Using a Network Ring Buffer for Science and Outreach

PAUL HUBBARD[1]
JASON HANLEY[2]
TERRY WEYMOUTH[3]
MATTHEW MILLER[4]

[1]San Diego Supercomputer Center
[2]State University of New York at Buffalo
[3]University of Michigan
[4]Creare Inc

We present results from the use of a networked ring buffer in the field of earthquake engineering and collaborative research. The system is used to buffer data, provide an abstraction layer, and act as a central server for live data. It handles and manipulates numeric data, video, audio and text, with a flexible method for handling timestamps and the correlation of disparate data sources. Its addition to the NEES system has significantly increased flexibility, modularity and robustness, with the subsequent authoring of applications that would previously have been impossible.

Categories and Subject Descriptors: H.3.5 [**Information Storage and Retrieval**]:Online Information Systems; H.4.3 [**Information Systems Applications**]: Communications Applications; J.2 [**Physical Sciences and Engineering**]
General Terms: Design, Experimentation
Additional Key Words and Phrases: Ring buffer, earthquake engineering, NEES

This work was supported primarily by the National Science Foundation under Award Numbers CMS-0117853, CMS-0086611, CMS-0086612. Authors' addresses: Paul Hubbard, San Diego Supercomputer Center, CA 92093-0505; Terry Weymouth, School of Information - North, The University of Michigan, MI 48109-2112; Jason Hanley, Structural Engineering and Earthquake Simulation Laboratory, University at Buffalo, NY 14260-4300; Matt Miller, Creare Inc., Hanover NH 03755

1. INTRODUCTION

The Network for Earthquake Engineering and Simulation (NEES) is a large-scale National Science Foundation (NSF) project to enable and encourage advances in earthquake engineering, seismic design and civil engineering through the integration of experimentation, theory, data, and model-based simulation. Following the initial development phase, NEES has transitioned to its operations phase and continues to design innovative tools and infrastructure for remote collaboration, telepresence, distributed experiments, educational outreach and data curation. Additional efforts are focused on user-friendly Grid security, a central data repository, analysis tools, documentation and quality assurance.

This paper concentrates on our use of a network ring buffer to hold and manipulate data and video. Our adoption of this paradigm has led to some novel possibilities for streaming, analysis and collaboration. While some similar commercial tools exist [2], our use of free software enables a broader reach of both users and contributions than would otherwise be possible, as well as adding some unique characteristics and features. The NEES distribution already includes multiple contributions from our experimental sites that enhance and extend its capabilities and usefulness.

2. BACKGROUND

NEES is the George E. Brown Jr. Network for Earthquake Engineering and Simulation. NEES is an NSF Major Research Equipment (MRE) project to provide a widely accessible, state-of-the-art science and engineering infrastructure through experimental testing facilities and the application of advanced computers, networking and software.

The Cyberinfrastructure component of NEES began in 2001 with the 3-year System Integration (SI) effort. This ran from 2001 to September 30, 2004. This part of the project, dubbed NEESGrid, was a collaborative development effort managed by the National Center for Supercomputing Applications (NCSA). Primary contributors were Information Sciences Institute (ISI) at University of Southern California (USC) (Architecture), Argonne National Lab (telepresence, data acquisition, and imaging), NCSA (packaging, administration, and repository) and the University of Michigan (collaboration). The SI was tasked with designing, creating and integrating the technologies required to make NEES possible.

As of October 1 2004, the SI finished their development efforts and transferred the software to NEES Cyberinfrastructure Center (NEESit). NEESit manages the maintenance, operations, and continued enhancement and evolution of the IT infrastructure developed for the NEES community. The NEESit team is led by the San Diego Supercomputer Center (SDSC) at University of California, San Diego (UCSD) and also includes partners at Oregon State University (data ingestion, usability), University of California, Berkeley (Open Sees Simulation), the University of Michigan (collaboration, telepresence), and Mississippi State University (Simulation Portal).

There are fifteen NEES equipment sites [18], encompassing a range of research: Field experimentation, geotechnical centrifuges, large-scale labs, shake tables and a tsunami basin. NEESit is in close collaboration with software development and technical staff at these sites.

All the software programs mentioned in this paper are available via CVS [10] and as part of the NEESit software distribution bundle [17]. We are very interested in talking to others who have similar requirements and are interested in reusing our work.

3. THE DATA TURBINE

DataTurbine®, also called Ring-Based Network Buffer (RBNB), is licensed by Creare [4] and is available free to non-commercial enterprises. It is a server and client library with a Java API that presents a networked interface to a store and forward server (implemented using a circular buffer). It supports the transport of audio, video, numeric, and byte data. Since data at the server can be kept both in memory and on disk, the ring buffer can be used as a persistent intermediate store, optionally reloaded when restarting, and providing TiVo-like instant replay of buffered data concurrent with live data distribution. RBNB is based upon a patented design tailored for streaming high-speed data acquisition, supporting up to thousands of channels and many megabytes per second throughput.

The client interface (the Java API) of RBNB is organized into:

1) Sources that produce data and push it to a server
2) Sinks that fetch data from a server and "consume" it (e.g to view)
3) Plugins that are both a source and sink of data, for example, to support "in-line" transformation of data.

There are also DAV, HTTP and binary-over-TCP/UDP interfaces. All data is time stamped with a 64-bit floating point number at creation or arrival time, and timestamps can be either user-defined or fractional Unix timestamps. Clients can

1) Request a time range of data as a one time request
2) Subscribe to one or more channels, with gapless best-effort delivery
3) Subscribe to one or more channels, where data may be dropped if the client falls behind.

Using DataTurbine allows multiple machines to synchronize data for viewing and playback. This is especially useful for distributed data acquisition systems spread across multiple computers. The ability to select, view and correlate disjoint data sources is very useful but does require well-synchronized clocks on the cooperating computers. We have had good results using commodity GPS clocks [8] with IRIG-B outputs. These cost approximately seven hundred dollars as of early 2005.

RBNB servers can easily be mirrored (push or pull), setup as parent-child (N deep) replicators, and authenticated with host-based or GSI security. Since the server code is pure Java, server machines can be any preferred operating system or hardware platform supported by Java. We regularly develop, test, and deploy on Linux, OS X and Windows without incident.

The RBNB server and distributed clients are closed-source but free for non-commercial use. [5] NEES has negotiated a redistribution license for NEES users and a source escrow contract.

4. ARCHITECTURE FOR USING THE TURBINE IN NEES

The DataTurbine provides a flexible and powerful abstraction for storing and accessing live or historic data, which makes it possible for the developers to integrate Data Turbine into their applications. Currently, NEES uses a modular architecture, where "feeder" programs are command-line Java applications that acquire data from an external source (e.g. video camera, data acquisition (DAQ) system, microphone, etc) and feed it into the turbine. Additional modules display and manipulate data fetched from the turbine. This allows flexible deployment where RBNB serves as a coupling between relatively simple and "single purpose" suppliers of data and users of data, which are presented a logical grouping of these physical data sources. RBNB supports the modular addition of new sources and sinks, and a clear separation of design, coding, and testing. Figure 1 shows an example of this. Note the Tomcat server has Creare-written plugins that support HTTP and DAV access to the RBNB.

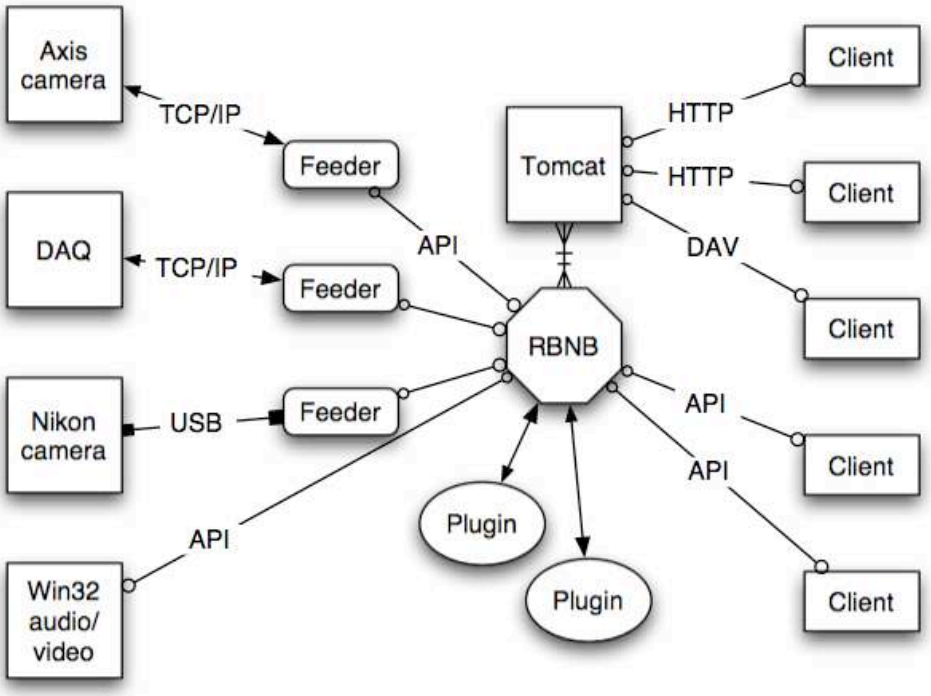

Figure 1: A typical architecture of a site using the turbine

Variations in the configuration of RBNB servers and the NEES-RBNB clients are simple to implement. It is just a matter of making the desired connections at the start of the experiment. For example, in an experiment on the stresses caused by earthquakes, University of California, Los Angeles (UCLA) instrumented a building [20] damaged in the Northridge earthquake and wanted to stream data over the Internet via their satellite link. To avoid saturating the satellite link, parent-child Data Turbines were set up (Figure 2) to support the transparent coupling of the ring buffers. Remote users could view data via the turbine on campus without affecting the turbine onsite.

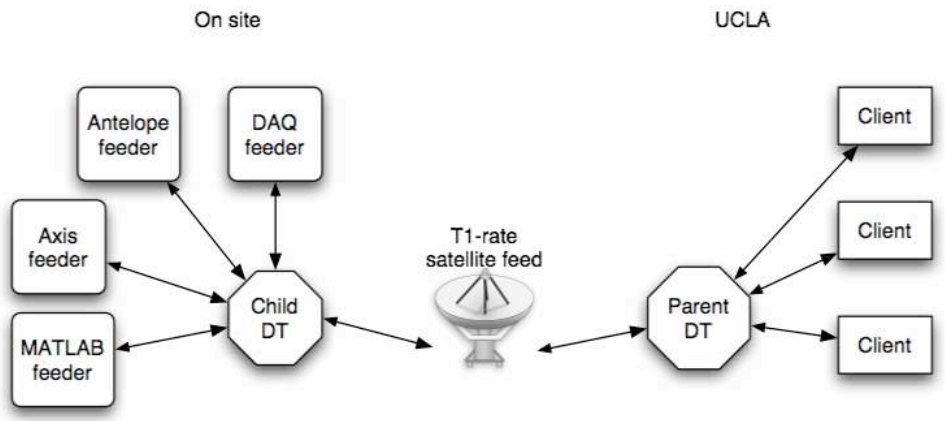

Figure 2: Architecture at an experiment using parent-child linking

In this case, the coupling of the RBNB servers was also a boon to security. Host-based access control was used, so that only local programs could write to the onsite RBNB, and only the parent could read from it. Despite problems with the feeder code, the modular nature ensured that the RBNB servers ran without interruption.

4.1. CLIENT PROGRAMS: CREARE SUPPLIED AND NEES-DEVELOPED

The DataTurbine is distributed with client programs for:

- Administration (set up mirroring, view sources/sinks, kill feeds, etc)
- Data plotting
- Video viewing
- Auto/video capture and playback (live or stored)
- Text chatting
- Plugins for producing graphs from numeric channels
- Video capture
- Data replay at different rates
- Binary data interface for high-speed transmission

While the bundled clients work well, NEES has additional requirements which were met through the development of new clients, for example:

1) Ability to plot X versus Y, where X and Y are data channels in the turbine, and additional requirements for site-specific plotting
2) Units and other metadata. Users need to know what units are defined for each channel, e.g. ‘meters’, ‘Newtons’, etc. This is an example of user-specified metadata
3) Overlaid plots or multiple plots per chart
4) Synchronized display and playback of video and data
5) Allow users to enter and display channel-specific metadata such as units
6) Spreadsheet-style display of numeric data

As a site programmer Jason Hanley developed a Java program that met all of these needs and has other features as well:

- Data viewers for numeric, image, video, and text data. Basic tools for data analysis in the time and frequency domain (FFT) are available
- Extension architecture for development of new data display, visualization, and analysis components, that are integrated within the application
- Uses the jfreechart [15] toolkit for plotting numeric data

- GUI interface for uploading local user data into DataTurbine
- Flexible layout and windowing with independently configurable data viewers
- Variable-rate playback (faster or slower than normal time)
- Synchronized display of data and video from different data acquisition sources.

This program, called Real-time Data Viewer or RDV [9] is now part of the NEES distribution and is shown here viewing data from a shake table test:

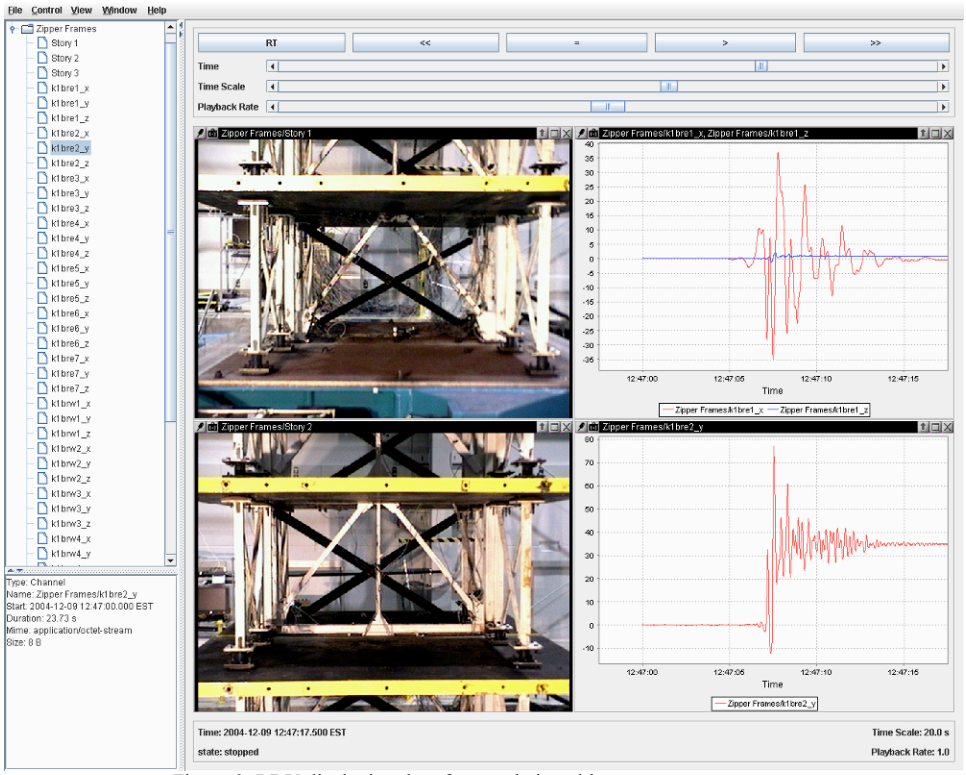

Figure 3: RDV displaying data from a shake table test

4.2. THIN CLIENTS

In addition, NEES is distributing a pair of small Java applets that run in the browser, which plot numeric data and display video feeds. These were written and contributed by Ken Fershweiler at Oregon State University.

While the thin clients tend to be more limited in terms of features and update rate, they are useful for outreach, demonstrations and public web pages because of their limited software prerequisites.

4.3. RUNNING THE CLIENT PROGRAMS

The client programs are distributed as compiled Java code, which presents problems for users that prefer GUI interfaces. Java WebStart (JNLP) files solve this problem and handle versioning and desktop integration. In the interests of education and outreach, NEESit hosts a turbine and applications where the public can experiment and learn. This page is available at http://it.nees.org/support/demos/turbine/.

All of the jar files are signed with a Thawte certificate. This was necessary because some of them need to open TCP connections to machines other than the JNLP host.

There are several methods of extracting data from the turbine that do not require a client program like RDV:

1) HTTP access. As seen at [6], Tomcat provides an HTTP access for data such as images.
2) DAV access. Using the Tomcat interface, users can mount the turbine as a DAV filesystem and read or write data through it. [7]
3) Plugins. Creare supplies a plugin to render numeric data as PNG graphics. The result can then be read via the usual methods. Very useful for quickly creating a web page showing data.

4.4. NOVEL APPLICATIONS

Additional clients perform more complex tasks. For example, a program called DataVideoGather that generates QuickTime movies from video feeds using a numeric channel as a timestamp. This allows a user to generate a movie where every frame is synchronized with the data acquisition system which is very useful for slow tests such as pseudo-dynamic structural experiments. The data file and movie this generates can be loaded into the CHEF [3] data viewer for synchronized playback of video and data.

The staff at the University of Minnesota MAST Lab [16] has written a set of programs to control Olympus digital cameras, with robotic placement during an experiment. A small RBNB plugin program takes these images and adds the physical locations (X, Y, Z and zoom) into the file itself using the EXIF comment block.

We look forward to incorporating more programs as our experience continues.

5. FEEDER PROGRAMS

We have developed several feeders for putting data into the turbine. So far, there are programs for:

- Video feeds, from Axis [1] network cameras, from flexible TelePresence System (flexTPS) [19] and from JMF-supported video capture cards
- Data acquisition (DAQ) systems [11]
- Still image capture from consumer digital cameras [12]
- Video and audio, using JMF-supported devices
- Data files stored on the filesystem
- Plugin to thumbnail video and image sources and decimate data sources for reducing bandwidth requirements.

Similarly, we have developed programs to save data from the turbine to disk and reload it on demand. This allows archiving and persistent storage of data and offline use via a locally installed turbine instance.

Further information on the feeder programs can be found at [13]

5.1. RAPID PROTOTYPING OF FEEDER AND CLIENT PROGRAMS

Feeder programs can be quickly scripted together without code by using the DAV interface. For example, on a site visit to University of California, Berkeley, we were able create an image viewer using the DAV interface to RBNB. We had an onsite requirement to capture and view 1Hz images from an industrial (grayscale) Firewire camera. The images were pushed into the turbine using file writes and viewed using a web browser. Figure 4 shows a diagram of the architecture.

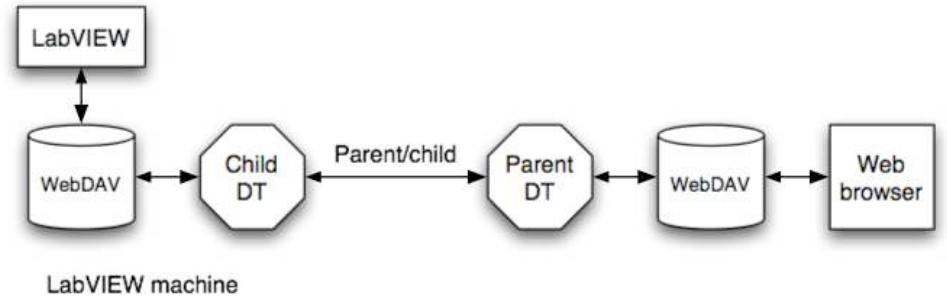

Figure 4: Architecture at UC Berkeley

Note that neither the source application (LabVIEW) nor the client (Internet Explorer) had to be modified for this to work. This provides a very powerful method of creating initial implementations for proof-of-concept and flexible testing.

Here is an image captured using this system:

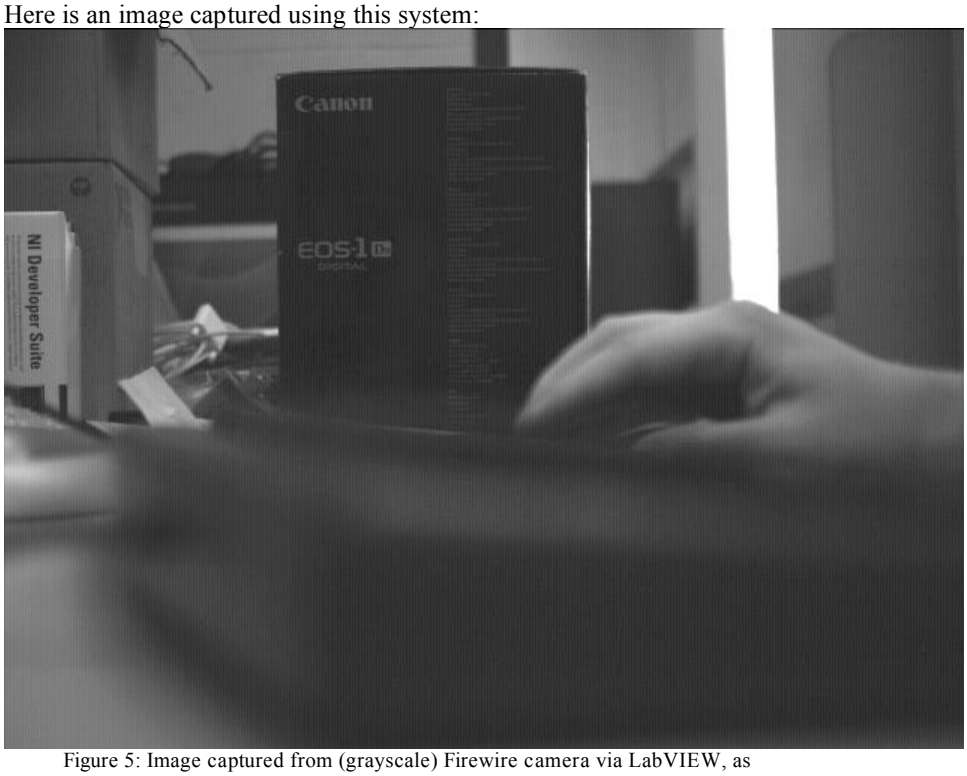

Figure 5: Image captured from (grayscale) Firewire camera via LabVIEW, as displayed in Internet Explorer

6. FUTURE WORK

Now that we have the turbine as part of our software distribution and a useful set of clients and feeders written, we are continuing development to broaden the uses of the system for science. We are working with hardware vendors to add turbine support to their products and talking with the engineers and scientists to learn their requirements.

We are also beginning to integrate audio into the system. Creare recently delivered code to capture and display audio and video via the turbine, and we are very interested to see how this will be applied by our user community.

We will continue to extend the current metadata implementation [14] by defining a standard set of metadata provided by the data sources to enable development of smarter data analysis and visualization tools.

The DataTurbine ships with GSI security as well as host-based access control. We are looking into adding GridAuth [21] support for more flexible authentication.

The Java-centric nature of the DataTurbine is becoming more of an issue as we add support for other systems. We are considering adding a SOAP interface to the API, which would also provide a good location to add GridAuth support. SOAP adds some overheard, but would yield a language-independent interface.

7. CONCLUSION

Many systems have been written to support streaming data, collaboration, video, etc. It is only recently that the continuing advances in computation and networking have enabled the use of a high-level tool like the DataTurbine for a broad user community. An inexpensive desktop computer can easily run a server streaming data at many megabytes per second. This combined with no-cost licenses and NEES open-source software opens up new avenues of exploration.

We find that a networked ring buffer provides several key benefits:

- Identical API for live and stored data
- Clean separation between sources and sinks
- Logical plugin API for data transformation
- Transparent replication and mirroring
- Parent/child/grandchild server hierarchies for unusual network topologies

In this architecture, the development of each new module gains power from all the previously developed modules. The RBNB package comes with a core set of modules and in the NEES project we have developed several general-purpose data acquisition and viewing modules. With these tools available, sites can turn to the development of specific modules for data acquisition, visualization, and analysis. RBNB provides a robust means of creating experiment-specific, networked configurations of these modules for remote data acquisition and collaboration, the creation of distributed experiments, and an easy coupling of instruments, data processing, data displays and archival to data archives.

8. ACKNOWLEDGMENTS

NEESit would like to thank its many contributors and experimental sites. We would especially like to thank Ken Ferschweiler and Tim Holt at Oregon State University for introducing us to the DataTurbine.

9. REFERENCES

[1] Axis Communications Inc., http://www.axis.com/
[2] Boulder Real Time Technologies Inc., *Antelope Real Time System,* http://www.brtt.com/
[3] The Chef Project, http://chefproject.org/
[4] Creare Inc., *RBNB Data Turbine*, http://rbnb.creare.com
[5] Creare Inc., *RBNB ®DataTurbine ®End User License Agreement*, http://outlet.creare.com/rbnb/EULA.html
[6] Creare Inc., *RBNB VideoCam*, http://outlet.creare.com/rbnb/rbnbVideo.html
[7] Creare Inc., *RBNB WebDAV Walkthrough,* http://outlet.creare.com/rbnb/WP/RBNB_WebDAV_WalkThrough.html
[8] CNS Systems Inc, *CNS Clock*, http://www.cnssys.com/cnsclock/
[9] J. P. Hanley, *RDV: A synchronized, streaming data viewer*, http://nees.buffalo.edu/software/RDV/
[10] P. Hubbard, *CVS (Concurrent Version System) at NEESit* http://it.nees.org/documentation/library/software_development/000208.php
[11] P. Hubbard, *Connecting your DAQ to the Data Turbine*, http://it.nees.org/documentation/pdf/TR_2004_29.pdf
[12] P. Hubbard, *An introduction to JCamera*, http://it.nees.org/documentation/pdf/TR-2004-57.pdf
[13] P. Hubbard, *Running the Data Turbine Applications*, http://it.nees.org/documentation/pdf/TR-2004-55.pdf
[14] P. Hubbard, *Channel Units in the Data Turbine*, http://www.sdsc.edu/~hubbard/neesit/dt-units.html
[15] Java FreeChart, http://www.jfree.org/jfreechart/
[16] NEES At Minnesota, http://nees.umn.edu
[17] NEESit, *NEES IT Software downloads*, http://it.nees.org/software/index.php
[18] NEESit, *About NEESit*, http://it.nees.org/about/
[19] C. Stanton, C. Dupre and J. P. Hanley, *flexible Telepresence System*, http://flextps.org/
[20] J. W. Wallace, J. P. Stewart, E. Taciroglu, and D. H. Whang, *Field Vibration Testing and Analytical Studies of a Four-Story RC building*, http://nees.ucla.edu/fourseasons.htm
[21] T. Warnock, A. Lathers, L. Miller and D. Frysinger, *Grid Authority*, http://gridauth.sourceforge.net/