LoDN in a Nutshell

Logistical Distribution Network

Jean-Patrick Gelas

August 9, 2004

This quick reference should be of interest to people who might want to install and maintain a LoDN *server.*

1 Introduction to LoDN

LoDN (pronounce Low-Down) is a file directory based on the IBP protocol and exNode file description, both designed for Logistical Networking by the LoCI research group. Users can access their directory to manage their files and sub directories of files. They can also access other users published data. If a user has not explicitly published a file, other users will not be able to access it.

LoDN provides a reliable storage service thanks to the automated duplication of data and periodic renewal of time limited storage allocations by a process called "Warming" (defined in Section 3). The Warmer helps to maintain reliability and fault tolerance by actively managing files, for instance augmenting the number of available copies when it falls below specified limits. LoDN gives fast access to the data because at download, the individual blocks of a fragmented, geographically distributed file can be downloaded in parallel from different depots. LoDN also increases performance by allowing the user to position data close to possible download sites. You can upload your file to a depot in your area and then create replicas on any of the 331 depots located around the world. At download time, the file is automatically retrieved from those depots providing the highest throughput. Because of all these features, LoDN provides a simple distributed file system, well suited for a Content Distribution Network (CDN).

When a user uploads a file into the Logistical Network through the LoDN interface, LoDN generates an exNode file. This exNode file is stored on the LoDN server and allows the Warmer (Sect. 3) to maintain availability and integrity of the data.

From an end-user point of view, LoDN is a web-based graphical user interface that allows the user to upload, download and publish content in his/her directory. When a file is *published* it becomes available for download by anyone. Even guest users who do not have a LoDN account.

On the client side, the software requirements are no more than, a standard web browser (with frame support and *JavaScript* enabled), a modern Java Runtime Environment (> 1.4.x), and a network connection (modem, xDSL/cable, T3,...). The Java client applications (*LoDNPublisher* to upload files and *LoDNClient* to download files) rely on Java *Web Start* technology. This technology is based on the *Java Network Launching Protocol* (JNLP), which provides a way for an application to run from a codebase accessible over the network. Java Web Start provides an environment similar to what would happen if URLs were supported in the classpath.

All communications, between the client and the following items are made through the TCP protocol.

- client to LoDN server,
- client to IBP depots,
- and client to L-Bone directory,

On the LoDN server side, the LoRS tools are required by the current implementation of the Warmer and by the *augment* and *view* commands available through the web interface. Currently, we use *Apache* as web server to serve the `html` pages and to call the CGI scripts (written in Perl). The server is also used by *LoDNPublisher* and *LoDNClient*. Because these applications access the user's local file system, the class files (compressed and archived in `.jar` files) are signed with our certificate provided by a Certificate Authority (CA).

To conclude this introduction, I will say that the main advantages of the LoDN approach over peer-to-peer networks in the context of file sharing or content distribution network are :

1. Users do not have to share their own local disk space because data are stored on publicly available storage resources (IBP depots).

2. Users do not have to stay connected to keep their published (or unpublished) data available to other users.

Actually, a user uses their own computer only to upload, download and manage data remotely through the LoDN web interface. The user does not have to store the file content (data) or the storage meta-data (such as file name, location, size,...) on their local system.

2 Basics for managing a LoDN server

2.1 Set the LoDN administrator e-mail address

In order to receive user requests for the creation of new accounts, you must specify your e-mail address in the LoDN configuration file. This file is called `lodn.cfg` and is located in the application directory (see Section 4.2 for more details about this directory).

To specify an e-mail edit the `lodn.cfg` text file, search for the `ADMIN_EMAIL` field and substitute the current e-mail address with yours.

2.2 Add or ban a LoDN user

When a new user applies through the create account web page, you should receive an e-mail titled *Disabled LoDN Account Created* (see the previous section to set your e-mail address as LoDN administrator). The body of the mail contains the following fields: first name, last name, e-mail address and login identifier of the new applicant. For example:

```
Jean-Patrick Gelas, gelas@cs.utk.edu, \
  jpgelas has created a disabled account
```

To allow a user to access his/her LoDN account you must allow him/her effectively. For that you must use the Perl script `auth.pl`. Without an argument, displays the list of all users who have applied for an account. Currently only the login identifier and account status (yes or no) is displayed. An account status *yes* means that the user can access their account, while *no* means either his or her account has not been enabled or the user is banned. To enable an account, use `auth.pl` with the login identifier as the first argument and the word *yes* as second argument. To ban a user replace yes with no. For example use, `./auth.pl monica yes` to enable the monica's LoDN account, and `./auth.pl chandler no` to disable the chandler's LoDN account.

2.3 Current limitations

Be warned that there is currently no way to retrieve a LoDN user's forgotten password. A future upgrade of the web page interface should provide a "Lost my password" button. This button will display a page asks the user a question (chosen during the account creation) and invites the user to enter a new password.

3 Automagically maintaining data availability in a Logistical Network: The Warmer

The storage service provided by Logistical Networking's underlying infrastructure, called IBP, is known as a "best effort" storage service.

This means that IBP does not provide guarantees of sustained data accessibility-stored data may become temporarily or permanently unavailable due to machine or network outages, or the space used for "soft" storage allocations may be reclaimed by the local system. IBP's primitiveness is by design and allows it to scale globally. Strong properties like reliable data accessibility, extended storage duration, and large file size (¿2MB), are achieved by aggregating individual IBP allocations into a distributed "network file."

Data is stored on IBP depots in allocations of predefined, fixed size, called capabilities, for a fixed period of time, according to a dynamically-negotiated storage "lease". The existing IBP interface allows a client or an application to renew this lease with the refresh operation. The augment command allows data to be copied from one depot to another. This operation allows the user to store replicas of file content at multiple locations. Thanks to these mechanisms we can provide robustness and persistence.

When a file is uploaded into the Logistical Network, it is divided into blocks of data. These blocks are spread and stored over one or more IBP depots. The exNode file is an XML description of this partitioning and geographical distribution of file content. The exNode allows us to associate the many individual IBP allocations holding blocks of file content into one "network file".

In order to assure data integrity and availability, the LoDN server must run a process known as the Warmer over all the exNode files for which it is in charge.

The term integrity is used here in the context of availability of individual blocks of data. If one block of file content is missing, we can say that the file integrity is compromised. However it is not definitive (final), as one or more copies may still be available on depots which did not respond at the time of the file integrity check.

In general, the Warmer must:

- walk through the exNode file directories of each user (called content directories, see Section 4.1),
- process the status of each block in each exNode,
- make some decisions (*refresh*, *trim* or *augment* data) according to local policy (max number of copies, percentage of hard and soft storage, supplemental lease time).

The Warming operations must be applied on a regular schedule. On the current LoDN server hosted by *promise*, we choose to warm the complete content directories twice a day thanks to the *crontab* utility.

The Warmer is currently a Perl script using the LoRS tools. The commands used are `lors_ls` to display the list of the blocks and their status (1=available, -1=unavailable), `lors_trim` to remove the supposed unavailable blocks, `lors_augment` to add one or more copies of the blocks, and `lors_refresh` to renew the lease of the blocks stored on IBP depots.

4 Directories and files of interest

This section lists and describes the purpose of the files of interest in LoDN. The paths of the files below are those used on the *promise.sinrg.cs.utk.edu* system. It does not mean that the files must be located in or follow this exact directory path and directory name.

4.1 The content directory

LoDN stores and maintains a collection of exNode files. The exNode files are stored in a very standard directory tree of the file system. We use the following organization. The root directory in LoDN is by default called `content`. Below this content directory you should find a set of directories each named with the user login identifier. Each directory belongs to a unique user. A user directory can contain none, one or many exNode files. Directories may also contain several levels of sub-directories holding exNode files. The management and directory tree structure below the user directory is left to the user's discretion. The read and write Unix rights must be the same as the web server used to run LoDN.

TIPS: The best option is to use a dedicated hard drive partition on which you mount the LoDN content directory.

4.2 The application directory

The LoDN application directory contains the following files. The `html` files, the images and logos in the `images` directory, the documentation in the `doc` directory and the CGI scripts in the `cgi-bin` directory. The application directory also contains the Java archive files (`.jar` extension). For more information about these files see Section 6 of this document.

In the `cgi-bin` directory, among the executable scripts written in Perl and run by the web server you should find two files of importance, i.e which must be backed up on a regular basis (using the `crontab` utility for example).

- The `userinfo` file contains a database table which holds all the information relative to the LoDN users. This information is basically that given during the subscription process, thanks to the create account web page. The `userinfo` file is a binary file, nevertheless only a hash of the password is stored inside. To read and modify this file you must use the `auth.pl` Perl script utility introduced in Section 2.

- The `publishlist` file is a plain text file which lists the published exNode files. The entries are sorted in ascending order, starting with the login identifier, followed by the path to access the published exNode, terminated by the name of exNode file itself. In other words this is a sorted file with a complete path to published exNode files. The `publishlist` file content might look like this:

```
/gelas/Photos/jpmoto.jpg.xnd
/linuxisos/Debian/debian-30r1-i386-binary-1.iso.xnd
/linuxisos/Fedora/FC2-i386-disc1.iso.xnd
/linuxisos/FreeBSD/5.1-RELEASE-i386-disc1.iso.xnd
/linuxisos/Gentoo/install-x86-minimal-2004.0.iso.xnd
/linuxisos/Mandrake/Mandrake92-cd1-inst.i586.iso.xnd
/mbeck/mudmosque.jpg.xnd
/parr/RvB-Season-1/RvB_Episode00_LoRes.mov.xnd
```

4.3 The LoRS tools

The parts of LoDN written in Java are now able to interact directly with the different components of logistical networking (IBP depots and the L-Bone). However, the *Warmer* (introduced Section 3) is a Perl script using some of the commands available in the LoRS tools suite. In addition, the current web interface of LoDN allows a user to *augment* his/her data and view the list of the allocations. These operations are actually wrapped in the CGI scripts which transform the resulting output into `html` format.

For a full introduction to and documentation of the LoRS tools, browse the `http://loci.cs.utk.edu/lors` web pages.

5 How to install a LoDN server

You have an old computer not used by anybody , a 24/7 Internet connection open and you feel ready to host a LoDN server? Ok! This section is for you.

5.1 Configuration

All you need is a GNU/Linux box[1] allowed to run a web server (e.g Apache) and a Perl interpreter. A Java Runtime Environment is required on the server side only if you wish to modify and recompile the Java sources of the LoDN project.

The best choice is to create a *lodn* account and run the web server with the same rights. For that, in your `httpd.conf` file you must set the `User` and `Group` fields to `lodn`.

```
User lodn
Group lodn
```

You should also add the following *Directory* entry (or something equivalent if you don't use Apache) in the `httpd.conf` file to allow the web server to run CGI scripts located in a given directory (here `/home/lodn/public_html/cgi-bin`).

[1]Any GNU/Linux distribution should fit the LoDN requirements.

```
<Directory /home/lodn/public_html/cgi-bin>
     AllowOverride FileInfo AuthConfig Limit
     Options +ExecCGI
     SetHandler cgi-script
     <Limit GET POST OPTIONS PROPFIND>
         Order allow,deny
         Allow from all
     </Limit>
</Directory>
```

Next you must install the LoRS *tools*. They are not included in the LoDN archive but are freely available through the LoCI web site (`http://loci.cs.utk.edu/lors`, section Software).

For LoDN itself, there is only one configuration file called `lodn.cfg`. Comments inside should be enough to help to customized it at your convenience. Below is the configuration file currently used on *promise*.

```
# -----------------------
# LoDN configuration file
# -----------------------

#
# L-Bone servers list
# syntax: LBONE lb1.dom1.edu:port1 lb2.dom2.edu:port2
#
LBONE vertex.cs.utk.edu:6767 acre.sinrg.cs.utk.edu:6767\
  galapagos.cs.utk.edu:6767

#
# Directory out of which CGI scripts are available
#
CGIBIN_DIR /lodn/cgi-bin

#
# Protocol and path to retrieve .jar files
#
CODEBASE http://promise.sinrg.cs.utk.edu/lodn

#
# Directory which holds users' exNode files.
#
# Do NOT add a slash at the end of the directory path.
#
CONTENT_DIR /export/home/lodn/content

#
# Directory which olds the LoRS tools
#
LORS_TOOLS_BIN /usr/local/lors-current/bin

#
# system host name
#
HOST promise.sinrg.cs.utk.edu

#
# DOMAIN your.domain.name
#
DOMAIN .sinrg.cs.utk.edu

#
# Adminstrator's e-mail
#
ADMIN_EMAIL gelas@cs.utk.edu
```

The Warmer should be called twice a day. To do that you can use the *crontab* utility and then add the following entry, using the `crontab -e` command.

```
# Call warmer.pl at 6:00am and 6:00pm every day.
0 6,18 * * * /home/lodn/warmer.pl
```

That's all!

6 LoDN internals

This section should be of interest to future LoDN contributors and developers. The first subsection explains how to retrieve the LoDN code sources from the CVS repository. The second subsection gives a useful overview of the code sources organization, and the two last subsections give a brief explanation about the underlying mechanisms used to upload data in the Logistical Network and retrieve them efficiently.

6.1 LoDN repository

The LoDN project is available on the CVS repository of the UT CS department. To download the latest version of LoDN you must first set your environment variables like this (assuming you use the shell BASH):

```
export CVS_RSH=ssh
export CVSROOT=:ext:YOURLOGIN@ \
  enterprise.cs.utk.edu:/cvs/homes
```

Next you can download (checkout) the current LoDN release in your local file system using the following command:

```
  cvs co lodn
```

If you are not very familiar with CVS, Stephen Soltesz wrote a very helpful tutorial available here `http://www.cs.utk.edu/~soltesz/tutorial/cvs/lors_tutorial_cvs.html`

Think to always do a `cvs update` before you start to work on the source(s) code or documentation, and a `cvs diff` before you `commit` your modifications to the repository.

6.2 Organization of the LoDN sources

The LoDN project sources are distributed in different subdirectories. The package name includes the domain name to package each component, to insure the uniqueness of package names.

The `exnode` directory holds the classes which allow the application to read and write an exNode file. The exNode file follows an XML structure (the namespace definition is available at `http://loci.cs.utk.edu/exnode`). The `exnode` directory contains also all the end-to-end services like *XOR obfuscation, AES encryption, compression, checksum, etc. . . .*

The `ibp` directory contains classes which allow the application to communicate directly with an IBP depot through the IBP protocol.

The `lbone` directory contains a unique class used to send requests to an L-Bone server. A request contains mainly the number of depots, and the duration and location of storage required by the user. The only method available called *getDepots* takes the previous arguments, contacts the L-Bone, and returns the L-Bone's response in a list of depots fitting the requirements as closely as possible.

```
lodn
|-- exnode
|   `-- edu
|       `-- utk
|           `-- cs
|               `-- loci
|                   `-- exnode
|-- ibp
|   `-- edu
|       `-- utk
|           `-- cs
|               `-- loci
|                   `-- ibp
|-- lbone
|   `-- edu
|       `-- utk
|           `-- cs
|               `-- loci
|                   `-- lbone
|-- lodnclient
|   `-- edu
|       `-- utk
|           `-- cs
|               `-- loci
|                   `-- lodnclient
|-- lodnhelp
|-- lodnpublisher
|   `-- edu
|       `-- utk
|           `-- cs
|               `-- loci
|                   `-- lodnpublisher
|-- tools
`-- web
    |-- cgi-bin
    |-- doc
    `-- images
```

The `lodnclient` directory holds the graphical user interface of the application which allows a user to download a file from the logistical network to his/her local file system.

The `lodnhelp` directory contains the embedded documentation in the application itself (LoDNClient and LoDNPublisher). The documentation structure is written to fit the *javahelp* engine requirements.

The `lodnpublisher` directory holds the graphical user interface of the tool which allows a user to upload a file from his/her local file system to the logistical network.

The `tools` directory is a toolbox which contains many useful scripts to:

- enable/disable a LoDN user account (`auth.pl`),
- compile the whole project (`compile.sh`),
- build the java archives (`buildjar.sh`),
- sign the java archives with the LoCI certificate, issued by the Thawtee CA (`signjar.sh`),
- and install the required file on the web server (`install.sh`, note that this script does not overwrite the configuration file (`lodn.cfg`), the `userinfo` file and `publishlist` file).

For more information about these files read the comments inside the source code (very self-explanatory).

The `web` directory contains some `html` web pages. Actually there are few pages in html because the LoDN web interface is mostly generated dynamically by the CGI scripts (located in the sub-directory `web/cgi-bin`). The on-line documentation is in the `web/doc` directory and figures are in the `web/images` sub-directory.

6.3 LoDN publisher

This section describes roughly how the java LoDN Publisher application works.

The LoDN Publisher is launched when the user click the button *Launch Java Upload Client* available on the LoDN web page. When the parameters (*e.g.* location, connection type, extra services,. . .) and the file to upload have been selected the user click the *Upload* button.

This event call the *actionPerformed()* method of the `DownloadAction.java` class file. The first operation is to build a list of IBP depots fitting the requirements (location, availability,. . .). Next, the content of the file is uploaded in the logistical network (*exnode.write(...)*).

This operation of uploading is multi-threaded. A `JobQueue` is filled of `WriteJob` object. Several `WriteThread` are instantiated (the number is function of the connection type) and consume the `WriteJob`s available in the `JobQueue`.

The `WriteThread` remove a `WriteJob` from the queue and execute it. The `WriteJob` read a portion of the file and store it in a buffer (a bytes array). If an extra service has been selected, it should be applied now. A depot is taken from the depots list. A space allocation is made on the selected depot. The mapping information about this buffer is built and the buffer is wrote to the selected depot. Then the job is done (figure 1 illustrate this process).

Finally, when all the job are done, the header of the exNode file is built, the mapppings is appended to it and sent, through the `http` protocol to the LoDN server.

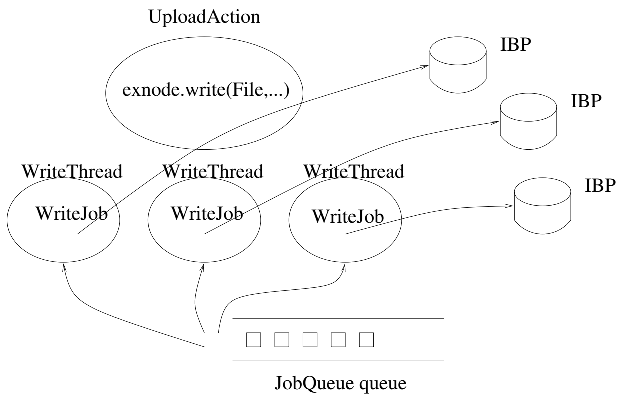

Figure 1: Queue of buffer wrote in parallel by different thread.

For more details read the following source files: `lodnpublisher/.../UploadAction.java` and `exnode/.../[Exnode|WriteJob|WriteThread|JobQueue].java`

6.4 LoDN client

This section describe roughly how the java LoDN client application works.

The LoDN client is launched when a user click the name of a file in his/her web browser. Before to display the GUI the corresponding exNode file is downloaded from the LoDN server to the user system and parsed. The parsing operation is used to find the mappings location. In function of the connection type chosen, the number of threads used to read the different mapping in parallel change (e.g a xDSL connection will use 3 threads). When the user click on the *Download* button the method *read(ouputFile,...)* is called. This method :

- Fill a queue of *ReadJob* object. A ReadJob object contains the references to a portion of the file to read (i.e one mapping) and the associate method to read the mapping from the IBP depot where the data are located.
- Spawn and start the requested number of *ReadThread* threads (the number is function of the connection type). These threads consume the queue of ReadJob. Each *ReadThread* thread remove a ReadJob and execute it. Once executed, a remove another ReadJob from the queue, and so on until the queue become empty. This process of jobs consumption is illustrated figure 2.

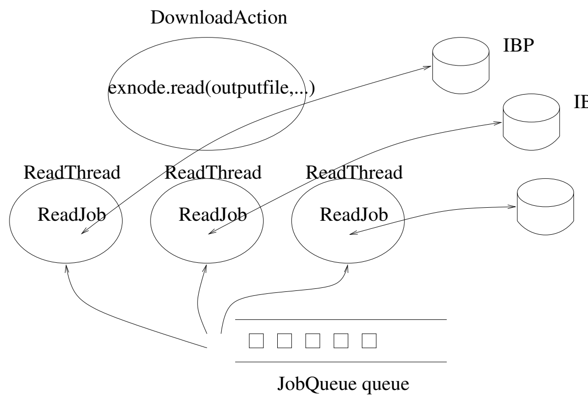

Figure 2: Queue of mappings read in parallel by different thread.

For more details read the following source files: `lodnclient/.../DownloadAction.java` and `exnode/.../[Exnode|ReadJob|ReadThread|JobQueue].java`

Acknowledgement

I would like to thank Jenée Mitchell for her great and in depth review of this document.

Links of interest

1. A very helpful tutorial about CVS. `http://www.cs.utk.edu/~soltesz/tutorial/cvs/lors_tutorial_cvs.html`
2. The introduction to LoDN web page. `http://promise.sinrg.cs.utk.edu/lodn/doc/Intro.htm`
   A French version is also available. `http://promise.sinrg.cs.utk.edu/lodn/doc/Intro-fr.htm`
3. The LoRS tools web page. `http://loci.cs.utk.edu/lors`
4. Introduction to the Java WebStart architecture. `http://java.sun.com/products/javawebstart/architecture.html`

Side notes

This page contains an *addendum* specific to the installation of LoDN on *promise*.

The LoDN project is currently hosted on the `promise.sinrg.cs.utk.edu` system under the user account *ibpvo* (for the password, please ask an entitled LoCI member).

The `auth.pl` script is available in the home directory of the *ibpvo* UNIX user account.
`/export/home/ibpvo/`*`auth.pl`*

The configuration file used by LoDN, `lodn.cfg`, is located in the following directory:
`/opt/local/www/html/lodn`

The content directory is located in `/export/home/exlodn/content`. The */export/home* directory is actually a mounting point of the `/dev/md1` hard drive partition.

The application directory is located in `/opt/local/www/html/lodn`. In this directory you should find the sensitive files *publishlist* and *userinfo*:

- `/usr/local/www/html/lodn/cgi-bin/`*`publishlist`*
- `/usr/local/www/html/lodn/cgi-bin/`*`userinfo`*

Both are backed up every day in the `/export/home/ibpvo/gelas/backup` directory.

The Apache configuration file on *promise* is located in `/etc/httpd/conf/`*`httpd.conf`*.
To compile and install LoDN, I call the scripts in the following order:

```
$ tools/compile.sh
$ tools/buildjar.sh
$ tools/signjar.sh xxpasswdxx
$ tools/install.sh
```

Below the current entries in the *crontab* for LoDN.

```
# LoDN Warmer (new version by Jp (from july 7th))
# (experimented on jp's account since june 22nd)
0 6,18 * * * /export/home/ibpvo/gelas/warmer-jp.pl > /tmp/warmer-jp.log

# Save every night the publish list and account informations of LoDN (Jp)
0 1 * * * /export/home/ibpvo/gelas/backup-lodn.sh

# Check the ISO images availability every day (Jp)
0 14 * * * /export/home/ibpvo/gelas/checkcontent.sh
```

The `checkcontent.sh` script send an e-mail in case of modifications of the *publishlist* file (you must change the e-mail address with yours inside the script to receive the notification.)