New concepts in control systems for SLR with remotely accessible, autonomous process cells

Alexander Neidhardt[1], Martin Ettl[1], Pierre Lauber[1], Andreas Leidig[1], Reiner Dassing[2], Matthias Mühlbauer[2], Christian Plötz[2]

[1]Forschungseinrichtung Satellitengeodäsie, Technische Universität München, Geodätisches Observatorium Wettzell
[2]Bundesamt für Kartographie und Geodäsie, Geodätisches Observatorium Wettzell

Abstract

The demands for data acquisition systems in space geodesy like satellite laser ranging systems increase continuously. Shared observations of future transponder targets, the distribution of the resulting data in real-time and the increasing number of possible targets with fast switching between passages will offer new possibilities but will also require new solutions for the controlling software. In response to these demands (semi-) automated, remote control systems will become more and more reality. Such complex systems require reliable, transparent and modular structures from upper controlling layers down to the basic single components in combination with sophisticated safety mechanisms in automation. A new idea with remote assessable, autonomous process cells, which can solve component specific requirements encapsulated and autonomously, can generally help to split up the complex system into maintainable, modular units. These units are remote controllable and highly automated. In the design of the new Satellite Observing System Wettzell these ideas have been implemented. A similar approach has been proposed for the remote control of the radio telescope at the German Antarctic Receiving Station O'Higgins for geodetic VLBI experiments, where the first tests are successfully done.

Satellite laser ranging systems as distributed computer systems

Modern laser ranging systems, like the newly designed Satellite Observing System Wettzell (SOSW) at the Fundamentalstation Wettzell, consist of numerous, separate components for single, specific tasks. In most cases each part runs as an independent, autonomous hardware like a processing unit or memory and with an independent timing. Only for realtime dependent aspects the internal clocks of the specific components are synchronized to the time standard of the site. All of the components are connected together via local networks or other communication links, like serial lines, to allow a synchronous performance of the entire laser ranging process.

Because of this given situation modern laser ranging systems can be described as distributed systems in terms of computer science, where 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 memory, clocks or other hardware and just communicate information while transferring messages via a computer network [SING94]/[PUD01].

Within a laser ranging system the following independent components can be identified:

- the telescope
- the dome

- the laser
- the transmit and receive unit
- the timing system and eventtimer
- different cameras
- a database which interacts with the outer world, like data centers
- the control system

and

- an additional, independent system monitoring for security and safety reasons

The human operator interacts with all of these components (mainly in a hierarchical structure via the control system but in some specific cases, e.g. development tests, also directly). The complete system itself is then a direct representation of the identified, independent, but interacting components. Therefore each hardware device is represented as a software component connected together via a 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 server side the order is processed and an answer message is returned to the client [SING94].

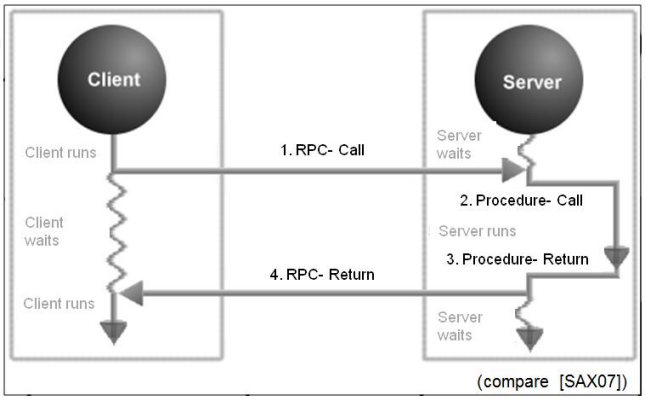

Figure 1. Remote procedure call

In classical communication networks each client server interaction is individually programmed during the software development process. Another attempt reduces the efforts of communication programming by defining a standardized way for the transmission of remote procedure calls (RPC). RPCs are comparable to local calls of procedures 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 [SING94]. Then the client just calls a procedure or function without the knowledge of the processing location and an additional RPC communication layer realizes the transfer between the remote processing server and the client. The response follows the same way vice versa to the client, so that the procedure call appears to be local (see figure 1). 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. These can directly be used in the application code. There are several realizations available because this communication method is also the basis for the distribution platforms of modern web services.

But for the challenges of simplifying the communication within distributed systems for laser ranging, low-level realizations are more flexible and reliable than huge, sophisticated, additional communication packages. Here the Open Network Computing Remote Procedure Call (ONC RPC) is the preferred communication technique, because it is available in each Linux operating system as a standard and well developed 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 operating system Linux. The generated code also includes the platform independent conversion of procedure parameter data using the External Data Representation (XDR) [STEV92]. So there is only the necessity to update the interface paradigms for modern object oriented implementations as an additional C++ layer over the C coded communication. This is done with a self-made additional generator.

The RPC-generator "idl2rpc.pl" and the usage of autonomous process cells

The new generator "idl2rpc.pl" is based on a script written in Perl language containing C++/C code templates, which uses a specific Interface Definition Language (IDL) as a high level description of an interface. So it converts the IDL description into RPC equivalent code using also the RPC environment already integrated in the operating system Linux . Several C++ adaptor classes to the C written RPC communication are created as well as the needed modules for threads creating parallel tasks or semaphores to protect critical sections (see figure 2). But for the application programmer only the following files are important where he has to include his specific code while the rest is just internally used for the communication:

On client side:
- "<INTERFACEFILENAME>[1*]_client.hpp" contains the class with the interface methods for remote procedures
- The client main program (e.g. a graphical user interface) which must be written completely by the application programmer

On server side:
- „<INTERFACEFILENAME>[2*]_server.cpp“ contains the skeleton methods where the application programmer has to include his own code into the method body

In this way the application programmer doesn't have to worry about communication matters but can concentrate on the actual application tasks. However he should use the provisions on the server side like threaded periodic loop activities etc. to realize independent servers. These should always keep stable states e.g. for the hardware devices connected to them . The servers can be located wherever they are needed so that they set up a distributed system. For stabilization each server contains something like a watchdog process which always restarts it after an unexpected crash. And it is planed also to include something like a "dead man's handle" comparable to trains. There it is used to check if the driver is still able to control the train. So he has to push a dedicated button periodically. In the distributed application a similar concept is used to check at server side if an active client exists. Therefore each client has to activate remote server functions periodically. These automatic safety devices can be activated optionally and starts a specific server routine which can lead to a stable and safe state again after a case of interaction failure [NEID08].

[1] [1*],[2*] <INTERFACEFILENAME> is replaced by the actual filename of the interface definition file

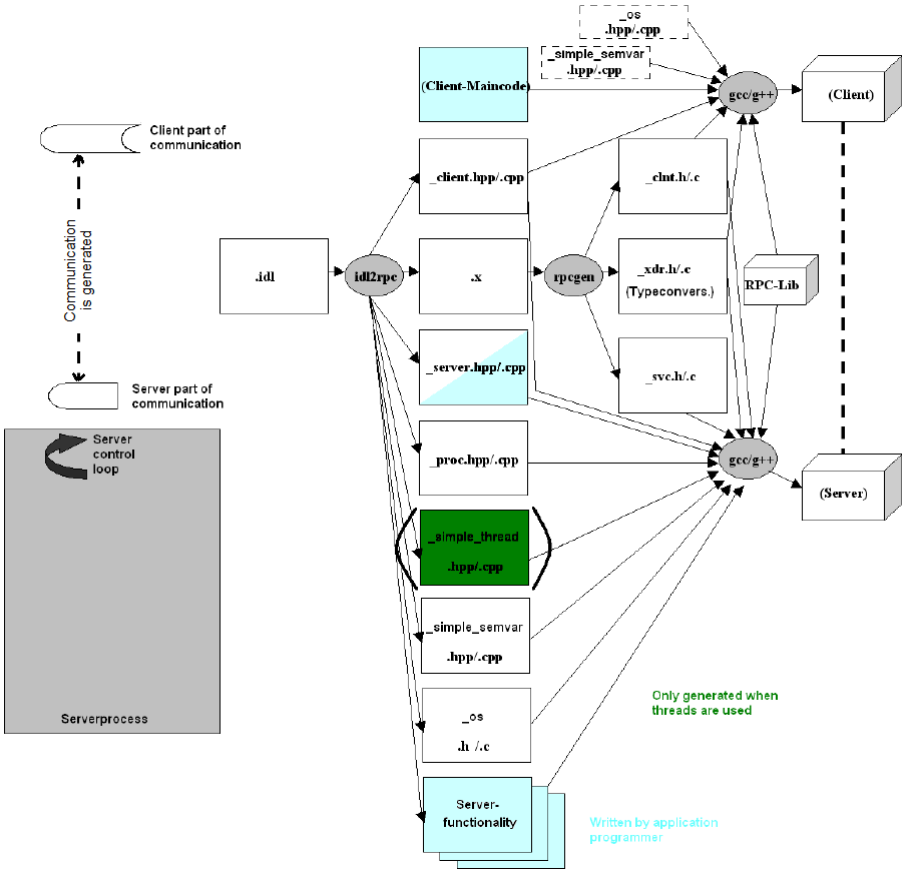

Figure 2. The generation process of "idl2rpc.pl" and the created files

All in all such method allows the creation of a distributed system consisting of several independent servers which act completely autonomous, so that these units are called autonomous process cells. These splits up a complex system like a laser ranging system into several manageable units interacting together with a general, standardized but also flexible communication. And in this communication all user or operator interactions can be included as additional clients realized as more or less complex user interfaces.

User interaction

The generated systems allow the integration of several different and parallel available user interface clients with different styles. This is because of the consequent separation between control and presentation logic. So it is possible to implement command line clients as well as high sophisticated web applications or graphical user interfaces. All of these possibilities are realized at SOSW so that for example the dome controller can be managed via browser window, command line and/or graphical user interface. And because of the automatically generated communication layer all of the user interfaces can directly be used to control the applications remotely.

For a first general realization all servers provide a command line control and a graphical user interface on the basis of wxWidgets. It is a C++ based open source framework for platform independent developments of graphical user interfaces [SMAR05]. 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. So a remote

operator can run the applications with highly sophisticated graphical interfaces. Because of the physical separation between the operator and the system it must be realized such that it protects human beings and the system itself from dangerous and error-prone situations.

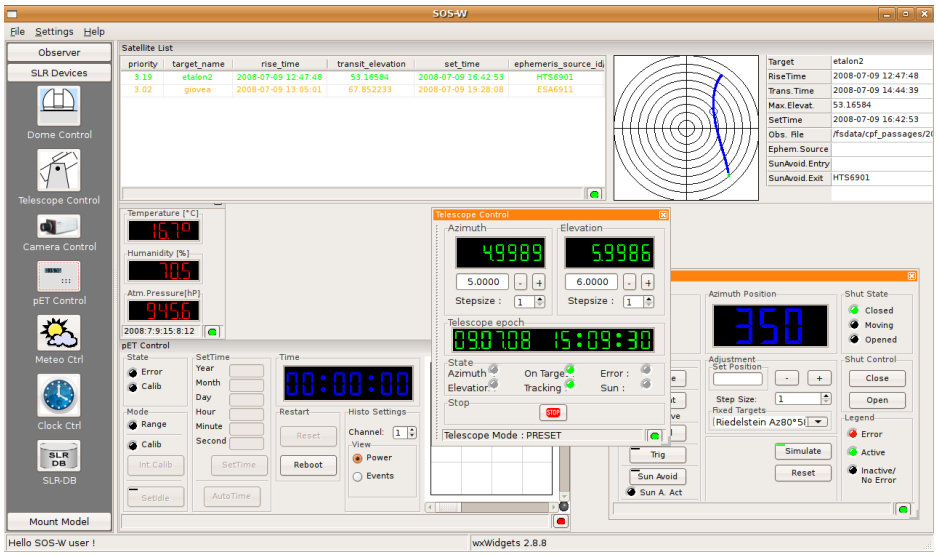

Figure 3. The graphical user interface on the basis of wxWidgets

Safety and security with the system

To realize such a protection safety and security concepts are in development. Safety hereby means the local protection given by local error states or local situations, like automatically moving hardware. One way to realize this is to write stable and autonomous software as given in the distributed process cells. In addition to that it is an advantage when the programmers follow some design rules as defined at Wettzell. They describe in general how code must be structured, documented, commented and so on. But no software is safe enough to have no bugs. So an additional, modular and multi-layered system monitoring hardware is in production which checks all of the important system states, like temperatures, weather conditions, safety switches and so on. This hardware is realized with standard equipment on a robust, well known architecture and supports several individual, vendor independent sensor devices. It is based on open source products in combination with the 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 to ensure safety, also dealing with emergency issues.

Security in this case means the protection of the system from not allowed activities by unauthorized attackers or users without sufficient access right. Because all of the communication activities are based upon simple socket communications with TCP/IP or UDP/IP with fixable ports on which the additional RPC layer is established, Secure Shell (SSH) based tunneling methods can be used to build up efficient access protections. SSH hereby allows several authentication options like passwords, passphrases and key files or a combination of them. For the internal access right control 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) as already implemented for other projects at Wettzell [NEID06]. This allows a safe and secure remote control from almost any place.

Remote control

To proof the functionality of the remote control and the general character of the implementations as well, several tests were initiated to run radiotelescopes for geodetic Very Long Baseline Interferometry (VLBI) also with the described software. Therefore adapted servers and clients were programmed to monitor the VLBI fieldsystem, which controls the VLBI experiments. Test sites were the radiotelescopes of the German Antarctic Receiving Station (GARS) O'Higgins in Antarctica, the Transportable Integrated Geodetic Observatory (TIGO) at Concepción/Chile and Wettzell. Several 24 hour and 1 hour intensive experiments were successfully run by remote control. These tests will be extended and will lead into a routinely operation for VLBI experiments at Wettzell. Co-operations with other institutes like the Max Planck Institute for Radioastronomy Bonn and the developers of the fieldsystem for VLBI are the first steps to establish this concept in the VLBI community.

Summary

The newly created software concept is a product of a long development done by several developers at Wettzell. The result is an option for upcoming Fundamentalstations with several different measuring systems like SLR systems and radiotelescopes 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). But nevertheless there are always some situations which cannot be controlled and handled by such an automated system (like power failures where the dome is not closed automatically), so that responsible, well educated engineers at the sites should always be the final instance of automation.

References

[NEID06] 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

[NEID08] Neidhardt, Alexander: Manual for the remote procedure call generator "idl2rpc.pl". Geodetic Observatory Wettzell 2008 (latest version)

[SAX07] Saxonia Systems: Remote Procedure Call, http://www.linuxfibel.de/rpc.htm, Download 2007-04-23

[SING94] Singhal, Mukesh; Shivaratri, Niranjan G.: Advanced Concepts in Operating Systems. McGraw-Hill, Inc. 1994

[SMAR05] Smart, Julian; Hock, Kevin; Csomor, Stefan: Cross-Platform GUI Programming with wxWidgets. Prentice Hall International 2005

[STEV92] Stevens, W. Richard: Programmieren von UNIX-Netzen. Grundlagen, Programmierung, Anwendung. Prentice-Hall International, Inc. London 1992

[PUD01] Puder, Arno; Römer, Kay: Middleware für verteilte Systeme. 1. Auflage. dpunkt-Verlag GmbH Heidelberg 2001