Data Acquisition Test Report

Version: 31.8.2009

Helge Rottmann

Max-Planck-Institut für Radioastronomie
Auf dem Hügel 69
53121 Bonn

Table of contents

1 Introduction

This report describes the tests performed with the prototype VLBI remote control system as described in Rottmann, Neidhardt et al. 2009. The main goals of the tests were to proof the feasibility of remote VLBI observations, demonstrate the functionality of the implementation and to collect information about the performance of such a system.

2 Installation and code verification

Tests were performed to verify the consistency between the “traditional” VLBI field system and the new client/server based field system extension (XFS hereafter). For the tests the XFS-Server was installed on the field system workstation in Effelsberg. A simple XFS-Client was run on a computer located at the MPIfR in Bonn (see Fig. 1).

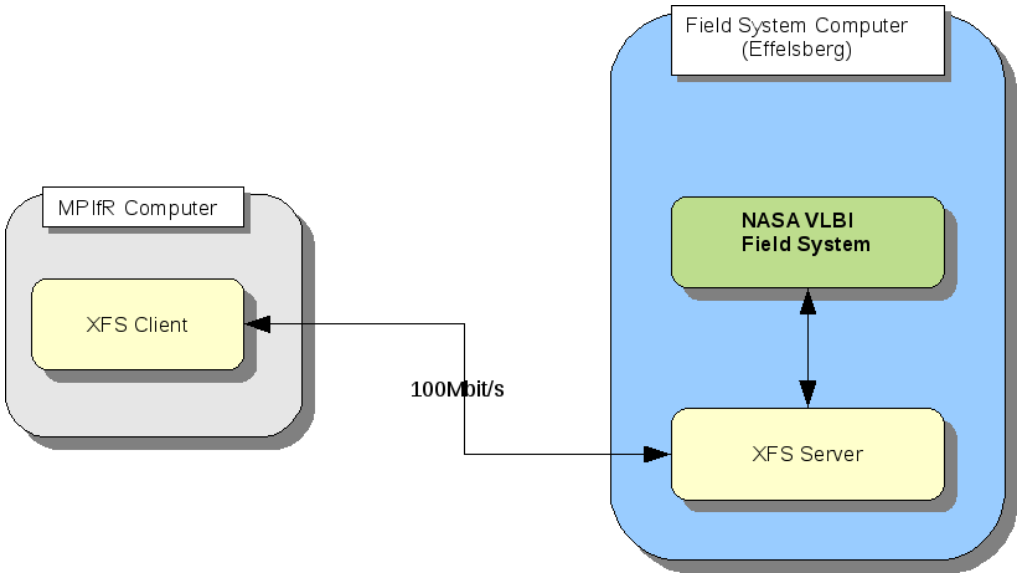

Figure 1: Sketch of the test setup. The XFS server was installed and started on the field system computer in Effelsberg. The XFS client was installed and run on a computer located at the MPIfR in Bonn.The client and server were connected over a 100 Mbit/sec line.

Installation of the XFS client and server software is straightforward and worked flawlessly. The software is supplied as a tar archive. After unpacking type:

```
configure
make
make install
```

The installation process will automatically determine the required system information (e.g. the installation root of the field system software) and will adopt default values for certain parameters

(e.g. the TCP and UDP ports). All parameters can be changed by supplying appropriate configure options (run configure –help for additional information).

The XFS-server currently provides field system state information identical to the native field system programs *monit2* (general status information), *monit3* (system temperatures), and *monit5* (Mark5 info). In the default setup the server queries the field system shared memory every second. The server exposes the field system log and accepts commands that will be passed on to the field system for execution.

The XFS-client makes a RPC calls to the server to obtain the field system state and outputs all information in a console. Formal equivalence between the output provided by the native field system and obtained via the XFS client/server was verified.

3 Connection stability

We have investigated how well the XFS prototype copes with interruptions of the client/server communication. We have looked at the following scenarios:

server stop & restart

server crash

client stop & restart

network failure

ssh tunnel shutdown (in case ssh tunneling is used for client/server connections).

The XFS-client is implemented to automatically try to re-establish a connection to the server in case a connection failure has been detected.

3.1 Server stop & restart

After the client connection to the server was established, the XFS server was killed. After restarting the server manually, the client automatically reconnected and instantaneously resumed operations.

Inspection with the *netstat* and *lsof* tools has shown that no hanging socket connections have remained either on the server or client side.

3.2 Server crash

A server crash was caused by deliberately creating a division by zero exception in one of the RPC routines that is called by the client. The server automatically restarted and the client could resume operations. The server restart is handled by a watchdog thread that is autonomously by the communication interface generated with the *rpc2idl.pl* generator (see Neidhardt 2008). The server crash did not leave any hanging sockets.

3.3 Client stop & restart

The client was stopped and restarted manually. All socket connections were terminated properly and after restart no hanging sockets were observed.

3.4 Network failure

In order to cause network failure, the network cable was disconnected from the client computer

after the client-server connection has been established. After reconnecting the cable the client and server automatically reconnected and resumed operation. Again no hanging sockets were left either on the client nor on the server side.

3.5 SSH tunnel shutdown

In many situations the connection between client and server will be realized via ssh tunneling. This could be problematic, because network failures would also shut down the ssh tunnel. SSH itself normally does not monitor the connection state and cannot automatically re-establish the tunnel in case of failure. Thus, a connection failure of the network would effectively prevent the client and server to automatically resume normal operations after the network comes up again.

To overcome these problems we have tested *autossh* (see http://www.harding.motd.ca/autossh/). The tool establishes and monitors ssh connections to remote hosts. In case the connections breaks down it will try to automatically restart it.

The tunnel can be established e.g. with the following command:

```
autossh -M 0 -L -N 50225:remote_host:50225 remote_user@remote_host -N
```

In order to cause network failure the network cable was removed from the client computer. After reconnecting the cable and a short waiting period the ssh tunnel was automatically recreated by *autossh* and the client-server connection was reestablished.

4 Remote control of the field system

Extensive tests were performed to remotely control the field system running in Effelsberg from a workstation located at the MPIfR in Bonn. The connection between the client and the server was realized through ssh tunneling.

Commands were issued to the field system through the XFS client. The field system command response and other log information was in return displayed by the client. The system behaved as expected: the commands were passed on by the XFS server to the field system and executed. The status and log information displayed by the client was in all cases identical to the native field system output. The system ran very reliably without any crashes or other failures.

In order to estimate the network latency we have measured the round-trip delay from the client in Bonn to the server in Effelsberg and back. The round-trip delays for a period of one hour are displayed in Fig. 2. The measured delays are on the order of a few milli-seconds as expected for good connectivity situations. The mean delay is 3.5 msec the highest measured delay is 277.3 msec.

The Wettzell group has performed test observations with much more remote sites like the the Transportable Integrated Geodetic Observatory (TIGO) at Conception/Chile and the German Antarctic Receiving Station (GARS). For a summary of the results see Neidhardt 2009.

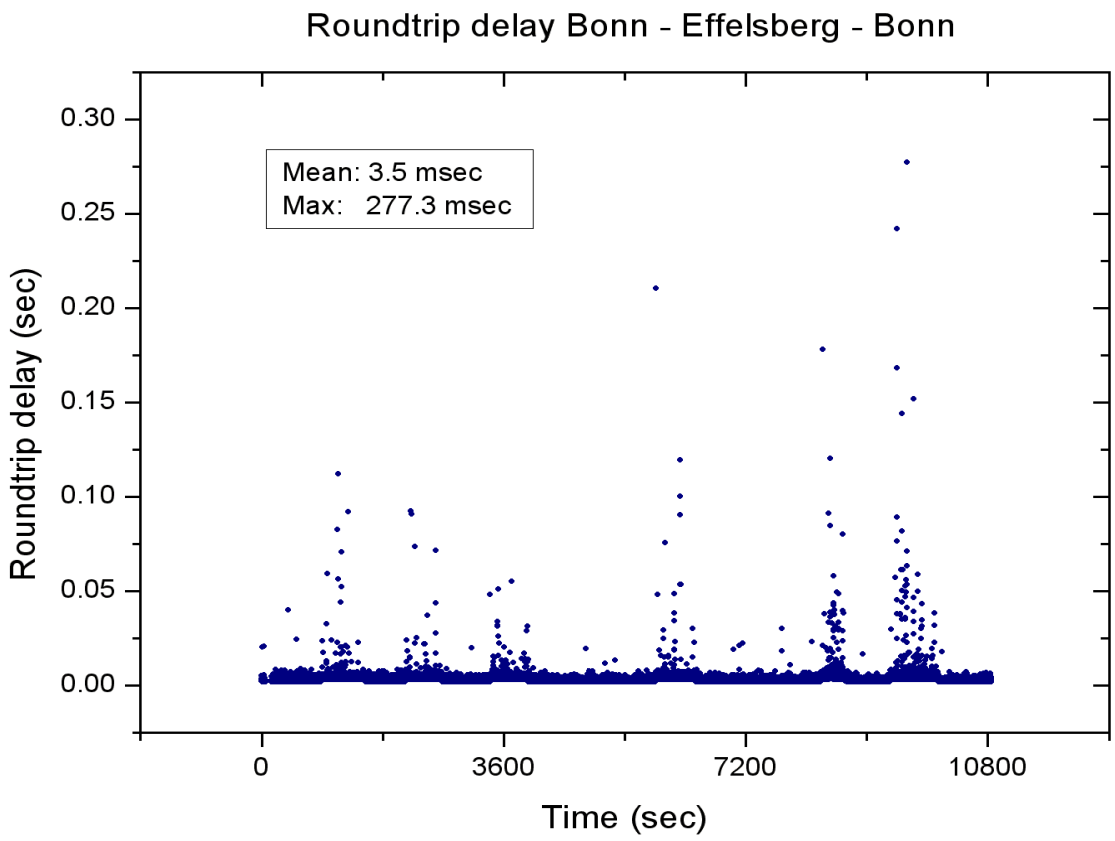

Figure 2: Roundtrip delay measurement between the client in Bonn and the server in Effelsberg for a three hour period. The mean round trip delay was 3.5 msec. The highest observed delay was 277.3 msec.

Bibliography

Neidhardt, Alexander, Manual for the remote procedure call generator "idl2rpc.pl", 2008

Neidhardt 2009, Concepts for remote control of VLBI-telescopes and for the Integration of new VLBI2010 Devices into theField System,
http://www.fs.wettzell.de/veranstaltungen/vlbi/frff2009/Part7/20090319FRFFIDL2RPCNEIDH.pdf

Rottmann H., Neidhardt A., Ettl M., Plötz C. , Himwich E., Data Acquisition Interface Design: Extending the VLBI Field System, 2009