SERIES
SEISMIC ENGINEERING RESEARCH INFRASTRUCTURES FOR EUROPEAN SYNERGIES

Workpackage WP2
Deliverable D2.2 – [Guidelines for implementing a telepresence node]

Deliverable/Task Leader: JRC
Revision: final

June, 2010
ABSTRACT

SERIES/NA2 Networking Activity aims towards the implementation of telepresence within the consortium’s laboratories, particularly those offering Transnational Access to external users. To this end, the following activities have been foreseen:

- Collection of the practical experience regarding telepresence among the beneficiaries of the consortium.
- Issuing guidelines for implementing telepresence (in particular the specifics regarding additional hardware and software).
- Physical implementation of telepresence at the labs (installation of hardware and software), with the support of the beneficiaries having already some experience in telepresence.
- Pilot use of the telepresence capacities during Transnational Access tests and collection of feedback.

Deliverable D2.2 contains guidelines for implementing telepresence tools. It is based on the experiences of JRC, UNIVBRIS & UOXF.DF in the installation, use and customization of the US George E Brown Network for Earthquake Engineering Simulation (NEES) tools.

Keywords: telepresence, architecture, security
ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Community’s Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 227887.

P. Pegon would like to express his thanks to all who contributed to the implementation of telepresence at ELSA: the friends from NEESit at San Diego (L. Van Den Einde, S. Whitmore and P. Hubbard), the IT colleagues (M. Wilikens, R. Sgnaolin, J. Meys and A. El-Abjani) and members of the ELSA team (P. Buchet, L. Papa, K. Mehr, O. Hubert and J. Molina).

Jonathan Evans deserves credit for his role in the implementation of telepresence at the University of Bristol.

All the partners that contributed to the final version of these guidelines are warmly acknowledged, in particular: Anna Bosi (JRC).
DELIVERABLE CONTRIBUTORS

JRC  Pierre Pegon
UNIBRIS  Matt Dietz
UOXF-DF  Ignacio Lamata Martinez
CONTENTS

List of Figures ................................................................................................................................ xi
List of Tables ................................................................................................................................ xiii

1 Introduction ..................................................................................................................................15
  1.1 The distributed laboratory ..................................................................................................15
  1.2 Telepresence .....................................................................................................................18
  1.3 Identification of the laboratories having experience in telepresence.................................19
  1.4 Contents of Deliverable D2.2 ...........................................................................................19

2 A summary presentation of the Available NEES tools ..............................................................21
  2.1 Data Turbine and DATA TURBINE utilities .....................................................................21
  2.2 RDV .....................................................................................................................................25
  2.3 NEESdaq ............................................................................................................................27
  2.4 FlexTPS ..............................................................................................................................28

3 Implementing a Telepresence Node: the experience at JRC ....................................................31
  3.1 Introduction ..........................................................................................................................31
  3.2 Some historical remarks ......................................................................................................31
    3.2.1 Stand alone approach ..................................................................................................31
    3.2.2 European approach ....................................................................................................32
    3.2.3 Collaboration with NEES .........................................................................................33
  3.3 What was needed ................................................................................................................34
  3.4 What was available ..............................................................................................................35
    3.4.1 Controllers and acquisition system ............................................................................36
    3.4.2 Video camera .............................................................................................................38
  3.5 What was missing ................................................................................................................39
  3.6 Implementation of telepresence at ELSA ............................................................................40
    3.6.1 Two new networks .......................................................................................................40
    3.6.1.1 Recommendations ................................................................................................40
Contents

4.6 Access Grid ................................................................................................................... 89
4.7 Practical details ............................................................................................................. 92
   4.7.1 Automatic calculation of Data Turbine cache and archive ........................................ 92
   4.7.2 Code of Practice governing CCTV in the EQUALS laboratory ................................ 93
   4.7.2.1 Initial scheme assessment ............................................................................... 93
   4.7.2.2 Camera types and siting .................................................................................. 93
   4.7.2.3 Camera access .................................................................................................. 95
   4.7.2.4 System operation ............................................................................................. 97
   4.7.2.5 Policing ........................................................................................................... 97
   4.7.2.6 Signage ............................................................................................................ 98
   4.7.2.7 Assessment ...................................................................................................... 98
   4.7.2.8 Example activities ........................................................................................... 99
   4.7.3 How Video Recording sets Cache and Archive size ............................................ 99

5 Implementing a Telepresence Node: the experience at UOXF.DF ................................. 101
   5.1 Introduction ................................................................................................................ 101
   5.2 Installation .................................................................................................................. 101
   5.3 Running FlexTPS ..................................................................................................... 103
   5.4 Configuration .............................................................................................................. 104
      5.4.1 Cameras configuration ................................................................................... 104
      5.4.2 Portal configuration ........................................................................................ 105
      5.4.2.1 Feeds ............................................................................................................. 107
      5.4.2.2 Sections ......................................................................................................... 107
      5.4.2.3 Users ............................................................................................................ 108
      5.4.3 Users configuration ........................................................................................ 110
   5.5 Other customizations .............................................................................................. 112
      5.5.1 Website header picture ................................................................................... 112
      5.5.2 Firefox 3 video error ...................................................................................... 112

6 Storage and bandwidth requirements of NEES telepresence tools .............................. 115
   6.1 Introduction ................................................................................................................ 115
   6.2 Data turbine storage demand .................................................................................... 116
   6.3 RDV bandwidth Requirement .................................................................................. 117
   6.4 RBNB traffic in real-time ........................................................................................ 119
      6.4.1 DaqToRbnb ..................................................................................................... 119
      6.4.2 AxisSource ...................................................................................................... 124
List of Figures

Fig. 2.1 Organisation of test data into Data Turbine frames and blocks ........................................ 22
Fig. 2.2 Data Turbine NEES page. .................................................................................................. 25
Fig. 2.3 RDV user interface. ........................................................................................................... 26
Fig. 2.4 RDV NEES page. ............................................................................................................... 27
Fig. 2.5 NEESdaq NEES page. ........................................................................................................ 28
Fig. 2.6 FlexTPS NEES page. ........................................................................................................ 29
Fig. 3.1 Home-made data collector process and Web displayer for signals. ................................ 32
Fig. 3.2 ELSA software configuration. .............................................................................................. 37
Fig. 3.3 ELSA network configuration. .............................................................................................. 38
Fig. 3.4 ELSA networking scheme. .................................................................................................. 41
Fig. 3.5 ELSA Telepresence data flow. ............................................................................................ 44
Fig. 3.6 ELSA Telepresence video flow. .......................................................................................... 46
Fig. 3.7 Telepresence flow. ............................................................................................................ 47
Fig. 3.8 Time synchronization. ....................................................................................................... 48
Fig. 3.9 Start_RBNB.bat. ................................................................................................................ 53
Fig. 3.10 DaqToRbnb_MS.bat. ........................................................................................................ 55
Fig. 3.11 AxisSource.bat. .............................................................................................................. 57
Fig. 3.12 RBNB.log ....................................................................................................................... 59
Fig. 3.13 DaqToRbnb_MS.log ....................................................................................................... 60
Fig. 3.14 AxisSource_1.log ............................................................................................................ 61
Fig. 3.15 FakeMSDAQ .................................................................................................................... 65
Fig. 3.16 Delays in RDV .................................................................................................................. 68
Fig. 4.1 UK NEES @ Bristol network diagram. ............................................................................ 74
Fig. 4.2 SimAcq (version 3) graphical user interface. .................................................................... 75
Fig. 4.3 The high level design of SimAcqDriver ............................................................................. 76
Fig. 4.4 FlexTPS @ University of Bristol user interface (logged in with the high level user account details). .................................................................................................................. 79
Fig. 4.5 UK NEES @ Bristol web application: Home. .................................................................. 83
Fig. 4.6 UK NEES @ Bristol web application: Acquisition Data Turbine. .................................... 84
Fig. 4.7 UK NEES @ Bristol web application: Record video. ......................................................... 87
Fig. 4.8 UK NEES @ Bristol web application: Generate mpeg ..................................................... 88
Fig. 4.9 UK NEES @ Bristol web application: Copy and resize video ........................................ 89
Fig. 4.10 The three-camera Access Grid Node at UK NEES @ Bristol ......................................... 91
Fig. 4.11 A UK NEES Access Grid Meeting with an invited speaker from the University of Auckland .......................................................................................................................... 92
Fig. 4.12 Test picture from the EERC lab high level camera. ....................................................... 94
Fig. 4.13 Appropriate signage. ........................................................................................................ 98
Fig. 5.1 FlexTPS Web page. .......................................................................................................... 104
List of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 5.2</td>
<td>Example file proxies.xml.</td>
<td>105</td>
</tr>
<tr>
<td>Fig. 5.3</td>
<td>Example file portal.xml (part 1).</td>
<td>109</td>
</tr>
<tr>
<td>Fig. 5.4</td>
<td>Example file portal.xml (part 2).</td>
<td>110</td>
</tr>
<tr>
<td>Fig. 5.5</td>
<td>Default FlexTPS header.</td>
<td>112</td>
</tr>
<tr>
<td>Fig. 5.6</td>
<td>Customized FlexTPS header.</td>
<td>112</td>
</tr>
<tr>
<td>Fig. 6.1</td>
<td>Empirical correlation of the storage and bandwidth demands of NEES telepresence tools.</td>
<td>116</td>
</tr>
<tr>
<td>Fig. 6.2</td>
<td>Traffic for 200ms FakeMSDAQ sampling rate.</td>
<td>120</td>
</tr>
<tr>
<td>Fig. 6.3</td>
<td>Traffic for 20ms FakeMSDAQ sampling rate.</td>
<td>121</td>
</tr>
<tr>
<td>Fig. 6.4</td>
<td>Traffic for 2ms FakeMSDAQ sampling rate.</td>
<td>122</td>
</tr>
<tr>
<td>Fig. 6.5</td>
<td>FakeMSDAQ traffic on the data socket.</td>
<td>124</td>
</tr>
<tr>
<td>Fig. 6.6</td>
<td>Traffic for AxisSource.</td>
<td>126</td>
</tr>
</tbody>
</table>
List of Tables

Table 1 Record of changes of the present report ................................................................. xiv
Table 4.1 UK NEES @ Bristol FlexTPS user accounts.............................................................. 80
Table 4.2 Siting of network cameras in the EQUALS lab....................................................... 93
Table 4.3 Access to the network cameras in the EQUALS lab ............................................. 96
Table 6.1 Traffic on the data socket for various values FakeMSDAQ sampling.................. 123
### Table 1 Record of changes of the present report

<table>
<thead>
<tr>
<th>Version</th>
<th>Author(s)</th>
<th>Change description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Pierre Pegon, Matt Dietz</td>
<td>First version</td>
<td>2010/06/21</td>
</tr>
<tr>
<td>1.1</td>
<td>Ignacio Lamata Martinez</td>
<td>Added Section 5</td>
<td>2010/06/30</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 THE DISTRIBUTED LABORATORY

Around the world, national and international earthquake-engineering communities are exploiting information and communication technologies to revolutionise their research activities. Recent years have seen solitary research infrastructures coalesce into networked collaboratories\(^1\) that offer enhanced modelling, testing and analysis environments of unprecedented scope, scale and sophistication. Most widely known is the United States’ George E Brown Network for Earthquake Engineering Simulation (US NEES) which, following substantial National Science Foundation (NSF) investment, became operational in 2004. Information regarding the conceptions and visions of US NEES can be found elsewhere (National Research Council, 2003). Starting also from 2003, the promotion and the implementation of the Distributed Laboratory is an on-going institutional action at European Laboratory of Structural Assessment (ELSA) of the Joint Research Centre (JRC) of the European Commission located in Ispra, Italy. Three years later, and taking US NEES as a template, the Engineering and Physical Sciences Research Council’s (EPSRC’s) UK NEES was founded from the synthesis of the three preeminent UK earthquake-engineering research infrastructures: the University of Bristol (UNIVBRIS), the University of Cambridge (UCAM) and the University of Oxford (UOXF.DF).

The aim of Distributed Laboratory is to use up-to-date IT tools to dramatically increase the value of testing. Using these tools, it is expected that the tests would become

- more *visible*: more people should be allowed to follow a tests. This does not concern only researchers, but also teachers and students, and even curious people surfing on the internet. Research is not only made to increase the knowledge of a research community,

\(^1\) The term collaboratory is derived from a contraction of the words collaboration and laboratory.
but also to fulfil some public targets. Increasing the visibility is not only for enlarging the scientific participation to a test (discussed further) but to open a window on a research field toward which many common people may have an interest. This implies a correct telepresence infrastructure, delivering enough information but at the same time not interfering with the running of the test.

- more interactive: even if the control of the test remains in last instance the responsibility of the laboratory running it, more interactivity should be introduced. Basic interactivity is provided when a remote user is able to specify the next maximum seismic acceleration in a series of tests of growing intensity. Larger interactivity is required when performing a distributed PSD test with substructuring.

- more documented: the tests are not enough documented. Usually, only the group of people present during the test is really able to know what really happened. Sometimes some minor problems occurred, which are not mentioned afterwards but can have an incidence on the modelling. Sometimes the test goes wrong with respect to its objectives, but still could be useful for modelling.

- generating more results: currently the tests are not generating enough results. This is mainly due to the tedious work of gluing sensors, cabling them with data acquisition system, performing correct calibrations, and verifying the quality of the results. Due to its intrinsic cost, this instrumentation is limited to the needs of the owner of the test. It is thus possible to obtain only a limited number of discrete information, insufficient for correctly identifying a complete finite element model, although these measures are usually quite accurate. Advance methods, based on embedded sensor network, or photogrammetry should be introduced and systematically used in order to have fields of displacements and strains.

- more retrievable: the tests that are archived should be as widely used as possible. They should be stored in a meaningful way, easily accessed and retrievable. This last point is crucial, and it is not expected to be solved at the level of a laboratory. What is needed is a distribution over a community.

All the points briefly outlined above serve to promote participation across the duration of a test: its preparation, running, analysis, exploitation and simulation. This is clearly a distributed effort and the result could be called a distributed laboratory. Within the NA1, JRA1 and JRA2 work
packages of SERIES, most of tools developed during the project will be made available to the laboratories and the users.

- The first step to distribute a test is to distribute the results. This implies the definition of uniform data formats, possibility to easily exchange and retrieve data. NEES decided to build up a central repository (NEESCentral) which would be fed by the NEES laboratories and possibly by external partners. Within SERIES, it has been decided to set up a Distributed Database (DDB) with a centralized access. This is the scope of NA1.1, NA1.3 and NA1.5. The description and the strategy to construct this DDB are given in the SERIES Deliverable D2.1 (Lamata et Al. 2010). Note also that the results uploaded at the local database will become richer using advanced measurement techniques as studied within JRA2.

- The second step is to ensure a correct telepresence. What is telepresence? It is the possibility for a remote user to participate to a test. The user is no longer in the laboratory but is provided with a sufficient amount of key information that permits him to be an active player of the test. Telepresence allows distributing the participation to the test, thus reaching more end-users and increasing visibility. Telepresence is also aiming at saving the time (and cost) of travelling, giving more flexibility in the test planning, limiting the stress and thus increasing the quality of the test. Telepresence is the subject of the present document.

- The third step is to distribute the tests over a network of geographically distant laboratory. The idea is to combine the specific capabilities of different laboratories to the realization of a joint test. The facilities are working on-line and exchange information using internet. Pseudodynamic testing with substructuring is the first approach to investigate for distributing a test because the pseudodynamic approach allows uncoupling the “dynamical” time or the time of the accelerogram and the laboratory time during which the experiment is realized. Effective real-time tests, coupling Shaking Tables, Reaction Walls and Centrifugues are close from being feasible, in particular taking into account the possible improvement of control and actuation system performed in JRA1. Distributed testing is the focus of NA1.4.
1.2 TELEPRESENCE

Central to the functioning of the networks of laboratories is the idea of telepresence. With a prefix from the Latin for ‘at a distance’ and suffix concerning itself with a sensation of existence within an environment, the term telepresence is self-defining. While presence can be usefully summarised as the impression of ‘being here’, telepresence is better thought of as ‘being elsewhere’. Similar to the more familiar experience of virtual reality, those with telepresence are instead commuted to a real (i.e. non-virtual) location. It is important to note that for telepresence to occur the relationship between the user and the remote environment must be one of dynamic interaction (Zahorik and Jenison, 1998). It is insufficient for a user to merely sense a remote environment. Telepresence demands that the actions of the user within that environment to be reflected back by the environment.

As such, telepresence is seen as a matter of degree (Sheridan, 1992). Perhaps the most rudimentary telepresence experience is provided by the telephone, a technology dating from 1876. With both audio and video transfer, video-conferencing is capable of offering an updated and more immersive telepresence. In these instances, by responding to perceived external audio/visual information, the user is an agent within the remote environment. More sophisticated telepresence systems give the user the power of manipulation. In teleoperation, user movements are sensed and reproduced by a remote machine (for example, in telesurgery). Sate-of-the-art haptic teleoperation systems provide tactile force-feedback. Here, donned attire offers levels of resistance appropriate to user action engendering the perception of mass and stiffness.

In the context of NEES, the term telepresence refers to the technology (hardware, software, and interface) that allows a person to engage with and contribute to concurrent research regardless of their geographical location (Sritharan and Shield, 2008). It should be noted that by this definition the technology designed to engage computers or processes with remote research (i.e. for distributed hybrid testing) is not considered to be telepresence since personnel are excommunicated during such experiments.

US NEES distributes four tools designed to provide telepresence over intermittent, high-latency, limited-bandwidth networks. In essence, these tools can be considered to be middleware that
provide the end user with a common interface for accessing the different backend systems at research infrastructures. As such, the software is released under an open source license which allows infrastructure sites to customise according to their specific needs. The tools can be freely downloaded from www.nees.org/it/. To capitalise on the substantial effort expended in their development, the implementation of the US NEES telepresence tools formed a significant part of the Distributed Laboratory at JRC and for UK NEES. This document details the experience of JRC, UNIVBRIS & UOXF.DF in this endeavour.

1.3 IDENTIFICATION OF THE LABORATORIES HAVING EXPERIENCE IN TELEPRESENCE

At the beginning of the SERIES project (March 2009), an inquiry was been circulated among the SERIES partners in order to identify those consortium members with experience of telepresence, It was apparent that only JRC and UK NEES laboratories (UNIVBRIS & UOXF.DF) have the experience of implementing, customizing and using telepresence tools.

1.4 CONTENTS OF DELIVERABLE D2.2

The US NEES software is designed as a collection of parts rather than a single installation. Each part has associated utilities. In addition, US NEES software development has occurred in a piecemeal fashion. This is reflected in both the downloads, wherein both principle and obsolete software components cohabit, and the scattershot documentation. As a result, the first challenge of implementation is to separate the wheat from the chaff and Section 2 of this document should help with this endeavour.

Section 3 will present the experience of JRC in terms of telepresence, Section 4 the experience of UNIVBRIS and Section 5 a complementary experience at UOXF.DF. The reasons for keeping these experiences separated instead of providing a synthesis are numerous and worth to be enumerated here in this introductory section.
• JRC mainly performs Pseudo Dynamic (PsD) tests with a laboratory time scale which is a dilation of the accelerogram time scale. UNIVBRIS with its shaking table (and UK NEES in general) is in the real time scale. A test at JRC can last for hours whereas it is hardly longer than a minute at UNIVBRIS or UOXF.DF. This clearly has an impact on telepresence and the tools in use.

• JRC started its implementation in 2003 and UK NEES in 2007. As such JRC inherited from a “vision” which is more related to the first IT group of NEES (NEESGrid), whereas UK NEES directly inherited from the second IT group of NEES (NEESit). As such, the developments performed at JRC are quite basic (telepresence is working), whereas more applications have been developed at UK-NEES (telepresence is working and user-friendly) and conceived for working as an integrated environment.

• The main difficulties encountered by the two laboratories were to adapt tools developed for a group of laboratories with dedicated connections and using rather uniform data acquisition systems (LabVIEW) and cameras (Axis), to their local hardware, software, and firewall policies. Since local hardware, software, and firewall policies are different between JRC, UK NEES and US NEES, the telepresence implementation performed by JRC and UK NEES needed different adaptations, presented herein as possible alternatives.

• The UK NEES installation works mainly on Linux whereas JRC has experience on both Linux and MS-Windows.

• Finally the experience of UNIVBRIS is presented from a high level point of view whereas the complementary contribution of UOXF.DF presents practical aspects regarding installing and running FlexTPS.

Section 6 gives some indication about storage and bandwidth requirements to run the NEES telepresence tools.

The present report uses Courier New fonts to name software components and present scripts.
2 A summary presentation of the Available NEES tools

The four telepresence tools distributed by US NEES are: Data Turbine, NEESdaq, Real Time Data Viewer (RDV) and Flexible Telepresence System (FlexTPS). They are all available from the link https://www.nees.org/it/.

Practical indications on how to run these components are given later as part of JRC and UNIVBRIS experiences.

2.1 DATA TURBINE AND DATA TURBINE UTILITIES

The principle software that handles simultaneous recording and streaming of data is the Data Turbine. This is supported by the Ring Buffered Network Bus (RBNB), a daemon process implemented in Java and developed by Creare Inc. (Note that Data Turbine and RBNB are used interchangeably in US NEES literature.) The strength of the Data Turbine lies in its ability to integrate diverse data types; it is equally adept at handling numerical and video data. The innovation of the Data Turbine is to combine ring buffers with network bus elements. Network bus elements perform the background multiplexing and routing of data streams. Ring buffers are configurable combinations of memory and disk that are used to hold data.

In Data Turbine parlance the disk-based component of a ring buffer is called the ‘archive’; the memory based component, the ‘cache’. The archive should be large enough to hold all of the data generated by a data source. If too small, once the archive is full, the oldest data are irretrievably discarded to make room for the new. The cache contains a copy of the most recent archived data for quick access. Like in the archive, once the cache is full, new data continually replaces old. The size of the archive is restricted by the size of the disk; the size of the cache is...
A summary presentation of the Available NEES tools

restricted by the available RAM and, as such, is typically a fraction of the size of the archive. (Note that the Data Turbine will crash if the total memory consumed by all caches exceeds the amount of allocated memory). Great care must be taken when configuring the size of the archive and cache.

A ‘frame’ is the basic unit of storage in the Data Turbine. As indicated in Fig. 2.1, each frame can contain one or more channels of data and each ‘block’ of channel data can contain one or more data points. Frame size is derived from the number of blocks per frame, and either the number of data points per block or the rate at which frames are flushed from a source to the Data Turbine. Hence, the archive size can be estimated from the frame size and the test duration. Note that within the ring buffer, a set of ten frames becomes a frameset and it is at frameset granularity that old data are discarded from the cache; in the archive, a set of ten framesets becomes a fileset, and it is these which are identifiable as folders on the disk (named RB1, RB2 etc. under the source name in the RBNB archive).

![Diagram of Data Turbine frames and blocks](image)

**Fig. 2.1 Organisation of test data into Data Turbine frames and blocks**
‘Source’ applications are used to both configure the Data Turbine and to assemble frames which are ‘flushed’ to the Data turbine when full. Two source applications will be introduced here. 

**DaqToRbnb** is responsible for managing the connection between Data Acquisition (DAQ) software and the Data Turbine. It can be run as a daemon process. Metadata passed by **DaqToRbnb** are used to configure the Data Turbine by providing an archive name, the ‘source name’, the channel names, and the channel units. Note that source names must be unique to prevent new acquisition data being added to old ring buffers. **DaqToRbnb** then builds incoming DAQ data into frames which, when full, are flushed to the Data Turbine. Command line parameters are used to establish the IP address of the DAQ source, the frame size (by specifying either the number of data points per frame or the flush rate), the archive size, and the cache size.

**AxisSource** is the mediator of video data between an Axis IP video camera and the Data Turbine. An instance of **AxisSource** is required per network camera. Command line parameters are used to configure the important parameters such as the IP address of the network camera in question, the source name, the resolution and the frame rate. Note that the source works also for Axis video server. By default, video is acquired at a resolution of 704×480 pixels at a rate of 30 frames per second.

Apart from **AxisSource** and **DaqToRbnb**, other Data Turbines Utilities are available at the NEES site:

- **Archive**: Automated Archive is a system comprised of components designed to archive the sample data derived from an experiment and to upload these data to the NEEScentral data repository.
- **DLinkSource**: DLinkSource acquires data from a D-Link DCS-900 camera and sends them to the Data Turbine server.
- **FileToRbnb**: FileToRbnb reads from a DAQ-formatted ASCII file and sends the contents to the Data Turbine server.
- **FlexTpsSource**: FlexTpsSource acquires data from a FlexTPS video stream and sends them to the Data Turbine server.
- **JpgLoaderSource**: JpgLoaderSource sends timestamped JPEG images within a specified time range from disk to the Data Turbine server.
A summary presentation of the Available NEES tools

- **JpgSaverSink**: JpgSaverSink requests images from the Data Turbine server and saves them to disk (independent of the Data Turbine ring buffer) so they can be backed up or otherwise manipulated.

- **OpenSeesToRbnb**: this utility reads an OpenSees xml file in the format per XSD: [http://opensees.berkeley.edu/xml-schema/xmlns/OpenSees.xsd](http://opensees.berkeley.edu/xml-schema/xmlns/OpenSees.xsd) and imports the data to Data Turbine server.

- **Orb2Rbnb**: Orb2Rbnb requests data from an Antelope ORB, translates them into a form compatible with RDV, and transmits them to a Data Turbine server.

- **PanaSource**: PanaSource acquires data from a Panasonic camera or video server and puts them into the ring buffer. By default, video is acquired at a resolution of 704×480 pixels at a rate of 30 frames per second.

- **RbnbToFile**: RbnbToFile extracts numeric data from channels of interest and creates an ASCII-formatted output file.

None of these Data Turbines Utilities will be further commented in this document.

The Data Turbine, the Data Turbine Utilities and the associated documentation can be found using the link [https://www.nees.org/it/software/dataturbine/](https://www.nees.org/it/software/dataturbine/) (see Fig. 2.2).
Fig. 2.2 Data Turbine NEES page.

2.2 RDV

The Real Time Data Viewer is a user-desktop application implemented in Java. Classed as a Data Turbine ‘sink’ application (i.e. one which retrieves Data Turbine data), RDV provides a configurable window in which data originating from a Data Turbine can be visualised. Utilizing Java Web Start RDV is downloaded, installed, and launched via a web browser. System security is maintained because Java Web Start programs are executed within a restricted sandbox.

RDV communicates with a Data Turbine via TCP traffic over a specified port (3333 by default). The user must identify the IP address or alias of the Data Turbine server within RDV. Once connected, the various archives or ‘sources’ loaded within the Data Turbine are visible as folders within the Channels window, as indicated by the RDV user interface presented as Fig. 2.3. Expanding a source reveals the channels available within that source; Data Turbine frames are
invisible. If live data are of interest, RDV subscribes to a Data Turbine source using the ‘view live data’ button. Frames flushed to the Data Turbine by source applications are then simultaneously streamed to RDV.

![RDV user interface](image)

**Fig. 2.3 RDV user interface.**

RDV is capable of displaying data in a variety of formats: tabular, time-series or x-y plots. The user can configure both the time window (from 1ms to 1 week) and the playback rate (from 0.001 to 1000) using drop down menus. Additionally, some basic signal processing (spectral analysis) can be conducted. Video data can be simultaneously displayed alongside DAQ signals but can impede RDV performance and appear staccato (particularly with high resolutions and frame rates) if the available memory and bandwidth are insufficient.

RDV and the associated documentation can be found using the link [https://www.nees.org/it/software/RDV/](https://www.nees.org/it/software/RDV/) (see Fig. 2.4).
For an infrequent user, it is recommended to Launch RDV rather than Download it. In this way, the user benefits from the latest release of the tool.

2.3 NEESdaq

Implemented on the National Instruments LabVIEW platform, NEESdaq streams live data-acquisition signals from a research infrastructure’s legacy data-acquisition equipment to the Data turbine via DaqToRbnb. NEESDaq is designed to interface with the data acquisition equipment, manipulate the output, apply a time stamp and deliver the data to the DaqToRbnb in the required format.

NEESdaq and the associated documentation can be found using the link https://www.nees.org/it/software/neesdaq/ (see Fig. 2.5).
2.4 FlexTPS

FlexTPS is a software system which enables the end user to view and robotically control live video over the internet through a web browser. (Mozilla Firefox is the preferred browser. Due to a lack of support of the MJPEG format, Internet Explorer must be able to run a Java applet sent by FlexTPS.) Multiple video streams can be handled simultaneously.

There are two components in FlexTPS: the portal and the proxies. The end-user connects to the FlexTPS portal and with appropriate authentication can view one or more video streams, administer the streams (i.e. set frame rate, robotically control cameras) and record them to disk. The proxies package handles the streaming of the JPEGs from the cameras to the portal and
A summary presentation of the Available NEES tools

robotic (pan, tilt, zoom) commands from the portal to the cameras. Both the portal and the proxies are configured with xml files. The portal is developed in Perl, JavaScript, and Java while the proxies are pure Perl.

**FlexTPS** and the associated documentation can be found using the link [https://www.nees.org/it/software/FlexTPS/](https://www.nees.org/it/software/FlexTPS/) (see Fig. 2.6).

![FlexTPS NEES page](image)

Fig. 2.6 FlexTPS NEES page.
3 Implementing a Telepresence Node: the experience at JRC

3.1 INTRODUCTION

This section reports the experience of JRC in terms of telepresence. The section starts with some historical remarks (section 3.2) that can be easily skipped by the readers only interested on technical aspects. It then describes what was needed (section 3.3), available (section 3.4) and missing (section 3.5), and finally what was done (section 3.6). Section 3.7 gives more details on practical aspects of interest.

3.2 SOME HISTORICAL REMARKS

3.2.1 Stand alone approach

As mentioned in the introduction, an action aiming at transforming ELSA from a stand-alone laboratory to a node of a distributed laboratory started in 2003. The very first implementation of telepresence concerned on-line signal visualisation was based on a simple house-made software (see Fig. 3.1).
Very soon the limitations of such a home-made approach showed its weakness, due to the complexity of the problem as a whole and the lack of manpower.

3.2.2 European approach

Collaboration at European level has been also foreseen, but, unfortunately, all the proposals aiming at constructing a community of earthquake laboratories at the European level (Research: OCELEE in 2004, Networking: DESTINEED in 2004, Access & Research & Networking: INTRALEDGE in 2003 and INTRAQUAKE in 2005) failed at obtaining financing. It is worth to remind the reasons given by the reviewers to reject the proposals:

- Focus only on well established Laboratories: although there are 16 shaking tables, 6 reaction walls and 8 centrifuges in Europe, the proposals concerned only the largest and most established laboratories. In fact it was felt that the setting up of a distributed approach needed a critical mass, an historical perspective, a tradition of collaboration and a limited number of partners in order to start an action.
- Lack of industrial partner: most the public laboratories in Earthquake Engineering are only scarcely working for the industry, and, when it is the case, in very specific matters.
It is thus not surprising that it was hard to involve industry in setting up a general approach like the implementation of a distributed laboratory.

- Low level of innovation: the distributed laboratory combines existing (and heterogeneous) capabilities geographically distributed and a huge use of internet. Although the approach is innovative in its field of application, the use of internet (Gigabit network, grid authentication, secure exchange of data, etc.) is only at the edge of the technology and does not really require new development at the network level.

As a matter of fact, SERIES is the first initiative aiming at constructing such a community and almost all the actions regarding the Distributed Laboratory were put in a Networking Activity (NA1) rather than in a Joint Research Activity.

3.2.3 Collaboration with NEES

NEES (the so called George E. Brown, Jr. Network for Earthquake Engineering Simulation) is “a shared national network of 15 experimental facilities including also collaborative tools, a centralized data repository, and earthquake simulation software, all linked by the ultra-high-speed Internet 2 connections of NEESgrid. Together, these resources provide the means for collaboration and discovery in the form of more advanced research based on experimentation and computational simulations of the ways buildings, bridges, utility systems, coastal regions, and geomaterials perform during seismic events”. NEES is a US/NSF initiative started in 2000 and covering a 15 years period.

In September 2004 a Harmonization Workshop in the framework of the I-SAMCO project was organized at the JRC-Ispra. On that occasion, the importance of having a detailed presentation of NEES and NEESgrid tools within Europe was highlighted.

ELSA organized in May 2005 a NEES and NEESgrid Seminar aiming to present to the EU community of mechanical laboratories what was available at the US level and what was the further direction of developments of the software. Of particular interest for ELSA was the presentation in the seminar of a data turbine, synchronizing and buffering very heterogeneous data flows (measurement signals, video signals, etc.) and the associated data
viewer processing the output of the data turbine and providing a Web access to the data. The main NEES informatics tools were developed, maintained and distributed by special NEES entities: NEESgrid, NEESit and now NEEScomm.

At that time it was decided to base the development of telepresence at ELSA on NEES tools.

### 3.3 WHAT WAS NEEDED

Telepresence provides a remote access to on-line multimedia data: video, photos, audio, documents, etc. Such data should be:

- **Synchronized** with one another. This is an important point allowing also to explain why telepresence is different from standard acquisition. During a PSD test, video camera are used to register a frame at some given instant (generally synchronized with a large step of the acquisition system) and corresponding to an amount of 100 to 50 frames per second of the accelerogram used as input of the test. It is then possible to play back the test at real accelerogram time. The same video camera can be used for telepresence, but should provide a framing allowing to understand what is occurring in the real (clock) time. The same device (video camera) is thus used for two purposes that are quite different, and requires a quite different treatment. Concerning the data acquisition, the data should be produced a priori synchronized, each frame corresponding to a given time of the accelerogram. In this case the recording device is triggered by the acquisition system and thus the data are a priory synchronized. For telepresence the issue is to provide as many information as possible. The focus is more on performance in terms of amount of data, than on exact synchronization involving triggering. This allows using the best suited stand alone systems (for instance a Video server performing the compression and the steaming of the video image, without any connection with the process collecting the data from the acquisitions) but implies an a posteriori synchronization at the moment of the display. The sequence of generation of the data should be respected, implying use of time stamps among distributed hardware using synchronized clocks.

- Accessed via a *unique* interface. Although this could be considered as a comfort point, it remains quite important for the user to use the same tool (or to enter a unique Web site)
to access all the telepresence data within a workspace. This allows watching a test without necessarily monopolising all the graphical resources of the remote user’s workstation for telepresence. This is particularly convenient for a PSD test that can last for hours.

- Including playback capabilities. It is quite important to be able to playback a test (or a part of it). In fact, it is a key instrument helping to decide what to do next during an experimental campaign. This allows also quick and on-the-spot inspection of the results when something goes wrong. Moreover, a remote user entering a test after it started is also able to review quickly what happened before stepping to the real time.

- Not interfering with the standard acquisition/recording system: it is intuitive that, to reach many remote users contemporaneously, the telepresence system will rely on internet and on various hardware and software using network capabilities. As it is well known, internet gives quite non deterministic performance, in particular when limited bandwidth is available. Telepresence tools can thus pause, miss data points and even stop depending on reasons having nothing to do with the test itself. Data acquisition, on the other end, is in general producing data that are not supposed to travel (at least during the test), and should be highly reliable. It is thus advisable to use different systems for handling the data.

- Passing through a firewall: the data are generated inside the JRC, they should be visible outside the JRC. Since the ELSA network, subnet of the JRC corporate network, is protected from intrusions from outside by a firewall, some provisions should be made when implementing telepresence in order to cope with this problem. Obviously this kind of security issue is not specific to the JRC.

### 3.4 WHAT WAS AVAILABLE

Apart the NEES tools already presented in Section 2, it is worth presenting very shortly the ELSA installation just before the implementation of the telepresence.
3.4.1 Controllers and acquisition system

From many years, the ELSA laboratory is running the continuous PSD method using integrated controllers and acquisition systems. These systems offer real-time features for processing the PSD algorithm (in an interrupt triggered every 2 ms) or storing data acquisitions of signals. It offers also remote features, with lower priority, allowing access from the network to almost all of the processed data, by means of a (MS) DCOM object (AcqCtrlService). In fact, during the in-house design and implementation of the system two ports (10000 and 10080) have been reserved for network applications. The first one (10000) is used for remote control applications (for instance substructuring) and the second one (10080) is reserved for Web applications (see Fig. 3.2).

This configuration is well suited for the telepresence application since it offers the possibility to collect the data from the controllers and the acquisition systems without impacting the normal running of the experiment. It also offers the flexibility to handle the firewall issues without changing the software used at the laboratory.

A possible network configuration could be proposed in Fig. 3.3 where a data-collector process devoted to telepresence can be run on an additional workstation.

From the software and the hardware point of view, telepresence can thus be seen as an add-on to pre-existing systems requiring no modification of those systems, and with no impact at the real-time level.
Implementing a Telepresence Node: the experience at JRC

Fig. 3.2 ELSA software configuration.
Some video cameras (up to 4) are currently used at ELSA to give a general view of the specimen. As already briefly highlighted, these cameras are connected to video recorders, triggered by the controller, allowing to capture images at some pre-fixed moments of the accelerogram used as input of the experiment.
It appears that the same camera could be connected at the same time to another hardware (Video server) used for telepresence application. At this level, there is a wide possibility of choice due to the expansion of the video-surveillance market.

### 3.5 WHAT WAS MISSING

What was thus missing for implementing telepresence was related to the differences in working standards between the NEES laboratories and ELSA/JRC. In particular:

- **Programming environment:** the default system for NEES is Java under Linux whereas in ELSA it is C# or C++ under Microsoft/.NET. Since Java is implemented under a Microsoft system as an almost separated virtual machine, it is not trivial to make a direct bridging for instance between the (MS) DAQ and the (Java) RBNB. It is worth to underline the value of a process such as DaqToRbnb which enables to implement and indirect approach using sockets.

- **Data acquisition:** the default system for NEES is LabVIEW and a complete interfacing of LabVIEW with RBNB is provided by NEESit. Given that the ELSA DAQ and control systems are home-made, an interface is required.

- **Firewall:** the JRC is a Research Centre and at the same time a Directorate of the European Commission. Most of its informatics resources are deployed within a firewall implementing strict rules. In anyway it is quite obvious that the system holding RBNB should be outside the main firewall (DMZ) since it acts somewhat like a data server for all the remote clients running RDV. On the other hand the systems producing the data are inside the firewall.

A general picture of the system can be then summarized. ELSA is inside the firewall. Data from the controllers and the data acquisition system are collected by a devoted machine (*Data Collector Server*). These data are sent to another devoted machine (*Telepresence Server*) running RBNB located outside the main firewall. A Video server, working as the Data Collector server regarding video is also sending data to the Telepresence server. Telepresence clients, around the world, connect the JRC Telepresence server and can participate to an experiment.
The main problem associated with this architecture is related to the rules of the firewall.

- Data from the DMZ entering inside the main firewall are strictly filtered. However, the connection between the Data Collector (gathering data) and the Telepresence server (running DaqToRbnb) is to be performed by means of two sockets: a data socket and a command socket. The data socket is a one-way pipe just sending data, whereas the command socket is a two-way pipe. For instance, before starting the streaming of data to RBNB, DaqToRbnb asks the other process questions about the channels to stream.
- A process inside the firewall cannot be a socket server for a process outside. Unfortunately, DaqToRbnb has been created as a socket client and not as a server.

At this point two choices were available: modifying the nature of the existing network, or changing DaqToRbnb. Although it may be obvious at a first glance that the second solution is easier, the first one has been chosen. The NEESit’s original product (and the future updates) can be used as it is and a new network strategy offers some interesting opportunities. This will be presented in the next chapter.

### 3.6 IMPLEMENTATION OF TELEPRESENCE AT ELSA

#### 3.6.1 Two new networks

**3.6.1.1 Recommendations**

- The Telepresence server should be on the DMZ (Public Zone network of JRC), in order to make it accessible from the internet.
- The Telepresence applications should run without any root/administrator privilege.
- There is the need to set up a private network, the ELSA Laboratory network, to be used for all laboratory equipments. A Virtual LAN (VLAN) will be used to host this network.
- There is the need to host in a specific subnet (the Intermediate Telepresence network) all systems that need to communicate with the Telepresence server, namely the Video server and the Data Collector server.
Implementing a Telepresence Node: the experience at JRC

- Considering that the Data Collector server needs to gather data from sensors, and that all sensors will be located in the Laboratory network, the Data Collector server will have two network interfaces. It is agreed that it shall not be used as a gateway, to prevent direct connections from one network to the other.
- All computers located in the *ELSA Intranet* network\(^2\), should be able to connect to the Data Collector server, in order to exchange files; communication in the opposite way will not be enabled.
- The Video server will be put on the Intermediate Telepresence network as well, and will be made accessible from the ELSA Intranet network as well as from the Telepresence server by means of the *AxisSource* data turbine Utility.

The resulting network scheme is illustrated on Fig. 3.4.

---

\(^2\) This is the original network where *all* the ELSA computers were connected.
3.6.1.2 The Laboratory network

As usual, any change involves advantages and drawbacks. Since the introduction of a new network involves considerable work and implies changes to working habits it is important to be aware of the negative implications of this change particularly since the existing systems perform satisfactorily.

In order to have the Laboratory networks protected as much as possible, the Data Collector server is the only bridge between the Laboratory network and other networks. The Laboratory network is thus isolated and only visible from the Data Collector server.

This implies the following practical drawbacks:

- The workstations on the Laboratory network do not have access to the public internet network. It is thus not possible to consult information pages, and download programs, in particular system updates.
- The workstations on the Laboratory network cannot be seen directly from the ELSA Intranet network. It is thus no longer possible to download directly the acquisition results of a test, for post-treatment, storage and post-processing to the database.
- The workstations on the Laboratory network cannot see the ones of the ELSA Intranet network. It is thus no longer possible to directly download program updates on controllers.

However, the fact that the workstations in the Laboratory network do not have any unwanted interference with other networks presents also many advantages:

- The controllers/acquisition systems cannot be any longer (pre-emptively) accessed by other workstations but the ones of the laboratory. The number of remote machines being reduced on the Laboratory network, the new network thus allows to reduce the occurrence of such conflict of interest.
- Since the controllers/acquisition systems are special workstations at both hardware and software levels, the fact that they are all on the same network, together with a limited number of auxiliary MS-Windows stations can be seen as an advantage at the
maintenance level, since the updates have to be handled in a different way. This fits with the JRC strategy to divide the IT structure of JRC into specific specialized zones.

- The traffic of data on the Laboratory network is now kept under control. Being isolated, intermittently run, bandwidth hungry applications no longer impact the traffic on the Laboratory network. Uniform network performance is of great benefit when conducting tests that rely on effective communication between workstations such as substructuring or special implementation of the conventional PSD technique.

- At the same time, traffic on the Laboratory network does no longer impact the ELSA Intranet network. This creates an opportunity for using wireless sensors. Previously, and for practical reasons, only one Wi-Fi access point per building Intranet network was allowed. This restriction is not imposed on isolated networks, allowing the introduction of multiple access points, each of them connected in turn to a grape of wireless sensors.

- Finally, the latter two abovementioned drawbacks could be overcome by considering that the Data Collector server can also be viewed as a backup server for the raw laboratory data and the development libraries. In fact, the setup of the networks is made in such a way that the Intermediate Telepresence network can be viewed from the ELSA Intranet network, and its disk resources can be mounted as a network drive. Even if the contrary is impossible (the Intermediate Telepresence network is unable to have access to the ELSA Intranet network), this is not a limitation in practice: the acquisition results of the tests can be downloaded from the controllers/acquisition systems on a shared location of the Data Collector server and thus be ready for any treatment from the ELSA Intranet network. Conversely, the .exe and .dll files resulting from the programming developments of the controllers can be directly stored on the Data Collector server and then downloaded on the controllers.

### 3.6.1.3 The telepresence network

The Telepresence network is an intermediate network, which allows decreasing the level of the security rules in the dialog with the DMZ network. In particular, a machine on this network can act as a socket server for a machine located outside the main firewall such as the Telepresence server.
The Video server is also directly connected to this network and is allowed to open video streams towards the Telepresence server. Note that the Laboratory network is thus not affected by the traffic of data resulting from the use of on-line video.

### 3.6.2 Data Flow

This section summarises how the data available at the level of the controller arrive to the data turbine (see Fig. 3.5). In order to be able to explain better what has been developed, and how, the presentation starts from the end point (the Telepresence server) and go forward toward the starting point (controllers or DAQs).

![Fig. 3.5 ELSA Telepresence data flow.](image)

- The last process involved in the data flow and running on the Telepresence server is the Java data turbine RBNB as described in Section 2.1. Additional practical information together with a short MS/DOS script is give in Section 3.7.1.1. An equivalent script for Linux is available for SERIES partners on request.
The Java process feeding RBNB and also running on the Telepresence server is DaqToRbnb. As already mentioned in Section 2.1, it is an intermediate between the acquisition system and the data turbine. Its role is to convert a flow of ASCII data with time stamps in a suitable format understandable by RBNB. This program is very helpful: it allows implementing a telepresence node benefiting from the data turbine without entering in the burden of knowing how to feed it. What is needed is a client socket connection on a control port (on which a data transfer transaction is performed) and a data port (on which the data are sent). Additional practical information together with short MS/DOS scripts are given in Section 3.7.1.2. Equivalent scripts for Linux are available for SERIES partners on request.

The MS-Windows process feeding DaqToRbnb and running on the Data Collector server is daq2rbnb. The data exchanged between daq2rbnb and DaqToRbnb transit between the Intermediate Telepresence network and the outside firewall network. daq2rbnb is also an intermediate program. Its mission is to read selected data available on various controllers/DAQ units distributed on the Laboratory network and available through the ActrlService DCOM object (see Fig. 3.2), to format them conveniently and send them to DaqToRbnb via the Intermediate Telepresence network. This is the only new tool that have been developed at ELSA. Details about this development, together with the presentation of a generic tool named FakeMSDAQ, are available in section 3.7.2.

3.6.3 Video Flow

This section explains how the data available at the level of the video camera arrive to the data turbine (see Fig. 3.6). No important development has been needed here but, for the sake of simplicity, the presentation starts from the end point (the Telepresence server) and go forward towards the starting point (the video cameras).
The last process involved in the video flow and running on the Telepresence server is again the Java data turbine RBNB.

The Java program feeding RBNB is called AxisSource (see Section 2.1). AxisSource acquires data from an Axis camera or Video server and puts it into the ring buffer. When launching the application, it is possible to precise the resolution and the frame rate. Additional practical information together with a short MS/DOS scripts are given in Section 3.7.1.3. An equivalent script for Linux is available for SERIES partners on request. The existence of AxisSource was important since ELSA did not intend to perform development for streaming video. Remember that even if only one video server is used (which can connect in the case of ELSA 4 cameras), an AxisSource process for each camera should be run on the Telepresence server.

Having already various video cameras, it was decided to connect them on an AXIS 241Q Video server. This server allows to encode the video signal of one to four analog video cameras and to use it on IP network. The Video server implements a web site that can be
Implementing a Telepresence Node: the experience at JRC

accessed as a standard user. As a matter of fact, \texttt{AxisSource} logs in on the Video server to extract the data. The Video server is put directly on the Telepresence network. No potentially huge traffic of video data is affecting the performance of the Laboratory network.

- More details are available for SERIES partners on request.

### 3.6.4 Telepresence client

Finally the data are available to the remote user, connecting the data turbine by means of a remote Java application (running on the local machine of the telepresence user, see Fig. 3.7).

![Telepresence flow](image)

**Fig. 3.7 Telepresence flow.**

- The first process involved, running on the Telepresence server, is the Java data turbine \texttt{RBNB}.
- The second process, connecting the data turbine to extract data is the Java Real-time Data Viewer \texttt{RDV} (see Section 2.2). Some practical details about \texttt{RDV} are given in Section 3.7.4.

### 3.6.5 Time synchronization

Since the \texttt{RBNB} data turbine ingests time stamped data and since \texttt{RDV} displays them in real time, it is very important that all the data sources pushing data on the various Java applications feeding
RBNB are synchronized. Various issues behind time synchronization are explained in (Hubbard 2006). Apart from being misleading, poor synchronization has spurious effect on the behaviour of RDV as explained in Section 3.7.4.

For synchronizing the Video server (video flow), the Data Collector server (acquisition data flow) and the Telepresence server (FakeMSDAQ data flow), the Network Time Protocol is used with a connection to a local NTP server machine located at time.jrc.it (see Fig. 3.8). Details of the NTP implementation are given in Section 3.7.3.

![Fig. 3.8 Time synchronization.](image)

### 3.6.6 Limitations

Some comments are made regarding the hardware in use for running the NEES telepresence tools at ELSA.

#### 3.6.6.1 Telepresence Server

The machine hosting RBNB should have, following the NEESit recommendation, at least the following capacity:

- Processor: Pentium IV (x86) or 1GHz PowerPC or Intel (Mac).
Implementing a Telepresence Node: the experience at JRC

- Physical RAM: 512MB, or 1GB if the system needs to be shared with other applications or services.
- Hard disk: 10GB free for basic use

The effective ELSA Telepresence Server is an “old” server (from 2004) running under Linux Red Hat with:

- Processor: 2 x Pentium IV 2800 MHz.
- Physical RAM: 2GB.
- Hard disk: 2x 160 GB

This is considered at ELSA as a minimum requirement, in particular in terms of RAM. For instance, the traffic evaluation presented in Section 6.4 was made with a computer having only 1GB of RAM, which was clearly insufficient as soon as many applications connected RBNB.

3.6.6.2 Data Collector

The Data Collector is a server (from 2007) running under MS/XP with:

- Processor: 2 x Dual Core Xeon 5110 1600 MHz.
- Physical RAM: 2GB.
- Hard disk: 2x 250 GB

This machine is definitely sufficient for its purpose.

3.6.6.3 Network

The ELSA intranet, the Laboratory Network and the Intermediate Telepresence network are all 100 Mbit/s broadband networks.
Although the machine running RBNB and the data turbine utilities is quite powerful, there is a bottleneck when various users try to load data at the same time. After performing specific tests, the following remarks were made:

- There is no limitation on the number of data channels.
- There are limitations on the number and the quality of the video channels. Two points of view can be afforded. Low quality (352x240) video streams can be issued with 30 frames per second whereas high quality (704x480) streams are limited to 5 frames per second.
- It is preferable that no more than 3 (external) users get access to the turbine when video channels are in use.
- There is also limitation in the playback rate (no more than 50 times faster).

3.6.6.4 RDV

In order to take the best benefit of RDV, it is recommended to use it with an up-to-date machine with at least 1GB of RAM.

Note that if RDV is connected to an overloaded RBNB Server, it may present an erratic behaviour that is not related to the machine on which it is run.

3.7 PRACTICAL ASPECTS

3.7.1 Data turbine and Data turbine utilities (running on the Telepresence Server)

The Telepresence server was originally a Red Hat Linux workstation providing the necessary performance to host a NEESpop (Point Of Presence). In fact at the time the work on Telepresence started, setting up a NEESpop (including all the software from NEESGrid) was the only (easy) way to install the telepresence software. The situation is now completely different. The available software are now divided in packages and can be installed individually quite easily and because the implementation of the Data Turbines and the associated utilities can be done on a Windows machine with the appropriate Java Runtime Environment (JRE), that should be
installed before RBNB. The requested version can be found in the Data Turbine User Guide of the version to be installed³.

This section presents some practical information regarding how to launch the Data turbine and the two data turbine utilities DaqToRbnb and AxisSource on a MS Windows machine. The latest version of the software available from https://www.nees.org/it/ at the time of production of the present report (see Fig. 2.2).

It is assumed that the default installation of the turbine has been made (in this case on C:\Program Files\RBNB\V3.1a and allowed for all users) and that the dtu installer has been allowed to install the dtu.bat script on C:\Windows (which is usually always on the default PATH). The script dtu.bat sets correctly the java CLASSPATH and starts the utility selected in the command line.

3.7.1.1 RBNB

There are two ways of launching the data turbine, for real-time purposes or for playback-only purposes.

- Real-time mode. In this mode, the data turbine receives data from various sources. If this is requested, some of these data are stored on the disk in an archive (directory tree). At ELSA, at the end of each experiment (which usually lasts for hours) RBNB is stopped and the archive is preserved. Note that in real-time mode, RDV can work either in real-time or in playback mode (for people joining the test after it started). The name of the archive is introduces by the –H keyword.

- Playback-only mode. After a test is completed and the RBNB archive correctly preserved, it is possible to start again RBNB, giving the archive file as input. In this mode, no source can be connected to RBNB and RDV can work only in playback mode. To start RBNB in playback mode, the –F keyword is used.

³ Since JRE is used implicitly by many processes for Web applications, it is likely that it is already present on the Windows machine.
This distinction is important and allows implementing various strategies for telepresence. It can be used prevalently in real-time (as for ELSA) to watch and participate to a (long) experiment. But for short events RBNB could be used only in playback mode, in slow motion, with data which are even not sent to RBNB in real time. In fact, the time stamps used by RBNB are not provided by RBNB when it received data, but by the source sending the data: this is why it is more important to have synchronised sources than having extremely fast connection between the sources and RBNB (a simple illustration of this is given in the practical section 3.7.4 regarding RDV). Stored data and video acquired during a short test by fast acquisition systems can be sent after completion of the test to RBNB, to be seen in playback mode by external users.

The .bat file presented in Fig. 3.9 allows to start RBNB in real-time mode (no argument, the archive is on D:\RBNB_archive\current), or in playback-only mode, giving a directory name. This directory name can be relative to D:\RBNB_archive or a full path name.

Note also that the message sent by RBNB are redirected to a log file (RBNB.log) located on the starting directory (folder) of the script.
Implementing a Telepresence Node: the experience at JRC

3.7.1.2 DaqToRbnb

According to the Data Turbine Utilities User Guide, “DaqToRbnb connects to DAQ server, subscribes to all data channels, converts the timestamps from ISO-8601 format to Java format, and then sends the data to the Data Turbine server”.

```bash
@echo off
rem ├─ Start script RBNB
rem │ On the "current" archive (on-line): Start_RBNB
rem │ On a saved archive (off-line)      : Start_RBNB archive_name
rem └─ ----------------------------------------------------------------
if "%1"="" goto current
set F=-F
set "RBNB_dir=D:\RBNB_archive\%1"
if exist "%RBNB_dir%" goto exist
set "RBNB_dir=%1"
if exist "%RBNB_dir%" goto exist
echo the archive in input does not exist
goto end
:current
set F=
set "RBNB_dir=D:\RBNB_archive\current"
if exist "%RBNB_dir%" goto exist
echo RBNB dir does not exist, it is created
mkdir "%RBNB_dir%"
:exist
set Current_dir=%CD%
cd C:\Program Files\RBNB\V3.1a\bin
  echo execute rbnb, see log file %Current_dir%\rbnb.log
  java -jar rbnb.jar %F% -H "%RBNB_dir%" >"%Current_dir%\rbnb.log" 2>&1
:end
```

Fig. 3.9 Start_RBNB.bat.
In contrast with older versions of DaqToRbnb, this process can be started at any moment, even if no data are sent from the Data Collector server. The process inspects from time to time the availability of the socket server running on the Data Collector server and connects when possible. When the socket server closes, some error messages are issued but the process still continues running, waiting for another possibility of connection. Note that DaqToRbnb listens for data coming from one source defined by one IP number and two port numbers. Then, if the data are not directly sent to the data turbine, the presence of multiple sources of acquisition data will imply the running of multiple DaqToRbnb processes. Note also that a choice has been made to run all the DaqToRbnb processes on the Telepresence server, but this is not mandatory. Finally, the data sent to DaqToRbnb should follow a strict format policy (see Section 3.7.2.1 and Hubbard 2004).

The .bat file presented in Fig. 3.10 allows starting DaqToRbnb. It lists first the different options available when launching DaqToRbnb. Those used at ELSA are: –S (to give the name MSAcquisition to the source), -r and -m (which allows to configure the ring buffer for 5 hours of DAQ, keeping 10% in memory), -q (to specify the IP bb1.bb2.bb3.bb4 of the Data Collector), -T (to group signals by time) and –t (to group the records every 200ms, which is the default period to collect data from DAQ at the data collector). Note also that DaqToRbnb is launched using the dtu.bat script.
In order to check if RBNB is working correctly it is important to start a data source on the same machine as RBNB. This is done using the source application delivered by the turbine at the moment of the installation. This can also be done using DaqToRbnb & FakeMSDAQ (see section 3.7.2.3), but without asking for an archive. This has been very useful for the following reasons.
Implementing a Telepresence Node: the experience at JRC

- To check of basic connectivity and functionality with a software architecture very close to the one of the DAQ
- To check time synchronization (this source and RBNB are naturally synchronized)
- To show something moving with RDV (sometimes a PSD test can be very boring)

Warning: although the default DAQ server is localhost (see Fig. 3.10), it is important to specify the IP number of the local machine when using DaqToRbnb together with FakeMSDAQ (or any other data collector process) running on the same machine. In a Microsoft environment, there is no direct mapping between localhost and the effective IP of the machine.

3.7.1.3 AxisSource

According to the Data Turbine Utilities User Guide, “AxisSource acquires data from an Axis camera or Video server and puts it into the ring buffer. By default, video is acquired at a resolution of 704x480 pixels at a rate of 30 frames per seconds”.

AxisSource is the listener for all the video camera (maximum 4) connected to the ELSA AXIS 241Q Video server (see annex I). Note that one video listener is associated to one camera. Then even if for instance two video cameras are connected to the same Video server, two AxisSource processes should be run in order to have two video channels sent to RBNB.

The .bat file presented in Fig. 3.10 allows starting AxisSource. It works with an optional parameter (default value 1), which indicate the number of the channel of the video server to be associated with the source. It lists first the different options available when launching AxisSource. At ELSA is used –n (to give the channel number associated with the source), –S (to give the name AxisSource_[channel number] to the source), -r and -m (which allows to configure the ring buffer for 5 hours of video, keeping 10% in memory), -A (to specify the IP cc1.cc2.cc3.cc4 of the Video Server), -U and -P (to give the username and password of guest account on the Video Server) and -f (to specify the expected number of frame per second). Note also that AxisSource is launched using the dtu.bat script.
@echo off
rem Start script for AxisSource
rem usage: org.nees.rbnb.AxisSource
rem -m percentage (%) of the ring buffer specified in -r to cache in memory *10.0
rem -A Video camera host name (required)
rem -C RBNB source channel name *video.jpg
rem -P password (no default)
rem -S RBNB Source Name *RBNBClient
rem -U username (no default)
rem -Z archive size *0
rem -c compression, percent, *25
rem -f frame rate, HZ, *30
rem -h Print help
rem -n camera number *1
rem -p RBNB Server Port Number *3333
rem -r length (in hours) to create the ring buffer for this source
rem -s RBNB Server Hostname *localhost
rem -v Print Version information
rem -z cache size *1024

set IP_SOURCE=cc1.cc2.cc3.cc4
set AXIS_USER=axisuser_name
set AXIS_PASS=axisuser_password
set AXIS_FR=30
if not ""%1""="""" goto axisnum
set AXIS_NUM=1
goto current

:axisnum
set AXIS_NUM=%1

:current
set "AXIS_NAME=AxisSource_%AXIS_NUM%"
set Current_dir=%CD%
echo execute AxisSource, see log file %Current_dir\%AXIS_NAME%.log
dtu AxisSource -A "%IP_SOURCE%" -U "%AXIS_USER%" -P "%AXIS_PASS%"
-r 5 -m 10 -n %AXIS_NUM% -s "%AXIS_NAME%" -f %AXIS_FR% > "%Current_dir\%AXIS_NAME%.log" 2>&1

Fig. 3.11 AxisSource.bat.
3.7.1.4 Log files

This section shows a short example of the content of the log files (for a one-minute-duration real-time session!). The following sequence of events is followed:

- 13:12:20: start RBNB
- 13:12:25: start DaqToRbnb (MSAcquisition)
- 13:12:28: start AxisSource (AxisSource_1)
- 13:12:30: start DAQ
- 13:13:02: stop DAQ
- 13:13:03: stop AxisSource (Crtl C)
- 13:13:15: stop DaqToRbnb (Crtl C)
- 13:13:20: stop RBNB (Crtl C)

Fig. 3.12 concerns RBNB. It indicates the starting time of RBNB and the location of the archive file., and then what is done at the archive level for the two sources AxisSource_1 & MSAcquisition (framesets, filesets, etc.). Note that when the DAQ and AxisSource are stopped, the data and video channels are disconnected. They are shutdown only when RBNB is stopped. This behaviour is related to the fact that an archive was requested for both the AxisSource_1 & MSAcquisition sources. If this had not been the case, the channels would have just been shutdown instead of disconnected. The fact that the data & video stored on archive are only disconnected when the source is stopped means that they can be reconnected. In the case of DAQ, there might be however problem at the display level when the number (or name) of the signal change during the same session of RBNB. It is a good practice to decide before hand, in collaboration with the potential user, what will be the broadcasted data channels and not change this during a test.
Fig. 3.12 `RBNB.log`.

Note `aa1.aa2.aa3.aa4` is the IP of the Telepresence Server.

Fig. 3.13 concerns `DaqToRbnb` as launched with the `DaqToRbnb_MS.bat` file (see Fig. 3.10). It indicates the starting time of `DaqToRbnb` and all activity once the connection with DAQ server (for us the Data Collector at IP `bb1.bb2.bb3.bb4`) is established. A first connection is attempted, which fails since the DAQ server is not ready. A new connection is then
attempted automatically after about 10s, with success. Then the log file indicates what is done to obtain the list of signals and then start to listen them. Note that after the DAQ is closed and until 
DaqToRbnb is stopped, it will try about every 10s to reconnect with the DAQ.

2010-06-20 13:12:25.369 CEST: Default time: Time Zone offset: -1.0
2010-06-20 13:12:25.385 CEST: Default time: Daylight Savings Time offset (in hours): -1.0
2010-06-20 13:12:25.385 CEST: Default time: Time Zone offset: -1.0
2010-06-20 13:12:25.385 CEST: Default time: Daylight Savings Time offset (in hours): -1.0
2010-06-20 13:12:25.385 CEST: Arguments to DaqToRbnb...
2010-06-20 13:12:25.385 CEST: DAQ: server = bb1.bb2.bb3.bb4; control port = 55055; data port = 55056
2010-06-20 13:12:25.385 CEST: RBNB: server = localhost:3333; source name = MSAcquisition
2010-06-20 13:12:25.385 CEST: Time offset (in seconds) = -7200.0, which is -2.0 hours
2010-06-20 13:12:25.385 CEST: Records from the DAQ will be flushed to RBNB every 200 milliseconds.
2010-06-20 13:12:25.385 CEST: The DataTurbine ring buffer will be created with 90000 frames and with a cache size of 900 frames.
2010-06-20 13:12:25.385 CEST: Use DaqToRbnb -h to see optional parameters
2010-06-20 13:12:25.385 CEST: Connecting to DAQ:
java.net.ConnectException: Connection refused: connect at ...
2010-06-20 13:12:36.338 CEST: Connecting to DAQ:
Created DAQ Control Port
ControlPort connected.
Enabled TCP keepalive on socket
2010-06-20 13:12:42.495 CEST: Set up connection to RBNB on localhost:3333 as source = MSAcquisition
2010-06-20 13:12:42.495 CEST: with RBNB Cache Size = 900 and RBNB Archive Size = 90000
Requesting DAQ channel list
For Channels, got:
Channel_0, Channel_1, Channel_2, Channel_3, Channel_4, Channel_5, Channel_6, Channel_7, Channel_8, Channel_9, Channel_10, Channel_11, Channel_12, Channel_13, Channel_14, Channel_15
Requesting DAQ unit list
For units, got:
>>Unit_0, Unit_1, Unit_2, Unit_3, Unit_4, Unit_5, Unit_6, Unit_7, Unit_8, Unit_9, Unit_10, Unit_11, Unit_12, Unit_13, Unit_14, Unit_15<<
Requesting DAQ upperbound list
For upperbounds, got: >>Unknown command 'list-upperbounds'<<
Invalid upperbound value.
Requesting DAQ lowerbound list
For lowerbounds, got: >>Unknown command 'list-lowerbounds'<<
Invalid lowerbound value.
2010-06-20 13:12:42.511 CEST: Preparing to listen to DAQ Channels: Channel_0, Channel_1, Channel_2, Channel_3, Channel_4, Channel_5, Channel_6, Channel_7, Channel_8, Channel_9, Channel_10, Channel_11, Channel_12, Channel_13, Channel_14, Channel_15
2010-06-20 13:12:42.511 CEST: With corresponding units: Unit_0, Unit_1, Unit_2, Unit_3, Unit_4, Unit_5, Unit_6, Unit_7, Unit_8, Unit_9, Unit_10, Unit_11, Unit_12, Unit_13, Unit_14, Unit_15.
2010-06-20 13:13:02.527 CEST: Closing Sockets...
2010-06-20 13:13:02.527 CEST: Closing RBNB connection...
2010-06-20 13:13:12.652 CEST: Connecting to DAQ:
java.net.ConnectException: Connection refused: connect at ...
Terminate batch job (Y/N)? Terminate batch job (Y/N)? ^C
Fig. 3.14 concerns **AxisSource**. It indicates when the process is started and the transaction with RBNB. About every second, it indicates the averaged number of frames per second which are sent to RBNB. Note that this number is not uniform (together with the duration of the time interval between two frames), and in this case always smaller than what was expected (30).

```
2010-06-20 13:12:29.276 CEST: Connecting to RBNB server with... RBNB Server = localhost:3333;
RBNB Cache Size = 1024; RBNB Archive Size = 540000; RBNB Source name = AxisSource_1
2010-06-20 13:12:29.370 CEST: contentType :multipart/x-mixed-replace; boundary=--myboundary
2010-06-20 13:12:29.448 CEST: Average frames per second: 28
2010-06-20 13:12:30.448 CEST: Average frames per second: 26
2010-06-20 13:12:31.416 CEST: Average frames per second: 25
2010-06-20 13:12:32.338 CEST: Average frames per second: 26
2010-06-20 13:12:32.448 CEST: Average frames per second: 26
2010-06-20 13:12:33.635 CEST: Average frames per second: 27
2010-06-20 13:12:34.510 CEST: Average frames per second: 26
2010-06-20 13:12:35.542 CEST: Average frames per second: 26
2010-06-20 13:12:36.307 CEST: Average frames per second: 25
2010-06-20 13:12:37.323 CEST: Average frames per second: 25
2010-06-20 13:12:37.635 CEST: Average frames per second: 26
2010-06-20 13:12:38.667 CEST: Average frames per second: 26
2010-06-20 13:12:40.307 CEST: Average frames per second: 25
2010-06-20 13:12:40.745 CEST: Average frames per second: 26
2010-06-20 13:12:41.182 CEST: Average frames per second: 27
2010-06-20 13:12:41.792 CEST: Average frames per second: 26
2010-06-20 13:12:42.276 CEST: Average frames per second: 26
2010-06-20 13:12:42.854 CEST: Average frames per second: 26
2010-06-20 13:12:46.042 CEST: Average frames per second: 26
2010-06-20 13:12:47.042 CEST: Average frames per second: 26
2010-06-20 13:12:47.370 CEST: Average frames per second: 25
2010-06-20 13:12:48.604 CEST: Average frames per second: 24
2010-06-20 13:12:50.167 CEST: Average frames per second: 26
2010-06-20 13:12:52.230 CEST: Average frames per second: 26
2010-06-20 13:12:53.261 CEST: Average frames per second: 26
2010-06-20 13:12:54.183 CEST: Average frames per second: 27
2010-06-20 13:12:57.433 CEST: Average frames per second: 26
2010-06-20 13:12:58.511 CEST: Average frames per second: 27
2010-06-20 13:13:00.355 CEST: Average frames per second: 25
2010-06-20 13:13:00.542 CEST: Average frames per second: 26
2010-06-20 13:13:01.402 CEST: Average frames per second: 25
2010-06-20 13:13:01.589 CEST: Average frames per second: 26
2010-06-20 13:13:03.355 CEST: Average frames per second: 25
2010-06-20 13:13:03.542 CEST: Average frames per second: 27
2010-06-20 13:13:03.542 CEST: Average frames per second: 26
2010-06-20 13:13:03.542 CEST: Error: Failed to load data: This operation requires a connection.
Failed to connect to the RBNB server.
Terminate batch job (Y/N)?
```

Fig. 3.14 **AxisSource_1.log**.
### 3.7.2 daq2rbnb & FakeMSDAQ

The data collector process is a Windows process that has been developed in-house. It takes the data from controllers and/or data acquisition units and casts them under a format readable by the DaqToRbnb Java process listening on the Telepresence server and feeding in turn the data turbine.

The use of this indirect strategy (use the intermediate process DaqToRbnb to feed the data turbine) is motivated by the fact that the data structures handled by DaqToRbnb are well defined (Hubbard 2004) and that a data acquisition tester is available for basic tests.

The dndtester can be downloaded from NEESforge (a NEES community code repository for sharing code and collaborating with other developers) at [http://neesforge.nees.org/projects/dndtester/](http://neesforge.nees.org/projects/dndtester/).

#### 3.7.2.1 What DaqToRbnb is waiting for

DaqToRbnb (running on the Telepresence server) is a program that can be started at any moment (even if neither the data turbine nor the acquisition is running). Basically, it verifies from time to time that a socket connection is available on a given host at two given ports (the so-called control port and the data ports), introduced by the options -q, -c and -d of the DaqToRbnb command line (see Annex E.4). Note that DaqToRbnb is a socket client.

When the sockets are available, DaqToRbnb uses a protocol in order to send ASCII messages on the command socket, waiting for the answer at any moments (Hubbard 2004). The main commands of the protocol are:

- **daq-status**: Error, Offline, Unknown, Running or Stopped
- **list-channels**: Active channel listing, comma delimited
- **list-units**: Active channel unit list, comma delimited
- **open-port{channel name}**: Start subscription to given channel
- **close-port{channel name}**: Stop subscription to given channel
The three first commands allow DaqToRbnb to verify that the acquisition is ready, to know the name of the signals and to get the units of these signals.

The fourth command asks the Daq to start to send, on the data socket, time stamps followed by the values of all the signals already subscribed.

The fifth command allows unsubscribing from a signal (it is not used yet in practice).

3.7.2.2 daq2rbnb

The minimum requirements for the process collecting the data on the controllers or data acquisition units is thus to:

- Create the control and data sockets, listen on the sockets and accept the connection (when requested).
- Implement in an infinite loop the data protocol on the command socket.
- As soon as a signal is subscribed, start a timer feeding the data socket with the correct data.

Since daq2rbnb is also equipped with a user interface for selecting the signals, to pause and even stop the acquisition, the following architecture has been chosen.

- Selection zones: The upper part of the user interface is devoted to the selections of the controller and/or DAQ units to be connected with (Connections), and to the definition of the signals to plot, together with a description of the current settings (Parameters). This corresponds to the main thread of the application and is implemented using usual MS .NET features.
- Running zones: when all what is needed has been chosen, the connection with the sister application DaqToRbnb application together with the sending of data can be start, pause/run and stop. This corresponds with two threads that have to share information
together, and that are launched by the main application. This requires slightly more
advanced use of MS .NET features.

Despite that fact that Daq2rbnb’s programming features are site-specific to ELSA, its source is
available for the SERIES partners on request. The interface has been implemented using standard
MS .NET forms. The programming is made in C#.

3.7.2.3 FakeMSDAQ

In order to share experience with other MS Widows users in SERIES, a simpler application,
called FakeMSDAQ is available for the SERIES partners on request. This application, which
does not require any acquisition system, is based on the same architecture as daq2rbnb. The user
interface of this application, at different stages is shown in Fig. 3.15.
The application is also made of a Selection part (Connection & Parameters) and a Running part (Command & Status).

- The Connection choice only regards the number of the port on which the control socket server will be opened (this number, incremented by one, is to be used for the data socket) server. The default is 55055. This parameter needs to be validated in order to proceed.
Implementing a Telepresence Node: the experience at JRC

- The parameters choice regards the number of fake channels (default 16), the basic period of the fake signals (default 5 seconds), and the sampling period of the data sending to the turbine (default 200 ms). These parameters need to be validated in order to proceed.
- When the selection process is finished, it is possible to Start, Pause/Run and Stop the data flow exactly as in daq2rbnb. The data sent are generated according to the input parameters as sinus signals

\[ \sin 2\pi(t/T)\sqrt{i}, \quad i \in [1, N] \]

where \( t \) is the absolute time and \( i \) the index of the fake signal.

FakeMSDAQ has been also developed and implemented using standard MS .NET forms and C#. The code source and further explanations are available for SERIES partners on request.

### 3.7.3 NTP

The Network Time Protocol (NTP) is a protocol designed to synchronize the clocks of computers over a network. As suggested in (Hubbard 2006), NTP is used to synchronize RBNB and all its sources, namely in the present case, the Telepresence server, the Data Collector server and the Video server. Due to the network architecture adopted, this task is not a priori obvious. For instance the machines located on the Intermediate Telepresence network are not able to connect to the public network, and thus are unable to communicate with the default NTP servers.

However, all the machines that should be synchronized are able to connect the Secure Hybrid System NTP time server running at the JRC (see http://time.jrc.it/). The name of the machine is time.jrc.it and its IP address on the DMZ is also known.

In practice, for the Windows machine, the NTP software is not directly available for windows machines (the official version is a compressed tarball to be deployed using a make file). Using the “NTP Public Services Project links page” from the NTP official site (http://www.ntp.org), it is possible to find recent installation for Windows.
During the installation, it is asked what user should run the NTP daemon (ntpd), noting that administrator is not the best candidate. The suggestion is to let the installer create a NTP user, owning further on the ntpd process. Note that this general method fails on a Windows Server machines, and more advanced administrative skills are then needed.

The `ntpd` process uses a configuration file called `ntpd.conf`, located in the etc folder of the NTP installation (for instance `C:\Program Files\NTP\etc`). This file indicates which servers to connect to perform the synchronization, and default content is provided, depending on the location of the installation.

These lines were commented and we just had to add a line at the end (the `iburst` keyword speeds up initial synchronization):

```
# Use specific NTP servers
server time.jrc.it iburst
```

For the machines located on networks without Domain Name Server, the IP number should be used (this is the case of the Data Collector).

### 3.7.4 Remark about RDV, the Real time Data Viewer

As already discussed, RDV is a sink which gives the possibility to a remote user to visualize in real-time or in play-back modes the data of the ring buffer.

Some troubles may arise with the real-time mode, even if all the sources are correctly synchronized. Consider the following situation involving the Telepresence server (running RBNB, AxisSource & DaqToRbnb), the Axis Video Server and the Data collector (running FakeMSDAQ). The Video camera films the screen of the Data collector where the current status of the FakeMSDAQ is shown in front of what can be seen on RDV (see Fig. 3.16).
Fig. 3.16 Delays in RDV.
The sampling rate used for FakeMSDAQ is 20ms in this case. It can be seen that there is a delay of $(13025-13017)\times 20\text{ms} = 160\text{ms}$ given by the difference of the current count and the one displayed by RDV. This delay is more or less what is missing in the data windows (red curve) for occupying it completely (which is the case when no video sources are present). This behaviour can be explained by the fact that a video frame with similar time stamps as data record arrives later in RBNB which sends to RDV the most recent synchronized data. If there is too much delay between the various sources to reach RDV, some erratic behaviour may occur such as sudden jumps in the data window, data not shown or shown incorrectly. A good practice to eliminate this disorder is to work with a decently large time windows (at least 20 times larger than this kind of delay).

This kind of behaviour can only worsen in the case the sources are not synchronized between each other, and are not synchronized with RBNB itself.
Implementing a Telepresence Node: the experience at UNIVBRIS

4.1 INTRODUCTION

Section 4.2 details the hardware procured by UK NEES @ Bristol to provide telepresence. The network architecture employed by UK NEES @ Bristol is also introduced.

Integrating a research infrastructure’s pre-existing ‘legacy’ systems within the NEES framework is perhaps the greatest challenge to the implementation of telepresence. Section 4.3 of this document illustrates how this integration can be achieved again by taking UK NEES @ Bristol as a case study. Since legacy systems are site specific, there seems little point in providing specific details of the site-specific programming code. Instead, the focus here is on the top level system design. It is hoped that with such an overview, a proficient Java programmer would be able to utilise the ideas contained. If required, however, copies of any UK NEES developed code and corresponding implementation notes are available on request for SERIES partners. Section 4.4 of this document addresses the different tools deployed to handle streaming and recording of video signals.

It is worth noting that since 2004 the uptake of NEES telepresence tools within the US earthquake engineering community has been slower than anticipated (Sritharan and Shield, 2008). This has been put down in part due to the added inconvenience to researchers associated with their utilisation. In response, UK NEES has significantly improved their usability. As a part of this drive, UK NEES @ Bristol has developed a web application to operate, monitor and manage the telepresence tools. This is introduced in Section 4.5. Once again, copies of the code behind the UK NEES @ Bristol web application are available on request for SERIES partners.
Finally, and in addition to the US NEES telepresence tools, the UK NEES partnering institutions have attained Access Grid videoconferencing capabilities to facilitate cost-effective and time-efficient collaboration. The UK NEES @ Bristol Access Grid Node (AGN) is introduced in Section 4.6.

Additional practical details are given in Section 4.7.

4.2 UK NEES @ BRISTOL: TELEPRESENCE HARDWARE AND NETWORK DESIGN

The telepresence systems introduced at each node of UK NEES infrastructures are essentially identical. To avoid repetition, this document focuses on the implementation by UK NEES @ Bristol.

4.2.1 Hardware

The equipment acquired for the UK-NEES system has been specifically chosen to integrate as closely as possible with the US system. Each of the three UK-NEES sites has a common base configuration as per the standard NEES specification, augmented to suit site-specific requirements. The system in Bristol comprises two Linux based servers: the NEES Point of Presence (NEESpop) server to run handle data acquisition signals, and the NEES Telepresence Management server (NEEStpm) to handle video. The NEESpop server has dual xeon processors, 4GB memory, 3.6TB of RAID disk space and a tape library backup system. The NEEStpm server has dual xeon processors, 8GB memory and 836GB of RAID disk space. It is worth noting that the deployment of separate servers to handle video and data acquisition is not mandatory. If a single server is instead deployed, performance monitoring should be undertaken since the video processing at adequate resolution and frame rate is extremely resource-hungry.

Defying the US NEES convention of utilising Axis hardware, UK NEES provides video support for telepresence with Sony network cameras (model SNC-RX550P). These cameras minimise blind spots with a 26x optical zoom, 360° endless pan, and 90° tilt. They also have high
Implementing a Telepresence Node: the experience at UNIVBRIS

sensitivity so they can be used in low light conditions (minimum illumination 0.15 lux). These features make the cameras suitable for use in large-scale laboratory environments. The cameras have a built in network card and password-protected web server where generated video streams can be viewed, controlled and administered. Additionally, UK NEES @ Bristol has deployed an Axis 2110 static (i.e. no PTZ) video camera.

4.2.2 Network

A diagrammatic representation of the UK NEES @ Bristol network is displayed as Fig. 4.1. The backbone of the network is the 1Gbit/s Departmental Local Area Network (LAN) which is connected to the UK government funded Joint Academic Network (JANET) through the University of Bristol firewall. Highlighted in green are legacy systems predating UK NEES: controller, a test machine, a collection of sensors and the associated signal processing equipment. The legacy data acquisition (DAQ) system requiring integration within the NEES framework is shown in yellow. Hardware procured as part of UK NEES is enveloped in red: the NEESpop server, the NEEStpm server and the assortment of network video cameras. Figure 3 indicates the principle NEES processes running on each of the UK NEES computers (NEESpop, NEEStpm and DAQ). System security is maintained using a dual layer of firewalls. The University firewall is configured to allow HTTP and HTTPS traffic to NEESpop and NEEStpm (necessary for FlexTPS and the UK NEES web application). Additionally port 3333, the RDV default, is open on NEESpop to TCP traffic. In addition, each element of networked hardware has its own firewall off the departmental LAN. The ports opened through these firewalls necessary for system function are as indicated in the diagram.
Implementing a Telepresence Node: the experience at UNIVBRIS

Fig. 4.1 UK NEES @ Bristol network diagram.
4.3 UK NEES @ BRISTOL: DAQ TO DATA TURBINE VIA SIMACQDRIVER

US NEES sets forth a framework for integration between site specific acquisition software and the Data Turbine based on a single data acquisition software platform, National Instruments LabVIEW. Since LabVIEW is not utilised at the University of Bristol NEESdaq is not deployed. Instead the NEES framework has been adapted to fit legacy systems.

The data acquisition (DAQ) systems at Bristol use a bespoke program called SimAcq which is designed to write acquisition data to a MATLAB-executable (.m) ASCII text file. Running on IBM-PC compatible computers, SimAcq issues DAPL to initialise and trigger a 64 channel Microstar Laboratories simultaneous data acquisition system. SimAcq software is written in the Aglient VEE environment, a modular-type programming environment wherein the user draws process flow diagrams (much like the better known Simulink) and creates graphical user interfaces (GUIs). The SimAcq GUI is displayed in Fig. 4.2.

![SimAcq GUI](image)

**Fig. 4.2 SimAcq (version 3) graphical user interface.**
SimAcq runs as a two stage process. The ‘Initialise’ GUI button runs the program and initiates stage one. During this stage the desired acquisition parameters (test name, number of channels, sampling frequency, duration, notes, etc) are inputted via the GUI. The operator then uses a ‘Set Parameters’ GUI button which: configures the data acquisition cards, creates a text file in which to write data, makes the ‘Start Acquisition’ GUI button live, and places the program on standby, ready for the second stage of the acquisition process. When the test operator is ready, the ‘Start Acquisition’ button is pressed. The data acquisition cards are immediately activated and acquisition data are written in the text file.

Under UK NEES, SimAcq (now at version 3) has been modified so that in addition to creating a local ASCII text data file, it is able to write data to the Data Turbine. This functionality is activated via the SimAcq GUI ‘Stream to RBNB’ toggle. In addition, a new multi-threaded Java daemon program called SimAcqDriver has been developed by UK NEES @ Bristol. SimAcqDriver mediates between SimAcq and the Data Turbine utility DaqToRbnb. The high-level system design is depicted in Fig. 4.3. The solid boxes represents hardware, dotted boxes represent processes. All NEESpop programs are written in Java and run as separate Java processes. All inter-process communication is via TCP sockets.

![Fig. 4.3 The high level design of SimAcqDriver.](image-url)
SimAcqDriver runs permanently in the background listening for new requests from SimAcq. When a new acquisition run is initiated, it receives setup messages from SimAcq (to which it has a single connection) and launches an instance of DaqToRbnb (to which it has two connections: control and data). It then acts as a buffer, receiving acquisition messages from SimAcq before sending them on to DaqToRbnb. DaqToRbnb assembles data into frames which it passes to the Data Turbine. At the end of acquisition DaqToRbnb is shut down in a controlled manner and the SimAcqDriver waits for the next acquisition. More information about the SimAcqDriver communication protocol is available on request to SERIES partners.

UK NEES @ Bristol has found adequate performance if 1024 MB of memory is allocated to the Data Turbine during start up (using the RBNB_MEM setting in the start up script contained in /etc/init.d/RBNB). Other Data Turbine properties are configured automatically between stages one and two of the SimAcq process. This is achieved using only the information inputted to the SimAcq GUI. Thus, the operator does not require intimate knowledge of either the Data Turbine or its command line driven utilities (such as DaqToRbnb). Data Turbine metadata is first configured: the source name (the locally-stored ASCII text data filename is adopted as the Data Turbine source name), the channel names (inputted to the GUI), and the channel units (also inputted to the GUI). Note that a list of calibration factors inputted to the SimAcq GUI is applied before data is passed to SimAcqDriver in order to convert data from voltages to engineering units. If channel name, channel units and calibration factor lists are not inputted, defaults are adopted (chan 1, chan 2, …; volts, volts, …; 1, 1, …). Additionally, and as detailed in Section 4.7.1, by adopting a hardcoded flush rate of 100ms SimAcqDriver automatically calculates and configures the Data Turbine frame size, archive size and cache size. Thus, the correct calculation and implementation of these critical parameters occurs behind the scenes and is of no concern to the operator.

4.4 UK NEES @ BRISTOL: VIDEO

Four Sony network cameras have been installed at UK NEES @ Bristol. The cameras are housed within protective casings to prevent contamination or damage that may result from being sited in the aggressive environment of an earthquake engineering laboratory. Two cameras are
permanently secured to look along the principle horizontal axes (x and y) of the shaking table. The remaining two cameras are not permanently attached and can be deployed wherever is required within the EQUALS laboratory. UK NEES @ Bristol utilises a fifth camera manufactured by Axis to provide a birds-eye-view of the laboratory. The internal web servers of all the cameras have been password protected and can only be accessed from the internal departmental LAN.

4.4.1 Live video

UK NEES provide access to the video streams via the US NEES FlexTPS software. As introduced previously, FlexTPS consists of a web interface called the portal. A user connects to the portal through a web browser (http://nees-centpm.cen.bris.ac.uk/portal) and, depending on authentication, can view and administer one or more video streams. Fig. 4.4 presents a screen shot of the FlexTPS @ University of Bristol web interface.
Fig. 4.4 FlexTPS @ University of Bristol user interface (logged in with the high level user account details).

Video streaming and robotic control to FlexTPS is mediated using proxies. Since the US NEES software was developed for Axis hardware, a new driver program (called SNCRX550P.pm) was developed by UK NEES @ Bristol to enable robotic control of the Sony cameras. (To instantiate the new driver, the source-type should be set to Sony:SNCRZ30N within the portal’s xml files.)

To comply with the 1998 Data Protection Act and to reassure the laboratory workers and visitors whose images may be captured, UK NEES @ Bristol developed a code of practice to govern the use of the telepresence video system (see Section 4.7.2). As indicated in Table 4.1 access is
controlled with user accounts. The public account does not require a userid/password and provides access to only the static Axis camera. Frame rates are limited to 1Hz and the resolution is insufficient to identify or monitor the activities of individuals. The low level account requires userid/password information and gives viewing (but no control) rights to all five cameras at frame rates up to 25Hz. The high level account again requires userid/login information and grants viewing and control rights to the user.

<table>
<thead>
<tr>
<th>Camera</th>
<th>Public</th>
<th>Low level user account</th>
<th>High level user account</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>View</td>
<td>View</td>
<td>View</td>
</tr>
<tr>
<td>Sony</td>
<td>No access</td>
<td>View</td>
<td>View + Control</td>
</tr>
</tbody>
</table>

4.4.2 Video recording using SonySource

US NEES purports that FlexTPS can be used to record video streams to disk. However, UK NEES @ Bristol had problems with implementing this functionality. While it was possible to create a compressed archive of jpeg image files using FlexTPS, “ffmpeg: unrecognized option ‘-hq’” errors are generated during mpeg creation. ffmpeg is the mpeg creation program. It appears that the code that calls ffmpeg is incorrect as ffmpeg can be used successfully to convert previously saved jpegs. However, ffmpeg does not allow the frame rate to be specified so anything less than 25fps will play faster than real time using standard video viewers such as Windows Media Player.

As a result of these issues, UK NEES @ Bristol developed a bespoke solution for managing video recording. An instance of Data Turbine is installed on NEES TPM to support video recording that has become known as the Media Turbine. To comply with the Data Protection Act, access to the Media Turbine is restricted (using the RBNB-accesscontrol file in /usr/share/java/RBNB-3.0). This is an intranet type application with no access outside the Departmental LAN and password restriction within the LAN. Port 3333 to the NEES TPM has
been opened to TCP traffic from the Departmental LAN to allow local RDV access to the Media Turbine. Should pre-recorded video files be suitable for wider distribution (i.e. the data is not confidential, personnel/visitors appearing in the video give their permission, etc), then they can be copied across to the public Data Turbine (on NEESpop) at the owner’s discretion. Using the Real time Data Viewer (RDV), a video recording can then be accessed remotely.

Due to the large volume of video data, if insufficient bandwidth or memory is available playback using RDV can appear staccato. This poses a problem. Local researchers require video be recorded at maximum resolution to aid analysis while remote researchers require a lower resolution to aid viewing via RDV. UK NEES @ Bristol has attempted to overcome this issue by introducing a scaling factor between the Media and Data turbine. The Media Turbine records video at default VGA (640x480) resolution at up to 25 frames per second. This scale factor determines the amount of compression is applied when copying between the Media Turbine and the Data Turbine. Larger scale factors promote smoother playback by reducing data volumes. In addition, Media Turbine sources can be extracted and converted into mpeg files thereby guaranteeing smooth playback. Note also that the 2048MB of memory allocated to the Media Turbine (in the startup script /etc/init.d/RBNB) is double that the one allocated to the Data Turbine to allow for the relatively large volumes of video data.

AxisSource.jar, the US NEES utility for adding video data to the Data Turbine, is designed for Axis hardware. UK NEES @ Bristol have developed an equivalent tool for Sony hardware. SonySource configures the Media Turbine (in terms of source name, channel names, frame rate, cache and archive size) and writes video data to the Media Turbine. The rules used to determine Media Turbine cache and archive size are presented in Section 4.7.3.

Operation of the video recording system is via the UK NEES @ Bristol web application.

4.5 UK NEES @ BRISTOL: WEB APPLICATION

The UK NEES @ Bristol web application (nees.bristol.ac.uk) meets the needs of two distinct different user groups: telepresence users and researchers. Telepresence users are provided with
links to FlexTPS @ University of Bristol and to a page where RDV can be downloaded. Researchers are provided with an interface to manage and administer both the Data Turbine and the Media Turbine and to operate video recording. They need not concern themselves with the intricacies of the command-line-driven utilities but can instead drive the suite of telepresence tools through the web application’s self-explanatory menus. This marks a significant usability improvement.

NEESpop serves the public home page which is displayed as Fig. 4.5. The menu (on the left hand side) is split into three sections. The first section is intended for telepresence users and provides links the home page, to FlexTPS @ University of Bristol and to a page where RDV can be downloaded. The second section, entitled Administer@cennees-pop, links to userid/password protected pages to administer the Data Turbine (on NEESpop). The third section, entitled Administer@cen-nees-tpm, provides links to userid/password protected pages to administer the Media Turbine (on NEESTpm) and support video recording (on NEESTpm). Access to sections two and three is restricted to the Department of Civil Engineering of the University of Bristol LAN.
Implementing a Telepresence Node: the experience at UNIVBRIS

The web application is designed around a ruby cgi application. Web pages are built dynamically using eruby to run ruby code. Where there is an interface to the operating system or to the Data Turbine, bash script files are invoked.

4.5.1 Administer @ cen-nees-pop

The ‘Acquisition Data Turbine’ link prompts for userid/password before access is granted to the webpage displayed as Fig. 4.6.
Within this page, the user can:

- View the sources (and corresponding channels) loaded within the Data Turbine.
- View all available source archives.
- Load and unload sources.
- Permanently delete sources from the list of available archives (thereby freeing disc space on NEESpop).
- Manage the Data Turbine utilities (SimAcqDriver, DaqToRbnb and the Data Turbine itself) including their starting and stopping, and viewing log files.

This functionality is achieved using a number of UK NEES @ Bristol developed Java/Ruby programs:
Implementing a Telepresence Node: the experience at UNIVBRIS

- deletesources; run deletesourcesruby script to delete archive directories from disk.
- Listloadedsources; list of all sources/channels in data turbine.
- Listloadedvideosources; list of video sources/channels in data turbine.
- Managesources; load or unload sources to data turbine.
- manage_RBNB; run the service command to stop or start RBNB.
- manage_SimAcqDriver; run the service command to stop or start RBNB.

4.5.2 Administer @ cen-nees-tpm

All links under the Administer@cen-nees-tpm menu are served by NEEStpm via a specified port. The NEEStpm firewall is configured to allow only computers on the Department of Civil Engineering of the University of Bristol LAN can access to this port. Additionally, access requires userid/password identification. The functionality contained within these pages is provided by the following files:

- video-RBNB-utilities.jar; holds SonySource (video recording to the Media Turbine), VideoSink (part of the processing chain to generate mpg files), VideoCopyResize (copy and optionally resize video from the Media Turbine to the Data Turbine).
- admin-RBNB-utilities.jar; holds ListLoadedSources (list sources in the Media Turbine) and ManageLoadedSources (load and unload from the Media Turbine).

Additionally, the pages allow the following bash scripts to be invoked:

- deletesources; run deletesourcesruby, the script to delete data turbine archive directories from disk
- recoRDVideototurbine; run SonySource to record video.
- extractvideofromturbine; run VideoSink to get images out of data turbine.
- createtitle; run convert utility to generate title images for mpg.
• createvideo; run ffmpeg utility to generate mpg from images.
• listloadedsources; run ListLoadedSources to list sources.
• managesources; run ManageLoadedSources to load/unload sources.
• manage_RBNB; run the service command to stop or start RBNB.

The ‘Video Data Turbine’ grants access to the admin page for the Media Turbine. This page is the equivalent of Fig. 4.6 but for the Media Turbine. Within this page, the user can:

- View the video sources (and corresponding channels) loaded within the Media Turbine.
- View all available video source archives.
- Load and unload video sources.
- Permanently delete video sources from the list of available archives (thereby freeing disc space on NEEStpm).
- Manage the Media Turbine (starting and stopping, and viewing log files).

The ‘record video’ link grants access to the page displayed in Fig. 4.7. Here, the user identifies the video streams to be recorded, the test name (which is adopted for the Media Turbine source name), the recording duration and the frame rate (which SonySource uses to configure the Media turbine cache and archive size). The record button activates the recording process.
Fig. 4.7 UK NEES @ Bristol web application: Record video.

The ‘Generate mpeg’ link grants access to the page presented as Fig. 4.8. Here, the user identifies a channel in a loaded Media Turbine source from which to generate an mpeg. A title screen is added to the video stream giving the source name, the channel name, the start timestamp and two lines of optional text. (The convert program which creates the title frames is supplied with an install of ImageMagick 6.07.) A list of generated mpegs can be viewed using the ‘list mpeg video files’ link from where they can be downloaded from NEES TPM.
Implementing a Telepresence Node: the experience at UNIVBRIS

Fig. 4.8 UK NEES @ Bristol web application: Generate mpeg.

Finally, the ‘Copy and resize video’ link grants access to the page displayed as Fig. 4.9. Here, any source loaded onto the private Media Turbine can be copied across to the public Data Turbine. The user has the option of compressing the Media Turbine jpegs to reduce network load. Such compression improves Data Turbine / RDV performance (making video streams appear less staccato) at the expense of video quality.
4.6 ACCESS GRID

UK NEES utilises the Access Grid (www.accessgrid.org) to facilitate cost-effective, time efficient collaboration between the three partnering institutions. Developed as a response to the deficiencies of H.320/H.323 videoconferencing by the Argonne National Laboratory in the USA, the Access Grid utilises the internet to provide a multi-party room-to-room collaboration environment. It is best seen as an advanced form of videoconferencing wherein live video, audio and desktop applications can be shared amongst the geographically-separated groups assembled at a ‘virtual venue’. The Access Grid is accessible using standard computers and standard internet connections.
Room-to-room collaboration is inherently more complex than peer-to-peer collaboration. Multiple network cameras are required at each Access Grid Node (AGN) to adequately capture video of multiple meeting participants. Thus, each node generates multiple video streams. Large format displays are required to view all generated video content. Additionally, to negate the necessity of headsets, echo-cancellation hardware is required so that loudspeaker output picked up by microphones is not transmitted back to remote participating nodes. Careful interior design is instrumental to the function of an AGN. Both cameras and display should be at eye level and audio emitted from loudspeakers should emanate away from the display. Audio performance can be improved by increasing acoustic absorption to reduce reverberation. A neutral backdrop to meeting participants reduces visual noise.

While the Access Grid software is free and open source, UK NEES invested in proprietary software to provide Access Grid capability. The IOCOM (www.iocom.com) Grid Node (IGN) software gives the Access Grid an intuitive graphical user interface. Moreover, IOCOM can provide a wide range of IGNs, from desktop-based ‘personal’ nodes to total room solutions. UK NEES @ Bristol has two IGN facilities. The first is a two-camera facility in the shaking-table control room to enable effective participation of the shaking table operator in Access Grid meetings. Secondly, a larger three camera facility has been installed in an adjacent meeting room to support interactions of larger groups of researchers. The layout of this IGN is depicted in Fig. 4.10. A UK NEES meeting in progress is depicted in Fig. 4.11.
Implementing a Telepresence Node: the experience at UNIVBRIS

Fig. 4.10 The three-camera Access Grid Node at UK NEES @ Bristol.
4.7 PRACTICAL DETAILS

4.7.1 Automatic calculation of Data Turbine cache and archive

SimAcqDriver mediates between SimAcq and Data Turbine. The following rules have been coded to avoid having to calculate cache and archive sizes:

- Use fixed number of samples per frame.
- Samples per frame = sample rate/10 (want flush approx every 100ms)
- Limit samples per frame to min of 1, max of 100
- Total number of frames = (sample rate * duration / samples per frame) + 100
- Set archive to total number of frames
- Set cache to total number of frames unless estimated total size > 512Mb. If limit is exceeded, reduce cache size to fit under limit.
4.7.2 Code of Practice governing CCTV in the EQUALS laboratory

4.7.2.1 Initial scheme assessment

The EPSRC funding UK NEES project makes provision for (five) network cameras and associated hardware and software to be installed within the Earthquake and Large Structures (EQUALS) lab at the University of Bristol (UoB). The primary purpose of the system is to foster research collaboration between Bristol and its partners by providing unprecedented access to concurrent research activities. The scheme provides additional outreach opportunities.

The purpose of the scheme is to:

- Monitor experimental setup.
- Monitor and record experiments.
- Provide and overview of research activity (outreach).

The scheme is not to be used to monitor the movements or activities of personnel.

4.7.2.2 Camera types and siting

The scheme requires two levels of functionality. Firstly, robotic Pan Tilt Zoom (PTZ) capability to allow authorised users to scrutinise the setup of specific tests in detail and record progress as the test is executed. Second, a fixed view across the entire laboratory to give an overview of all current research activity. To meet the objectives, two camera types will be installed: 4 x Sony RX550P (PTZ) Cameras, 1 x Axis 2110 (static) camera. Siting of these cameras will be as indicated in Table 4.2.

Table 4.2 Siting of network cameras in the EQUALS lab

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Siting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cen-nees-cam01</td>
<td>Axis 2110</td>
<td>Outside Rm 1.90 at height 12m</td>
</tr>
<tr>
<td></td>
<td>Camera Code</td>
<td>Camera Type</td>
</tr>
<tr>
<td>---</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>2</td>
<td>Cen-nees-cam02</td>
<td>Sony RX550P</td>
</tr>
<tr>
<td>3</td>
<td>Cen-nees-cam03</td>
<td>Sony RX550P</td>
</tr>
<tr>
<td>4</td>
<td>Cen-nees-cam04</td>
<td>Sony RX550P</td>
</tr>
<tr>
<td>5</td>
<td>Cen-nees-cam05</td>
<td>Sony RX550P</td>
</tr>
</tbody>
</table>

Notes:

- Cam 01 will be set up such that it has the widest possible view to avoid recording any identifiable faces and will have a low frame rate (circa 1fps) to avoid movement tracking. A Still taken from Cam01 is displayed as Fig. 4.12 indicating the level of lab worker anonymity provided.
- The siting of Cam02 and Cam03 will ensure that they are unable to view inside the technicians’ office.
- The demountable cameras Cam04 and Cam05 should be sited in such a way that they monitor only those spaces which are intended to be covered. Provisions should be made so that they cannot view unintended spaces such as the technicians’ office.

Fig. 4.12 Test picture from the EERC lab high level camera.
4.7.2.3 Camera access

Access to the PTZ cameras will be controlled using user names and passwords. (User accounts and passwords will be set by a person with appropriate authorisation.) Users will fall into one of four groups:

- System Administrator.
- Resident technicians of the Equals lab.
- Local users – authorised UoB personnel who have an **intellectual interest** into the ongoing research.
- Remote users – e.g. offsite UoB personnel, personnel at other institutions, commercial clients, schools, or other UoB partners who have interest in an ongoing lab activity.

Access to the static camera will be uncontrolled, giving rise to another user group:

- Public users – non-authorised UoB personnel and members of the general public.

Cameras can be accessed in two ways:

- The cameras can be viewed directly by the System Administrator for maintenance purposes only.
- The cameras will stream live video images to the Tele Presence Management (TPM) server which can display the cameras’ output on a webpage. This is the normal way in which all user groups will view the cameras.

Three levels of access restrictions will be employed. In order of increasing power:

- *No access* – camera streams cannot be viewed. PTZ cameras cannot be controlled.
- *Viewing* rights granted – camera streams can be viewed. PTZ cameras cannot be controlled.
- *Control* rights granted – camera streams can be viewed. PTZ cameras can be controlled.
The access rights typically granted to the different user groups are as displayed in Table 4.3.

Table 4.3 Access to the network cameras in the EQUALS lab

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Camera</th>
<th>Access rights by user group</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Admin</td>
</tr>
<tr>
<td>System maintenance</td>
<td>Cam 01-05</td>
<td>Control</td>
</tr>
<tr>
<td>Overview of research activity</td>
<td>Cam 01</td>
<td>View</td>
</tr>
<tr>
<td>Experiment setup</td>
<td>Cam 02-05</td>
<td>No access</td>
</tr>
<tr>
<td>Experiment execution</td>
<td>Cam 02-05</td>
<td>No access</td>
</tr>
</tbody>
</table>

Occasionally control rights may need to be granted to remote users over a short period of time (circa three hours). This is to allow:

- collaborative partners to scrutinise a particular test in detail and at their own pace.
- UoB personnel to provide remote tours of the EQUALS lab when visiting schools, giving presentations etc.

Before being granted this right, remote users must be approved by the person responsible for the scheme. This approval process will make the applicant aware of the privacy issues associated with use of the scheme and confirm their compliance with the acceptable uses of the scheme. To this end, a form will be used. Any personnel who may be active in the lab when remote users have control rights must be made aware of the situation so they can organise their activities accordingly.
4.7.2.4 System operation

All cameras will be in operation 24/7. If the scheme is not in use (i.e. the majority of the time) the cameras will be turned to point at a wall.

A set of presets will be stored to enable Cams 02-05 to be quickly configured to monitor set locations, such as zoomed into the shaking table or pointing backwards at the lab walls. These pre-set positions should be configured in such a way that they monitor only those spaces which are intended to be covered and no unintended areas.

If the scheme is in use but camera feeds need to be restricted (e.g. during commercial tests) then all cameras will have their viewing rights temporarily disabled. The TPM website will display an appropriate message.

When recording video streams, Cams 02-05 will be configured in such a way so that the test occupies the full picture frame with the smallest possible additional coverage of lab. BEELAB’s Quality Assurance procedures for recorded video data will be put into practice.

4.7.2.5 Policing

Before being given control rights, users must be made aware of the purpose for which the scheme has been established. They must also confirm that they will use the equipment only for the purpose for which it has been installed.

A screen will be installed in the lab to give continuous on-site viewing of all 5 cameras to help identify any system misuse. All PTZ requests to the cameras will be logged so system misuse can be traced back to the perpetrator.

If any misuse of the scheme occurs, the perpetrator will be issued with a warning issued by the person responsible for the scheme. Persistent misuse will result in the access rights of the perpetrator being withdrawn.
4.7.2.6 Signage

Signs will be placed so that visitors to the lab are aware that they are entering a zone which is potentially being monitored. The signs will read be clearly visible, at eye level and be of A4 size. Appropriate wording is seen in Fig. 4.13.

![Signage Image]

**Fig. 4.13 Appropriate signage.**

4.7.2.7 Assessment

An annual consultation will be undertaken to evaluate the effectiveness of the scheme taking into account the views of all user groups. More regular monthly consultations will be necessary in the first year after implementation.
4.7.2.8 Example activities

- Recording video for BEELAB tests (i.e. BNC rather than web interface) – local users only
- Research testing on shaking table/reaction walls – local users (control) and remote users (view)
- Time lapse video of reaction wall tests/model construction – local users (control)
- General publicity of research activities – public access
- IDEERS tests in lab – local users (control) and remote users (view)
- Video tracking using vision system– local users only

4.7.3 How Video Recording sets Cache and Archive size

Video recording via java program SonySource provides the following rules:

- Use one image frame per Data Turbine frame.
- Total number of frames = (duration * frames per second) + 250
- Set archive to total number of frames.
- Set cache to total number of frames unless estimated total size > 512Mb. If limit is exceeded
- Reduce cache size to fit under limit. Note that estimated total size is based on guess of 32K per jpeg image frame.
5 Implementing a Telepresence Node: the experience at UOXF.DF

5.1 INTRODUCTION

The telepresence experience at University of Oxford is mainly based on the work done for UK-NEES.

This section is intended to briefly explain the installation and basic configuration of FlexTPS at University of Oxford. FlexTPS is software that allows viewing and controlling live video over the Internet. Its principal advantage is that it is accessed through a Web browser, no additional software is required.

Within the UK-NEES project, after the cameras installation, FlexTPS is used as a telepresence tool, allowing people to follow experiments remotely and see what is happening by means of live video.

Please, notice that this section does not deal with issues such as SSH certificates, site collaboration or advanced FlexTPS configuration/modification. For any other information, refer to the official FlexTPS manual (at the time of writing: https://www.nees.org/research/dl_detail/flextps_2.2_user_guide/).

5.2 INSTALLATION

FlexTPS can be installed on a Linux Red Hat computer by following the instructions from the official FlexTPS Web page.
First, a system update might be required. This can be achieved with the following command:

```
# up2date-nox -u httpd mod_perl perl-Crypt-SSLeay perl-Digest-SHA1 perl-Time-HiRes mod_ssl SDL libjpeg libjpeg-devel perl-XML-SAX perl-XMLNamespaceSupport perl-LDAP
```

This command will update the required packages, after which FlexTPS can be downloaded from the URL: ftp://ftp.nees.org/pub/site-software/flextps/neesgrid-flextps-2.2.0-rhel4.tgz

To download the file, this command can also be used:

```
```

The downloaded TAR file can be decompress by means of:

```
# tar -zxvf neesgrid-flextps-2.2.0-rhel4.tgz
```

There are some RPM packages inside the TAR. To install the RPM packages, it is necessary to use the following commands:

```
# cd neesgrid-flextps-2.2.0-rhel4
# rpm -Uvh *.rpm
```

Proceeding as explained should not give any installation problem. If an unusual dependency problem should happen, the required dependent RPM packet can be installed individually and then proceed with the remaining packages.

After the installation, FlexTPS can be found installed on the directory /opt/flexTPS.

- The Web files are under /opt/flexTPS/portal/www/
- The log files are under /var/log/flexTPS/portal/
5.3 RUNNING FlexTPS

After installation and before configuration, FlexTPS can be started by running the following commands:

```
# /sbin/service flextps_proxies start
# /sbin/service flextps_httpd start
```

To start FlexTPS automatically after rebooting, these commands can be used:

```
# /sbin/chkconfig flextps_proxies on
# /sbin/chkconfig flextps_httpd on
```

Any time the machine starts, FlexTPS will be running.

To check the FlexTPS interface is working, open a Web browser and type the following address:

```
http://localhost/portal
```

A Web page similar to the one of Fig. 5.1 should be shown:
5.4 Configurations

FlexTPS configuration required three parts:

- Cameras configuration.
- Portal configuration.
- Users configuration.

Modifications in some of these files might require restarting the server to have effect.

5.4.1 Cameras configuration

Cameras are configured on the file: `/opt/flexTPS/proxies/etc/proxies.xml`

On this file, the camera IP address, port and camera type are defined. This XML configuration file has a structure as follows:

```xml
<proxies>
  <proxy>
    INFORMATION CAMERA 1
  </proxy>
  <proxy>
    INFORMATION CAMERA 2
  </proxy>
  <proxy>
    INFORMATION CAMERA 3
  </proxy>
  ...
</proxies>
```

To configure this file, the IP address of each camera must be known.

This was configured for each of our cameras by using a Web interface, so every camera has assigned an IP address (read camera instruction manual for further details).
Our camera type, *Sony SNC-RX550P*, requires using a new driver program written at UNIVBRIS. This is necessary for the camera robotic control. The file should be copied under the following directory (it might overwrite the old driver file):

```
/opt/flexTPS/proxies/lib/perl/plugins/Robotic/Sony/SNCRX550P.pm
```

The value “SNCRX550P“ must be used for the “robotic-type“ option, but keeping the value “SNCRZ30N“ for the “source-type” option in the proxies.xml configuration file.

A configuration example for one-camera installation is provided in Fig. 5.2.

```
<proxies>
  <proxy>
    <ip>127.0.0.1</ip>
    <port>2010</port>
    <max-connections>10</max-connections>
    <source-type>Sony::SNCRZ30N</source-type>
    <source-protocol>http</source-protocol>
    <source-ip>10.0.1.2</source-ip>
    <source-fps>25</source-fps>
    <persistent>true</persistent>
    <exit-on-failure>false</exit-on-failure>
    <verbose-log>true</verbose-log>
    <status-log>true</status-log>
    <robotic-type>Sony::SNCRX550P</robotic-type>
  </proxy>
</proxies>
```

**Fig. 5.2 Example file proxies.xml.**

### 5.4.2 Portal configuration

The Website part is configured on the file: `/opt/flexTPS/portal/etc/conf/portal.xml`

The structure of this file is as follows:

```
<portal>
  PORTAL GENERAL INFORMATION
  <proxy>
    CAMERA 1 IDENTIFICATION
  </proxy>

  <group>
    <feed>
```

Implementing a Telepresence Node: the experience at UOXF.DF

The file starts with some general information about the Portal. For instance:

```
<site-name>University Of Oxford</site-name>
<site-ip>165.1.6.77165.1.6.77</site-ip>
<off-site-ip>www.ox.ac.uk</off-site-ip>
<site-admin-email>ignacio.lamata@eng.ox.ac.uk</site-admin-email>
```

After that, every camera that will be used through the FlexTPS Portal must be defined, providing an alias or ID for each one. In the following example two cameras are defined:

```
<proxy>
   <id>camerasony01</id>
   <ip>127.0.0.1</ip>
   <port>2010</port>
</proxy>

<proxy>
   <id>camerasony02</id>
   <ip>127.0.0.1</ip>
   <port>2011</port>
</proxy>
```

It is time to define the groups of the Website. Groups are useful to define different roles or operation restrictions with the cameras. Typically, there will be convenient to have a Group for “Public access” with access to public views cameras and no permission to move or operate them, and a different Group for “Private access“, to see and operate (moving, zooming, etc) all cameras.

Inside a Group feeds, sections and users are defined.
5.4.2.1 Feeds

Feeds define the streams for video watching. Every stream has a camera identified by the “proxy-id” that has been previously assigned on the same file under the `<proxy>` section. This ID should match in both sections.

The “robotic-controls” section indicates the allowed operations for the camera: pan, zoom, etc. Usually, for the public access all those parameters will be set to “false”.

```xml
<feed>
  <id>Oxford Dynamic Laboratory</id>
  <stream>
    <id>Principal experiment camera</id>
    <proxy-id>camerasony01</proxy-id>
    <initial-fps>5</initial-fps>
    <max-fps>25</max-fps>
    <robotic-enabled>true</robotic-enabled>
    <robotic-controls>
      <pan>true</pan>
      <tilt>true</tilt>
      <zoom>true</zoom>
      <iris>true</iris>
      <focus>true</focus>
    </robotic-controls>
    <video-box-size>medium</video-box-size>
    <max-connection-length>0</max-connection-length>
  </stream>
</feed>
```

5.4.2.2 Sections

Sections define the Web options shown over the Web, ie, the available options in the Portal menu for a specific Group. This includes a list of the menu option ID, as the following example:

```xml
<sections>
  <id>local_video</id>
  <id>embedded_local_video</id>
  <id>dvr</id>
</sections>
```

The sections available on FlexTPS are:

- local_video
5.4.2.3 Users

Within the Users section, the users of the group are defined. It is done by declaring a username (the password will be defined in another file).

```xml
<users>
  <id>ignacio</id>
  <id>john</id>
</users>
```

To indicate any user, we can use the asterisk symbol.

```xml
<users>
  <id>*</id>
</users>
```

Following, an example file is provided (Fig. 5.3 and Fig. 5.4). This file configures two groups:

- A private group, for the users “ignacio” and “admin”, with access to the two cameras and with the ability to operate the cameras. The Website will show the menu option “dvr”.
- A public group, for everybody, with access to one of the cameras in a view-only mode. The Website only displays the default menu options.
Implementing a Telepresence Node: the experience at UOXF.DF

Fig. 5.3 Example file portal.xml (part 1).
5.4.3 Users configuration

Users configuration includes two configuration files:

- `/opt/flexTPS/portal/etc/conf/portal.xml`, to define the access to the Portal: users within a group, as seen in the previous section.
- `/opt/flexTPS/portal/etc/conf/user_auth.conf`, to define the passwords of the aforementioned users.

In the `user_auth.conf` file users and passwords are specified as follows:
username1=password1
username2=password2
username3=password3
...

The password must be created by using SHA. This allows passwords to be stored in an “encrypted” form, so people reading the file directly will not be able to read the passwords in clear. This can be easily done in Linux with the `sha1sum` command:

```
# sha1sum
```

Now the desired password should be typed and Control + D should be pressed to finish (it might required to press it twice). The program will print the “encrypted” password that must be copied into the `user_auth.conf` file.

Example:

```
[root@telepres]# sha1sum
hola99800b85d3383e3a2fb45eb7d0066a4879a9dad0
```

So for the password “hola” the SHA-1 hash or encrypted password is:

“99800b85d3383e3a2fb45eb7d0066a4879a9dad0”. Now the following line should be included in the `user_auth.conf` file:

```
my_username=99800b85d3383e3a2fb45eb7d0066a4879a9dad0
```

Providing the username “my_username” with the password “hola”.

NOTE: Alternatively the following command can be used to create a password:

```
# echo -n "my_password" | sha1sum
```

But it is not recommended as the password in clear will be copied into the system’s log.
5.5 OTHER CUSTOMIZATIONS

5.5.1 Website header picture

The picture on the Website header can be modified to slightly customise the Website. Initially, the default FlexTPS installation will show the header of Fig. 5.5:

![Fig. 5.5 Default FlexTPS header.](image)

By modifying the image, a more customised look for UOXF.DF site was achieved, as seen on Fig. 5.6:

![Fig. 5.6 Customized FlexTPS header.](image)

This can be done just by replacing the file `/opt/flexTPS/portal/www/images/logo.jpg` with a local picture. It is recommended to use an image resolution of 120x120 or similar.

5.5.2 Firefox 3 video error

Thanks Donald A. Patterson (at Berkeley University) for providing this fix written by Gemma T. Hentsch (at Rensselaer Polytechnic Institute).
**Incidence:** A “Connection error!” message is shown in the video window when one of the video streams is opened.

Some video problems can be experienced when using some Firefox versions with FlexTPS. This problem has not been detected with early versions (Mozilla v.1) but some problems have appeared using Firefox 3. The video worked fine with Internet Explorer.

**Solution:**

Open the file `/opt/flexTPS/portal/www/javascript/video_box.js`

Search for the function called “streamVideo()”.

Change the following line (value “true”):

```javascript
location = mjpegResourceURL(index) + "" + streams[index].FPS + 
"?status_frame=true&amp;random=“ + Math.random();
```

For this one (value “false”):

```javascript
location = mjpegResourceURL(index) + "" + streams[index].FPS + 
"?status_frame=false&amp;random=“ + Math.random();
```
6 Storage and bandwidth requirements of NEES telepresence tools

6.1 INTRODUCTION

A series of forty tests was conducted at the University of Bristol to empirically quantify the demands of Data Turbine and Remote Data Viewer (RDV). Tests consisted of acquiring N channels of data at frequency F hertz and for duration D seconds using legacy data acquisition software (SIMACQ) adapted to stream data to Data Turbine (on the CEN-NEES-POP server) via the University of Bristol coded SIMACQDRIVER.jar and the US NEES tool DaqToRbnb.jar. This system is described in more detail in Section 4.

The acquisition parameters adopted were in the ranges indicated below:

- Number of data acquisition channels (N): 1 - 64
- Acquisition frequency (F): 10 – 32000Hz
- Acquisition duration (D): 1 – 1000sec

Various results are presented in Fig. 6.1.
Fig. 6.1 Empirical correlation of the storage and bandwidth demands of NEES telepresence tools.

Complementary tests were performed at ELSA to illustrate the data traffic at the level of the Telepresence Server (see Section 3 for more details about the JRC/ELSA installation) for different real-time telepresence applications.

6.2 DATA TURBINE STORAGE DEMAND

Each acquisition run produces an archive directory within the home\RBNB\RBNB_archive folder on the CENNEES-POP server that is identifiable by (source) name (as defined by the source
program SIMACQ). An archive folder contains multiple files and directories. The acquisition data itself is stored in .rbn file format within the underlying RB1, RB2, … directories.

It is found that the size on disk of a Data Turbine archive increases with the number of channels, the acquisition frequency, and the duration. Minimum archive size is found to be around 40kB, which presumably accounts for the structure of the archive and not the acquisition data it holds.

As indicated in Fig. 6.1a, the approximate size of a Data Turbine archive resulting from a typical data acquisition run can be estimated with the following relationship:

\[
\text{Estimated Size } S \text{ (Bytes)} = \begin{cases} 
8FD(1 + N), & \text{if } S > 10^5 \\
10^5, & \text{if } S < 10^5 
\end{cases} 
\]  
(Eq. 1)

The total storage requirement placed upon a Data Turbine can be estimated by accumulating the output of Eq.1 across a laboratory infrastructure’s estimated throughput of tests.

6.3 RDV BANDWIDTH REQUIREMENT

Data from each of the pre-recorded forty tests was streamed from Data Turbine to an offsite pc running RDV over a nominal 2Mbit/sec (home) broadband connection. For each test, RDV was configured to display all available data channels. An RDV playback rate of 1 was employed (i.e. the data was streamed at the rate at which it was recorded). During streaming, a network packet analyser was used to filter out and quantify the communication between source (Data Turbine) and sink (RDV) by isolating the packets to and from the source machine (CEN-NEES-POP server) ip address.

Before presenting the output of the network packet analyser some preliminary remarks on performance are given:

- Multiple streams of the same acquisition run can be associated with different network demands. In particular, an increased demand was intermittently placed upon on the network that is probably associated with background processes. This increased demand was measured on nine occasions (around 18% of the total test count) and was associated
with low acquisition frequencies (F < 24Hz) and medium to low acquisition channel numbers (N = [4 8 16 32]). During these tests, the average number of packets transmitted was increased by (on average) 235% and the average packet size was increased by (on average) 160%. When such a high demand was recorded, the test in question was restreamed and the demand was seen to reduce to more typical levels.

- The NEES software is designed so that when the RDV/Data Turbine network demand exceeds the available bandwidth functionality is maintained. In this event, the rate of data transmission slows and the duration of RDV playback extends beyond the duration of the test (i.e. the playback rate in RDV is automatically reduced). However, no provision is made to inform the user that playback rate has dropped.
- Staccato playback within RDV is occasionally observed even when operating well below the network bandwidth. It appears that such behaviour coincides with the receiving of bad tcp packets.
- During the initial seconds of RDV playback it is common for a short segment of data of length typically less than 1 second to be missing. It seems as though this is a display problem since throughout the period, tcp packets continue to arrive.

It is found that as frequency and number of channels increases, so do both the number of packets and the average size of the packets transferred between the Data Turbine and RDV. However, as seen in Fig. 6.1b, the average size of packet appears to be limited to 1kB. It is not clear however whether this is due to local computer/network settings or the design of the telepresence software.

Fig. 6.1c contrasts the Data Turbine archive size with the amount of data streamed from the Data Turbine to RDV. It is seen that the communication is optimally efficient (i.e. lies closest to a 1:1 line) when the number of data acquisition channels (N) is one. As N increases, the streamed bytes exceed the Data Turbine archive size. It is thought that this is because within the archive, one timestamp is associated with N channels of data whereas when streamed, one timestamp is provided per channel.

Finally, the relationship between estimated Data Turbine archive size (as evaluated using Equation 1) and the average bandwidth consumed during an acquisition run is presented in Fig.
6.1d. The trendline of this figure provides an empirical estimate of the bandwidth demand of RDV and Data Turbine.

\[
\text{Bandwidth demand } B \text{ (Mbit/s)} = 0.0002(S / D)^{0.76} \quad \text{(Eq. 2)}
\]

Note that by reducing the playback rate, the NEES tools remain functional when the bandwidth exceeds the bandwidth available.

### 6.4 RBNB TRAFFIC IN REAL-TIME

In this section, some traffic evaluations are made at the level of the Telepresence Server (the machine running the Data Turbine and the Data Turbine Utilities. Two configurations have been analysed, using DaqToRbnb and AxisSource.

#### 6.4.1 DaqToRbnb

In this configuration, after starting RBNB, the following tasks are made in sequence:

- Start DaqToRbnb on the telepresence Server together with FakeMSDAQ with 64 channels and 3 sampling times (200ms, 20ms & 2ms) running on the Data Collector.
- After few seconds, connect RDV and open 4 X/Y graphs windows.
- Start the real time mode first on 1s time window, later enlarged to 10s, 30s and 2mn.
- Disconnect RDV.
- Stop DAQ.

A packet analyser (Wireshark) is used to analyse the data traffic and three I/O graph statistics are extracted in bytes/s: total traffic, traffic on the data socket (FakeMSDAQ to DaqToRbnb on port 55056) & traffic related to RDV (port 3333, end point RDV machine). The following figures are then obtained: Fig. 6.2 for 200ms sampling rate, Fig. 6.3 for 20ms and Fig. 6.4 for 2ms.
Fig. 6.2 Traffic for 200ms FakeMSDAQ sampling rate.
Fig. 6.3 Traffic for 20ms FakeMSDAQ sampling rate.
Fig. 6.4 Traffic for 2ms FakeMSDAQ sampling rate.
The following remarks can be made:

Concerning RDV:

- The stabilized load (plateau) for the visualization is around 80 to 100kBytes/s and does not change very much with the sampling rate.
- Some load peaks appear when the connection is established and each time more data are requested.
- The extension (that is the increase of duration) of the time windows may be quite traffic demanding, and logically increase with the sampling.

Concerning DaqToRbnb:

- The traffic on the data socket is quite regular as can be seen in the green curves. The values are given in Table 6.1.

<table>
<thead>
<tr>
<th>DAQ sampling time</th>
<th>Bytes/s</th>
<th>Packets/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ms</td>
<td>117888</td>
<td>192</td>
</tr>
<tr>
<td>20ms</td>
<td>58944</td>
<td>96</td>
</tr>
<tr>
<td>200ms</td>
<td>9210</td>
<td>15</td>
</tr>
</tbody>
</table>

- The evolution of the values is highly non linear, as can be seen in Fig. 6.5. This may be explained by the fact the configuration of DaqToRbnb did not change with the sampling rate (the amount of time to group records remained the same and equal to 200ms).
- For a low sampling rate, the traffic due to DaqToRbnb is comparable to the one of RDV but becomes preponderant for the high sampling rate.
Fig. 6.5 FakeMSDAQ traffic on the data socket.

No analysis has been done combining several DaqToRbnb & several RDV but since these applications work almost independently with respect to the data turbine, it can be guessed that the effect on the traffic is additive.

6.4.2 AxisSource

In this configuration, after starting RBNB, the following tasks are made in sequence:

- Start AxisSource on the telepresence Server (together with the Axis Video Server 241QA).
- After few seconds, connect RDV and open I graphs window.
- Start the real time mode.
- Disconnect RDV.
- Stop AxisSource.

A packet analyser (Wireshark) is used to analyse the data traffic and three I/O graph statistics are extracted in bytes/s: total traffic, traffic on the video port (Video Server to AxisSource on
port 80) & traffic related to RDV (port 3333, end point RDV machine). Fig. 6.6 is obtained, displaying the results of 2 runs.

The following remarks can be made:

- The traffic related to AxisSource is quite similar to the one of RDV in terms of bytes/s, it is almost the double in terms of packets/s (not shown in the figure).
- The Video camera is filming a gently changing scene when AxisSource is started and stopped (714kBytes/s).
- When RDV is running, the camera films also partly the RDV window (which is further shake by the operator using the mouse). The traffic may rise up to 900kBytes/s.
- If the camera is filming an almost still scene (as during the first run in the middle of the RDV session, and as it frequently occurs during a PSD test), the traffic may lower to 600kBytes/s.

Again, no analysis has been done combining several AxisSource & several RDV but again, since these applications work almost independently with respect to the data turbine, it can be guessed that the effect on the traffic is additive.
Fig. 6.6 Traffic for AxisSource.
7 Summary

Telepresence is now a reality.

It takes always some time to make it working, due to the adoption of a strategy, the heterogeneity of the products that need to be connected together, the development of a robust tool to extract the data from DAQ and send them to the data turbine.

It takes also some time to demonstrate that the whole telepresence system (working normally or hanging due to a too heavy flux of data) is not interfering with running of the current test and is not impacting its performance. This is crucial since large tests on non-linear structures cannot be done again, and thus should have the highest priority in terms of reliability.

At the time the telepresence tools were first deployed at ELSA, it was part of an integrated package (NEESpop), and it was not easy to build-up applications at the top of it. Fortunately the NEES software was subsequently made modular. However, even if the successive releases of the needed software were always installed, the use of scripts instead of smart interfaces remained. Telepresence tools are used by one or two partners during each major test, with an always increasing reliability. The playback mode is very often used after the tests by the ELSA team in order to have a fast browsing of the signals and then focus on particular events. SERIES is a good opportunity to modernize the access to the tools and gain more participants to future tests.

US NEES developed telepresence tools, together with modification where necessary, have been bought into commission by the three partnering institutions of UK NEES. Live and pre-recorded data acquisition and video signals can now be streamed to remote parties over the internet. Responding to the slow uptake of these tools by the US earthquake engineering community, the usability of the telepresence tools has been significantly improved in order to reduce the
inconvenience to researchers. Now, all user input is through graphical user interfaces and the intuitive menus of a web application. Operation does not require any specialist knowledge of command line driven utilities. The UK NEES telepresence systems have proved performant in a number of individual cases. Commercial clients have used the system to check product performance and monitor test progress during seismic qualification tests. International research colleagues (from both Italy and New Zealand) have participated in concurrent research. Tours of UK NEES infrastructures have been remotely guided by speakers at overseas workshops. Now, the UK NEES telepresence system requires a rigorous series of testing to gauge its limitations and user feedback to gauge its efficacy.

It is expected that the present report will help the interested SERIES partners to set up their own strategy to implement telepresence, alleviate some burdens related to installation and running of the various NEES software and give some valuable hints at practical level.
8 References


