<table>
<thead>
<tr>
<th>Document Type</th>
<th>LIGO-T080035-00-Z</th>
<th>2008/02/20</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network Data Server Protocol Specification</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Josh Smith and others</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Distribution of this draft:*

LIGO Scientific Collaboration
1 Introduction

This document is being prepared.

The Network Data Server (NDS) currently running on the framebuilder computers at the sites has proven to be very useful. A host of software current uses the NDS as a data spigot. However, the current NDS has shortcomings, and is due for an overhaul. As an aid to the development of a new NDS we want to understand the current system and lay out a plan for improving it. This document describes the protocol of the current NDS, lists the software clients that currently use NDS and proposes steps by which NDS could be improved. (This content might be better split into separate documents, one for the current protocol, one for planned improvements).

2 Current NDS protocol

The current Network Data Server (NDS) protocol is documented by Alex Ivanov on the website “Network Data Server Access” at http://www.ligo.caltech.edu/ aivanov/nds_access/.

Most of this section has been taken from the information posted on that page.

The above site also includes links, one of which, http://www.ligo.caltech.edu/ aivanov/nds_access/com_prot11.html, details the protocol used by the Data Acquisition Daemon (DAQD), which Alex says is older and “not entirely correct” for the current version of NDS. However it looks like it might also be helpful since it is more full with examples, error codes and such.

2.1 Communication Protocol

All shorts and ints are sent as ASCII characters, int is 8 Hex digits and a short here is 4 Hex digits. Floats are sent represented as ints. Table 1 shows a list of commands and server responses.

Data is requested using ‘start net-writer’ command variations. In response the server sends the data stream which consists of a header, block of data and reconfiguration blocks. All data variables are binary here, big endian. Table 2 shows the representation of the data stream from the server.
### 2.2 Data Valid Codes

We have 0xbad status set by either DAQ controller or a frame builder when the DCU is not on time (out of sync). There is also status bit 0x1000 that’s added when the data for that block did not check out using the CRC, i.e. checksum mismatch between front-end and frame builder. There is also status bit 0x2000 which gets added when the DCU configuration is different in front-end and frame builder. That is you can change and .ini file an then reload DAQ configuration with Epics button, which reconfigures the front-end, but leaves frame builders with invalid old configuration. They will detect this change and set the status to 0x2000 to indicate this condition. You will have to restart frame builders to pick up new .ini file and set status back to zero for the affected DCU.

DAQ controller only handles out of sync DCU condition and frame builders handle all three error conditions.

### 3 Examples of software that currently use NDS

- dataviewer (C)
- DTT/DMT (C++)
- ligoviewer (TCL)
- ligoDV, mDV, simulink all via Ben’s NDS Matlab client (Matlab)
- dbcommand (C)
- Ben’s c-library (C)
- miniNDS (?)
### Data Structure Representation

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>header (length,</td>
<td>long length; // 16 + following data block’s length</td>
</tr>
<tr>
<td>seconds, gps,</td>
<td>long seconds; // time series data length for what is following next in the data block</td>
</tr>
<tr>
<td>seq_num)</td>
<td>long gps; // GPS time of the first sample in the following data block</td>
</tr>
<tr>
<td></td>
<td>long gps.nanoseconds; // GPS time nanosecond fraction of the first sample in the following data block</td>
</tr>
<tr>
<td></td>
<td>long seq.num; // Sequential number of this block; Starts with zero.</td>
</tr>
<tr>
<td>data_block</td>
<td>time series data grouped by channel (not by time) at the sampling rate requested in 'start net-writer' command. If requested two channels then in this data blocks server sends data for the first channel and then data for the second channel.</td>
</tr>
<tr>
<td>signal_conv</td>
<td>float signal_slope;</td>
</tr>
<tr>
<td></td>
<td>float signal_offset;</td>
</tr>
<tr>
<td></td>
<td>int data_valid; // Data Valid code for the following data</td>
</tr>
<tr>
<td>reconfig_block</td>
<td>header(seconds = -1) followed by reconfiguration data in the following format: signal_conv.t[n], where n = (header length - 16)/16; Reconfiguration blocks can be inserted in the data stream of blocks at any point and there is normally one reconfiguration block in the beginning of data transmission, in front of the very first data blocks.</td>
</tr>
</tbody>
</table>

Table 2: Representation of data stream from server.

### 4 Future NDS improvements

#### 4.1 Stopgap improvements

A stopgap measure to open the possibility for use of software like ligoDV and mDV by a larger portion of the LSC (than the currently small group of commissioners allowed by CDS) would be setting up another NDS at the sites, additional to the ones on the framebuilders, to allow more load (especially from offsite). Also, requiring users to ssh tunnel to that machine would be a significant increase in data security. One question is how to get the data from framebuilder to proxy server?

#### 4.2 Longer term improvements

Very well described by John Zweizig on the February 13, 2008 DASWG telecon. This section/document is certainly best written by John.

### 5 NDS Matlab Client Wishlist

The current NDS Matlab client was developed by Ben Johnson. Source code for Solaris, Linux and Mac as well as a few compiled mex (Matlab executable) files for Windows and a README file are available on Ben’s website at http://idas-jobs.ligo-wa.caltech.edu/ bjohnson/NDS/.
Table 3: NDS communications protocol.

Although this client works very well, there is some development work still to be done, even for the existing NDS system. The following lists some improvements that could be made to the Matlab NDS client.

1. There is currently a bug that stops the client from working on platforms that run 64 bit Solaris and newer 64 bit Matlab. The workaround currently being employed at LHO is to use Matlab2006b instead of 2007b, which is obviously not a permanent solution. Finding and fixing this bug in the source code is a high priority for control room operation.

2. One of the benefits of software such as ligoDV and mDV is that they comprise platform independent m-files. However, since they rely on the NDS Matlab client to get their data, this client is the bottleneck for platform independence. Ben did a really nice job of creating executables that compile on Solaris, Linux and Mac, but because Windows users do not normally have the machinery to compile mex files (cygwin, gcc, etc) he went the route of supplying precompiled mex files for given platforms. Unfortunately, these do not work unless one is lucky enough to have the right matlab and Windows. A small library of precompiled mex files for a few common combinations of say, WinXP SP2 and Matlab2007b would allow Windows users to get in the game.

3. Currently to retrieve a channel list, one uses the command NDS_GetChannels, with usage:

```
Table 3: NDS communications protocol.

Although this client works very well, there is some development work still to be done, even for the existing NDS system. The following lists some improvements that could be made to the Matlab NDS client.

1. There is currently a bug that stops the client from working on platforms that run 64 bit Solaris and newer 64 bit Matlab. The workaround currently being employed at LHO is to use Matlab2006b instead of 2007b, which is obviously not a permanent solution. Finding and fixing this bug in the source code is a high priority for control room operation.

2. One of the benefits of software such as ligoDV and mDV is that they comprise platform independent m-files. However, since they rely on the NDS Matlab client to get their data, this client is the bottleneck for platform independence. Ben did a really nice job of creating executables that compile on Solaris, Linux and Mac, but because Windows users do not normally have the machinery to compile mex files (cygwin, gcc, etc) he went the route of supplying precompiled mex files for given platforms. Unfortunately, these do not work unless one is lucky enough to have the right matlab and Windows. A small library of precompiled mex files for a few common combinations of say, WinXP SP2 and Matlab2007b would allow Windows users to get in the game.

3. Currently to retrieve a channel list, one uses the command NDS_GetChannels, with usage:
sc = NDS_GetChannels(host)
Must be run before any other function.
Returns NDS server’s channel list. The server
is specified by the host parameter.

This command takes as arguments only the host server and port. It does not allow one to
query a channel list for a specific time. Since channel names occasionally change, especially
on the boundary between S5 and enhanced Ligo, it would be useful to get a channel list for
a specific time.

4. The NDS has a Data Request Command called ”gps”, which is described as Get current
GPS time. It would be useful to have an NDS client command that allowed one to get the
current GPS time from the server (especially if this time were the latest time of any data
stored there). This would be especially useful for retrieving data in near real time - when the
latest time from the server updates, get the new data.

6 Summary