stable and safe state again (Neidhardt, 2009).

In case of the FS an additional client-server-communication is defined using the new idl2rpc.pl. In the first implementation it consists only of a few methods returning the local information output as string arrays to the remote requesting client. The client can also send a string command to the server. The server itself is the main part of the proposed extension to the FS.

2.2 The field system extension as a remote accessible, autonomous process cell

The server skeleton is created automatically and is completed with functions to communicate to the existing FS. Therefore an additional C-written adapter (FS monitor) allows the connection to the FS via shared memory access. The commands are injected into FS using the supported injection methods. To smooth the communication behaviour the server uses threads to separate between the asynchronous remote

procedure calls and the contact with the FS. Semaphore protected variables allow the handling of critical sections when both tasks work in parallel with the same variable values.

Overall the created server acts completely autonomously. It can be used to check system status information independently and can decide what to do to keep stable and safe states. Such controlling utilities can be defined as autonomous process cells. The generated watchdog process keeps it alive and the automatic safety device allows to register if a responsible client is connected. After a breakdown of the communication to the client the server can operate completely autonomous until a critical situation (e.g. increase of wind speed to a level which is critical for the telescope) forces it to run into a safe state. In combination with additional monitoring information around the site (meteorology, power supply, air conditioning status, etc.) that compact server extends the FS for a reliable remote control.

2.3 The field system client as a remote graphical user interface

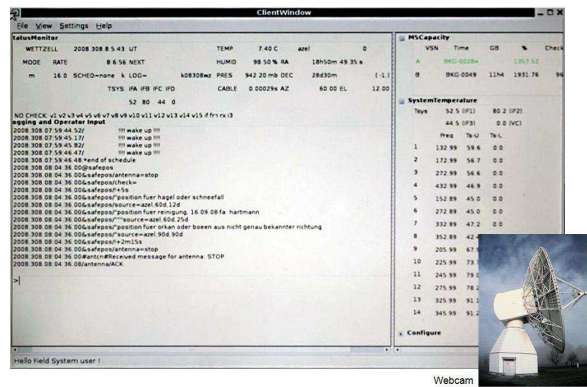


Figure 4. The graphical user interface on the basis of wxWidgets.

On the other end of the remote control the operator interacts with the system. For that it is possible to use different and parallel user interface clients. Because of the consequent separation of control and presentation logic the client can be realized in different ways. So it is possible to implement command line clients as well as high sophisticated web applications or graph-

ical user interfaces. This permits management of devices remotely via browser, command line and/or graphical user interface.

For a first general realization all servers can have a rudimentary command line control. In addition to that it is useful to offer a graphical user interface. For the described FS extension wxWidgets is the preferred way to do this. It is a C++ based open source framework for platform independent developments of graphical user interfaces (Smart et al., 2005). Although the current RPC generator only supports Linux systems (32 and 64 Bit), the graphical user interface is modular enough to support different platforms like Windows, Linux, OSX and others. In terms of the proposed FS extension a new graphical user interface was created using the wxWidgets framework for its realisation (see Fig. 4). To keep usage similar, the display elements are organized to be like those of the current local interface to the FS.

2.4 Safety and security

To protect humans and the system itself safety and security concepts are in development in addition to the inline safety functionality of the generated communication. Safety hereby means the local protection against local error states or critical situations, possible for automatically moving hardware. For the basis of an autonomous and stable software application, the programmer should follow previously defined design rules. For example, at Wettzell the developed elements follow the Wettzell design rules, which describe how code must be structured, documented, commented and what is allowed. But no software is good enough to have no bugs. So an additional, modular and multi-layered system monitoring hardware is in production which should check all of the important system states, like temperatures, weather conditions, safety switches and so on, parallelly to the FS. This hardware is realized with standard equipment on a robust, well known architecture and supports several individual, vendor independent sensor devices. It is created with open source products in combination with Linux operating systems (also with a minimal installation) and implements internally also the "idl2rpc.pl" created communication system. So it is an additional parallel monitoring system for safety reasons, also used for emergency issues.

Security however means the protection of the

system from not allowed activities done by foreign attackers or users without the sufficient right policies. All of the communication activities are based on simple socket communications with TCP/IP or UDP/IP with fixable ports on which the additional RPC layer is established. To bring in an additional level of security the Secure Shell (SSH) tunneling methods can be used to build up an access protection. SSH hereby allows several authentication possibilities like passwords, passphrases and key files or combinations of them. For the internal correct access for operator actions it is planned to realize an authentication (registration of a user with username and password) and authorization (personification of a user for a specific remote procedure with dedicated rights) similar to what is already implemented in other projects at Wettzell (Neidhardt, 2006). This should allow a first attempt of safe and secure remote control from almost any place.

3 Remote control tests

To prove the functionality of the remote control and the general character of the implementations as well, several tests were initiated to run the radio telescopes operated by Wettzell with the described software. Several 24 hour and 1 hour intensive experiments were successfully run by remote control also at the very remote site O'Higgins. These tests will be extended and will lead to routine remote operation of VLBI experiments at Wettzell.

4 Summary

Overall the described method allows the development of distributed systems consisting of several independent servers which act completely autonomously. It extends given structures to have a remote control possibility and splits complex systems up into several manageable units interacting together with a general, standardized but also flexible communication method. The concept can also be used to update the internal structures of the FS to connect it to a future network of controllable instruments. It offers a possibility to add devices like the new Digital Baseband Converters (DBBC) with standardized and stable methods to the FS.

The described software concept is a product of a long development done by several developers at Wettzell. The result is an option for up-

coming Fundamental Stations with several different colocated measuring systems like radio telescopes and laser ranging systems to realize remotely controllable, autonomous subsystems on a basis of a stable, flexible and general communication platform. It can be used to reduce development time for highly available systems especially along the goals of the Global Geodetic Observing System (GGOS). New observing strategies proposed in VLBI2010 can be realised also with very remote stations. But nevertheless there are always some situations which cannot be controlled and handled by such an automated system (like power failures where e.g. the laser telescope dome is not closed automatically), so that responsible, well educated engineers at the sites should always be the final instance of automation.

References

- Neidhardt, Alexander: Verbesserung des Datenmanagements in inhomogenen Rechnernetzen geodätischer Messeinrichtungen auf der Basis von Middleware und Dateisystemen am Beispiel der Fundamentalstation Wettzell. Dissertation, Mitteilungen des Bundesamtes für Kartographie und Geodäsie, Nr. 37, Bonifatius GmbH 2006
- Neidhardt, Alexander: Manual for the remote procedure call generator "idl2rpc.pl". Geodetic Observatory Wettzell 2009 (latest version)
- Singhal, Mukesh; Shivaratri, Niranjana G.: Advanced Concepts in Operating Systems. McGraw-Hill, Inc. 1994
- Puder, Arno; Römer, Kay: Middleware für verteilte Systeme. 1. Auflage. dpunkt-Verlag GmbH Heidelberg 2001
- Saxonia Systems: Remote Procedure Call, <http://www.linuxfibel.de/rpc.htm>, Download 2007-04-23
- Smart, Julian; Hock, Kevin; Csomor, Stefan: Cross-Platform GUI Programming with wxWidgets. Prentice Hall International 2005
- Stevens, Richard W.: Programmieren von UNIX-Netzen. Grundlagen, Programmierung, Anwendung. Prentice-Hall International, Inc. London 1992