

Web Library: Data Movement Fall 2006 Report

Dmitriy Shtokman
Cornell University

Table of Content

1.	The Semi-Automated System	3
1.1	Results of Downloads using the Semi-Automated System.....	4
1.2.1	System Resource Shortage.....	5
1.2.2	Filezilla Instability.....	5
1.3	Internet Archive Access Problem.....	6
1.4	Non-Responsive Nodes.....	6
1.5.1	DP Crawl Status.....	8
1.5.2	DJ Crawl Status.....	9
1.6	DJ Crawl Data Duplication.....	9
1.7	Transfer Rate.....	10
1.8	Routing Anomaly.....	10
1.9	Conclusion (The Semi-Automated System).....	11
2.	The Automation System.....	11
2.1	Proposed Workflow Steps for the Automated System.....	12
2.2	The Automation System Description.....	13
2.3	Conclusion (the Automation System).....	14
3.	Acknowledgements.....	15
4.	References.....	15
	Appendix A: Glossary.....	16
	Appendix B: Traceroute to an Internet Archive node that went.....	17
	through the Internet2 Connection	
	Appendix C: Traceroute to an Internet Archive node that went	18
	through the National Lambda Rail	

Abstract

The purpose of the Data Movement and Tracking Group is to facilitate data transfer from the Internet Archive to Cornell University. The Semi-Automated System developed during the Winter of 2005-2006 and Spring of 2006 was used during the Fall of 2006 to download ARC and DAT files to Scidata1. Since the long-term goal is to switch completely from the Semi-Automated System to the Automation System, the work was done during Fall 2006 to develop the first version of the Automation System. This report describes the data transfer process during the Fall of 2006 and the steps completed towards the automation of the download system.

1. The Semi-Automated System

Throughout the course of the semester the Semi-Automated System that relies upon PHP scripts and multiple Filezilla instances was utilized to transfer data to Scidata1. The structure of the system was thoroughly described in the Fall 2006 final report of the Data Movement Group. Here are the steps necessary to set up downloads:

1. Generate a list of the nodes that contain data for the specified crawl using *get_nodes_for_crawl.php*.
2. Generate queue files for the specified crawl using *get_folders_crawl_node_use_xml_from_file.php*.
3. Transfer the generated queue files to Scidata1.
4. Set up multiple Filezilla instances and specify download options.
5. Upload queue files and process them.

This semester it was decided that it could be beneficial to preserve date and time of the downloaded files. To do so, prior to processing a queue file, go to Edit → Settings... → File transfer settings → Preserve date/time of downloaded files:

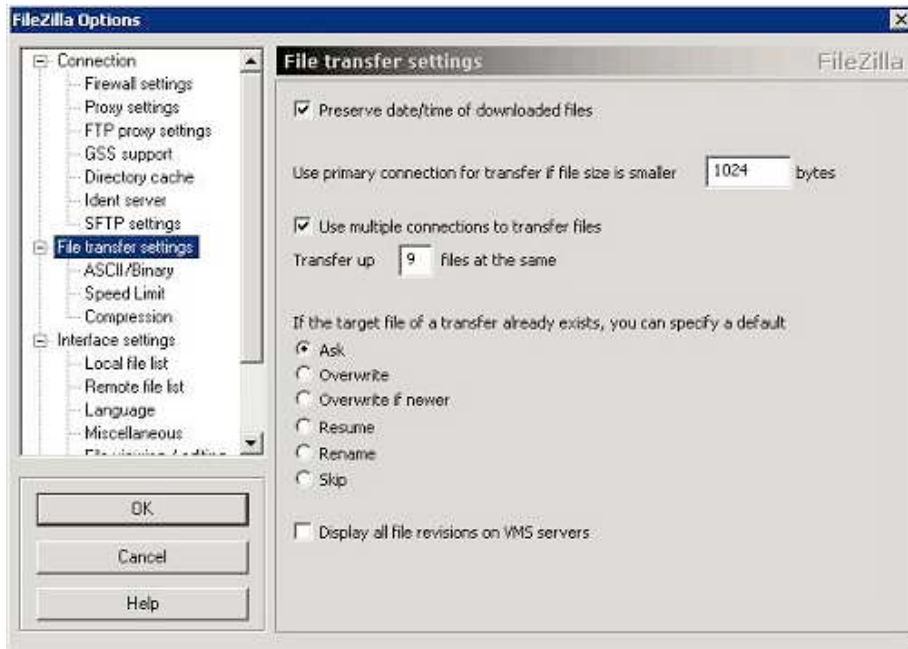


Figure 1.

1.1 Results of Downloads using the Semi-Automated System

The Semi-Automated System was used to download data for the DJ and DP crawls during the Fall 2006 semester. At the beginning of the semester, the DJ crawl was considered complete. It had been processed during the summer of 2006, and about 7.5 TB of data was downloaded then. However, during mid-semester it was found out that certain nodes that contained DJ crawl data had not been processed because of the problem of non-responsive nodes (please see section 1.4). As a result, it was necessary to pause downloading DP crawl data and return to the DJ crawl. Once all the remaining files were downloaded, it became possible to continue downloading DP crawl files.

There are approximately 8.42 TB ARC and 0.55 TB DAT DJ crawl files. The total size of the DJ crawl files is thus 8.97 TB. Since about 7.5 TB of them were downloaded during the Summer of 2006, about 1.47 TB of DJ crawl data was downloaded during the Fall 2006 semester.

The total size of the DP crawl is currently unknown (please see section 1.5.1). 8.46 TB of ARC files and 0.47 TB of DAT files were downloaded over the course of the semester. The total amount of DP crawl data downloaded during the Fall 2006 semester is 8.93 TB.

The total amount of data downloaded over the course of the semester is thus 8.93 TB + 1.47 TB = 10.4 TB. Downloading 10.4 TB over 14 weeks of the semester (no downloads were performed during the week of October 15 – October 23, please see section 1.3) gives an average of 108 GB/day. This is significantly less than the target goal of 250 GB/day, however, several data transfer problems arose during the semester. These problems are discussed in the next sections.

1.2.1 System Resource Shortage

During the Spring 2006 semester it was possible to run nine instances of Filezilla simultaneously. However, during the Fall 2006 semester Scidata1 was extensively utilized by another Web Laboratory team. As a result, the system did not have enough resources to handle nine instances of Filezilla. Multiple attempts were made to have nine instances run at the same time, but they all failed. The system froze or ran very slowly, making it impossible to manage the download process.

1.2.2 Filezilla Instability

A large number of Filezilla instances running simultaneously caused the system load to increase rapidly. When the system load was high, active Filezilla instances often crashed with or without an exception soon after being launched. Sometimes the following exception was generated:

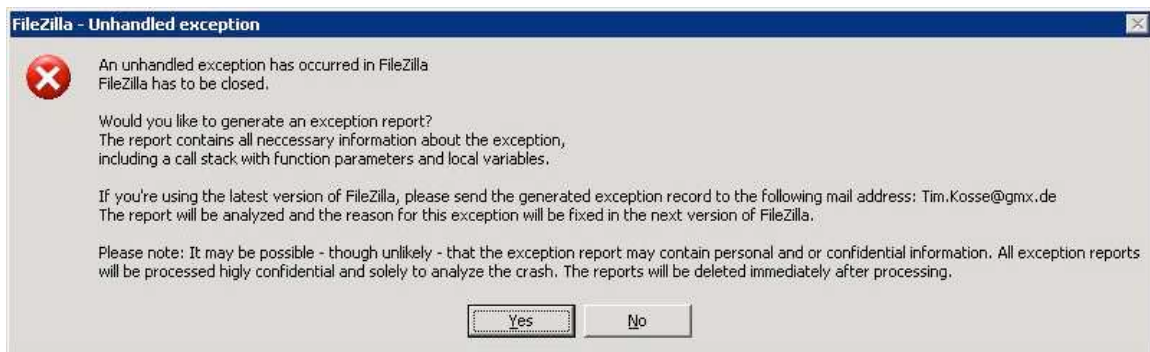


Figure 2.

The error that caused Filezilla to crash is unknown, because it was never possible to generate an exception report. Clicking ‘Yes’ on the unhandled exception message box resulted in the following message:



Figure 3.

As a result of frequent system resource shortage and Filezilla instability, the number of Filezilla instances running simultaneously had to be reduced to five, four, and sometimes even three, decreasing throughput.

1.3 Internet Archive Access Problem

On October 15, 2006, all the Filezilla instances that were launched failed to connect to the Internet Archive nodes. The status fields of all the files in the queues were changed to “transfer error.” Closing and re-opening Filezilla and re-uploading the queue files did not help, and the problem persisted, making any downloads impossible. The problem was not resolved until October 23, 2006, and no downloads were performed between October 15 and October 23. While the suspected reason for no connectivity was the Internet Archive servers’ failure, it turned out that traffic was locked down on Scidata1 via the IPsec policy. The subnets for the inaccessible nodes were added to the policy, and the connection was restored. Here is the list of the Internet Archive subnets that are currently added to the IPsec policy on Scidata1:

208.70.26.0
208.70.30.0
208.70.31.0
208.70.28.0

1.4 Non-Responsive Nodes

Early in the semester inconsistencies in the size of queue files obtained by different PHP script executions for the same crawl were noticed. Often nodes did not respond to script requests or responded incorrectly, which resulted in incomplete (most of the times empty) queue files for the corresponding node being generated. In a correct way, an

empty queue file is generated only if there are no files for the specified crawl at the specified node. Here is how such a file looks like:

```
<Filezilla>  
  <TransferQueue />  
</Filezilla>
```

This issue was first noticed when the remainder of the DJ crawl data was being processed. It was observed that some nodes corresponding to empty queue files actually do contain DJ crawl data. Also, it was observed that many nodes that were completely non-responsive actually did exist and contain DJ crawl data. Thus, the two main problems were nodes that did not respond to script requests at all and nodes that gave incomplete responses, resulting in incomplete queue files being generated. It was necessary to deal with these two problems to preserve integrity of data transfers.

Multiple script executions were performed and their results were compared. The purpose of these multiple executions was to ensure that for each node there were at least two sets of script-generated queue files that could be compared to verify node completeness. When processing nodes for the DJ crawl, a node was only considered complete if at least two sets of queue files generated by different script executions had the same size. Scripts were run multiple times until the desired queue files were obtained. Eventually all the queue files for the DJ crawl were verified for completeness, except for ia300604.us.archive.org and ia300602.us.archive.org, which were persistently non-responsive, so that only one set of queue files was generated for these two crawls.

The problem persisted when the DP crawl was being processed. Script executions often resulted in queue file sets being generated that were substantially different from their counterparts generated on another day. Multiple script executions were chosen again as a timely but effective and relatively reliable solution. Unlike the generated DJ queue file sets, the generated DP queue file sets seemed not to contain any partially complete files – queue files were either complete, i.e. with all the information about the data available, or empty. Unfortunately it was not possible to check every one of the generated DP queue files for completeness, but several files were randomly checked, and all of them were complete.

Non-responsive nodes were one of the main reasons that average daily throughput for the Fall 2006 semester was less than the target 250 GB/day – because it took a long time to perform multiple downloads and go through the results to compare them. Furthermore, it took a substantial amount of time to compile a list of all the nodes containing data for the DJ crawl (and a similar list for the DP crawl). This list was also generated by a PHP script and also was often incomplete. It was necessary to compare several generated lists and create a list of nodes from them. When such a list was compiled from several lists, it was also necessary go through the resulting list, checking whether the nodes followed the logical sequence, i.e. ia310230, ia310231, ia310232. If there were reasonable grounds to believe that relevant crawl data may be contained on one of the nodes that was not in the list, such node was assessed via a web browser and its content was checked.

1.5.1 DP Crawl Status

It was not possible to verify all the DP queue files for completeness due to the time constraints. Nevertheless, it was still necessary to perform multiple script executions, because many nodes that contained DP data were non-responsive and PHP scripts repeatedly generated empty queues for them. Several days before the end of the semester finally it was possible to generate queue files for all the remaining eight non-responsive nodes. However, on December 15, a set of nodes with DP crawl data that had not been processed before was discovered.

According to the estimates made during the Spring 2006 semester, there were 93,565 ARC and 93,565 DAT files total for the DP crawl. As of December 10, 2006, 91,804 ARC and 91,804 DAT DP files were downloaded, which constituted about 98% of the total number of files. The remaining 2% were expected to be downloaded by the end of the Fall 2006 semester. However, with several dozens additional nodes with DP crawl data discovered one day before the end of the semester, it was not possible to complete the transfer of all the DP crawl files. It is unclear whether that DP crawl data was just recently moved to these nodes or whether those nodes were non-responsive for the entire semester.

1.5.2 DJ Crawl Status

All the data available for the DJ crawl that had not been processed during the summer was downloaded to Scidata1 during the Fall 2006 semester. A very small set of DAT (and possibly ARC) files was duplicated. The next section explains the duplication problem in details. Downloaded ARC files for the DJ crawl were archived.

1.6 DJ Crawl Data Duplication

One of the important issues that came up during the Fall 2006 semester was data duplication in the downloaded DJ crawl files. During the download process a slight inconsistency in the number of DAT and ARC files downloaded was noticed. It was observed that the number of DAT files was slightly greater than the number of ARC files. To determine the reason causing this inconsistency, the number of files in corresponding ARC and DAT folders was. In most cases, that number was the same – for example, for the node ia300402 there were 21 files in the ARC folder and 21 files in the DAT folder. However, for some nodes the number of DAT files was greater than the number of ARC files. For example, for the node ia300934 there were 112 files in the DAT folder and 89 files in the ARC folder. The reason causing the inconsistency is the following.

Data transfer for the DJ crawl started with downloading DAT files first. Then ARC files were downloaded along with any DAT files that had not been downloaded before. In the meantime, the Internet Archive moved some DJ files to different nodes. So, for example, *DJ_alex3.20020208165803.dat* was originally located on the ia300934 node. Since DAT files were to be processed first, it was downloaded to the corresponding ia300934 folder on Scidata1. Later on, the Internet Archive moved this file to the ia310237 node. Thus, when this node was being processed later this semester, the DAT file was downloaded again. Because of non-responsiveness issue, all the nodes that contain DJ data were processed more than once during the semester, so that all the data available would be downloaded to Scidata1. As a result, currently, there are two copies of this DAT file on Scidata1 – one in the ia310237.us.archive.org folder and another one in the ia300934.us.archive.org folder. However, there is only one copy of *DJ_alex3.20020208165803.arc* on Scidata1, because this file was moved from the ia300934 node to the ia310237 node by the time ARC files were being downloaded from

the ia300934 node. The same thing happened with some other files. That's why there are slightly more DJ DAT files than DJ ARC files on Scidata1.

All the nodes that contain DJ crawl data were processed and the files were downloaded. The total number of DAT files downloaded is 93,631, and the total number of ARC files downloaded is 93,485. So, there are approximately 146 duplicate DAT files. The word “approximately” is used because it’s possible that some ARC files were also duplicated. However, since DAT files were downloaded first, it’s reasonable to assume that the number of duplicated ARC files (if any) is less than, say, 1/3 of the number of duplicated DAT files. Thus, the total approximate number of duplicate files is less than or equal to 200.

$$200/(93,631+93,485)*(100\%) = 0.10689.$$

So, less than 0.107% of the total files for DJ crawls are duplicated. It was unclear what to do about these files, because it would have required significant manual or programming work to detect and remove all the duplicate files.

1.7 Transfer Rate

Average transfer rate varied significantly during the course of the semester. The lowest rate observed was about 15 KB/s, and the highest rate observed was about 600 KB/s. However, such extremes were rare. Usually the transfer rate varied between 100 and 300 KB/s. Towards the end of the semester, the transfer rate was noticeable higher than at the beginning of the semester.

1.8 Routing Anomaly

On October 18, 2006, it was noticed that the packets sent to the Internet Archive nodes went over the Internet2 Connection rather than the National Lambda Rail. This change didn’t seem to influence the data movement process. No significant increases or decreases in transfer rate were observed. The traceroutes to the Internet Archive through the Internet2 Connection and the National Lambda Rail are attached in the Appendices B and C. The regular traffic through the National Lambda Rail resumed on October 26, 2006. An interesting observation was made – even though traceroutes to individual

Internet Archive nodes went over the Internet2 Connection, traceroutes to www.archive.org always went through the National Lambda Rail.

1.9 Conclusion (The Semi-Automated System)

Once again, the Semi- Automated System has proved that it is only a temporary solution. Various problems that arose throughout the course of the semester decreased throughput and required much time to resolve. It is possible to improve the Semi-Automated System by emulating the scripts in C#. NodesApplication, which was written in C#, usually generated a more complete list of relevant nodes than the PHP script. Furthermore, SmartFTP, which seems to be a more stable application than Filezilla, could substitute the latter in the Semi- Automated System, since in SmartFTP Version 2.0 Build 999 (October 6, 2006) an option to import/export XML queue files was added. Whether to invest into such major changes next semester remains to be seen and mostly depends on the persistence and severity of the problems with the Semi- Automated System in its current form as well as the progress in the development of the Automation System.

2. The Automation System

The purpose of the Automation System is to enable the data transfer process to function without significant human intervention. The development of the Automation System began in the Spring 2006 semester and provided good basis for this semester's work. Here is the schematic description of the system:

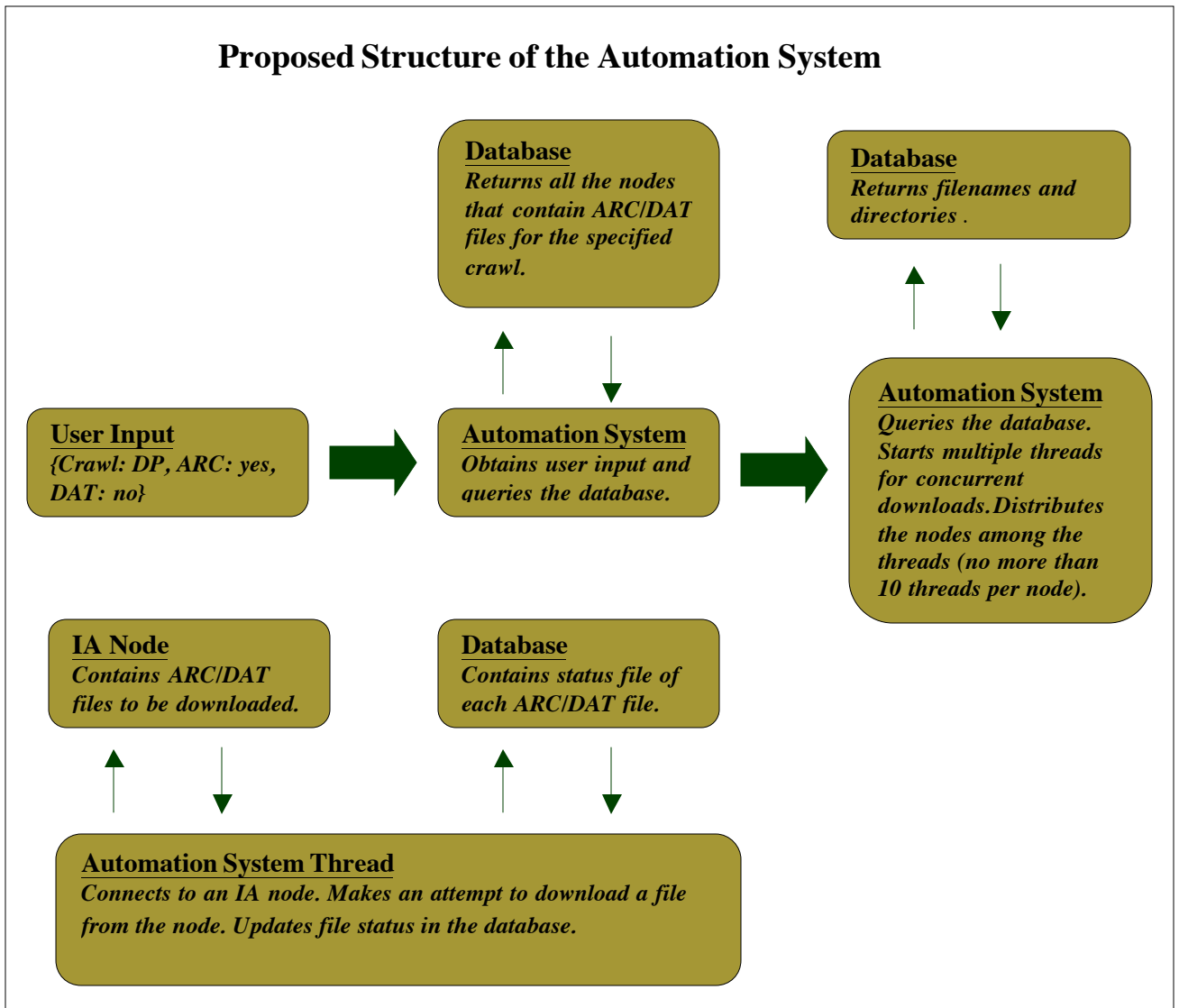


Figure 4.

2.1 Proposed Workflow Steps for the Automation System

The steps in the workflow are as follows:

- The Automation System obtains the crawl and types of files (ARC/DAT) to be downloaded from the user.
- The Automation System queries the database, which is updated by the Data Tracking System, and gets all the nodes that contain ARC/DAT files for the specified crawl.

- The Automation System queries the database and gets names and directories of the relevant files that are located on each node in the list.
- The Automation Systems starts multiple threads for concurrent downloads and distributes the nodes among the threads, so that no more than ten distinct threads access the same node simultaneously.
- Each thread connects to an IA node, makes an attempt to download a file from the node, and updates file status in the database.

2.2 The Automation System Description

The goal is to make the Automation System a full substitute for the Semi-Automated System. During the Fall 2006 the first working version of the Automation System was developed. It utilizes SmartFTP Library to provide file transfer functionality. Currently it does not have all the functionality it takes to substitute the Semi-Automated System. A lot of values that are to be obtained from the database and the user are currently-hard coded. For the test purposes, “DP” was hard-coded as a name of the crawl to process. The method that is to query the database and obtain all the nodes for the crawl currently returns five hard-coded node names. These are real nodes that contain DP crawl data. The methods that are to query the database and obtain all the names and directories on these nodes have 12 hard-coded filenames and 12 hard-coded directories for each node, the total of sixty filenames. All the hard-coded values are valid. This approach was taken to check whether the direction chosen was right or whether the proposed schema (please see Figure 4) was not the one to follow.

The multithreaded application, which is the core of the Automation System, successfully accessed the Internet Archive nodes and downloaded 50 DP crawl DAT files to the CSUGLAB machine with the average transfer rate of 125 KB/s per thread. The current version of the Automation System is also capable of downloading ARC files, but for test purposes DAT files were chosen. The Automation System follows the structure described in Figure 4, but lacks certain functionality, such as database access and interactivity. Also, it is necessary to investigate certain access issues. Sometimes the Automation System was not able to access a node or retrieve a node’s content. As a result, 10 or 11 out of 60 test files were not downloaded.

The Automation System code was uploaded to the CVS under the DmtTransferComponent project. The structure of the current version of the Automated System is outlined below:

1. IA nodes with DP crawl data are retrieved (five nodes, hard-coded values).
2. For each of these nodes a new thread is launched.
3. Each of these five threads attempts to connect to the corresponding node to check connectivity.
4. If connectivity is successful, the filenames of those files stored at each node are retrieved (twelve filenames for each node, hard-coded values).
5. Threads are launched to download the files located at each node. No more than nine threads access each node. Each thread is corresponding to its own filename.
6. In each thread a directory where thread's file is located is retrieved (a directory for each file, hard-coded values).
7. An attempt to access the directory and download the file is made.

2.3 Conclusion (the Automation System)

These are the next steps in the development of the Automation System, in order of importance:

- Resolve access/retrieval failures.
- Add Resume capability and Logging.
- Integrate the Automation System with the Data Tracking System
- Perform extensive testing on the Automation System and determine the optimal number of threads/nodes empirically.
- Design a GUI or a web interface to add interactivity.

Once these steps are completed, the Automation System will be a robust and efficient solution to the data transfer problem. It has a great potential to outperform the Semi-Automated System in terms of throughput, usability, and reliability. During the next semester our efforts should be directed towards improvement and expansion of the Automation System.

3. Acknowledgements

I would like to thank my advisor Professor William Arms and my group leader Ruth Mitchell for giving me the opportunity to work on this project. I am very grateful for their guidance, encouragement, and constant support.

4. References

1. Sosa, C. B., Jain, P., Shtokman, D., [Web Library: Data Movement Spring 2006 Report](#). May 2006
2. Jain, P., Shtokman, D., Tiwari, H., [Data Movement Research Project](#). December 2005
3. Internet Archive <http://www.archive.org/>
4. Filezilla Homepage <http://filezilla.sourceforge.net/>

Appendix A

Glossary

- Crawl:** A snapshot of the web.
- Data Tracking System:** A system that populates the database with crawl information.
- Internet Archive:** An organization that maintains an archive of the Web.
- Node:** A server at the Internet Archive with crawl data. We are primarily interested in so-called SOLO nodes, because each of them contains data that is not replicated at other Internet Archive nodes.
- NodesApplication:** A C# program that creates a collection of XML files for the specified crawl. Each XML file in the collection is named after a node storing data for the specified crawl and contains directories located at that node.
- SmartFTP:** An FTP client for Windows. SmartFTP Library is used to provide file transfer functionality to the Automation System.

Appendix B

Traceroute to an Internet Archive node that went through the Internet2 Connection

October 26, 2006, 12:09 AM.

Z:\>tracert ia310038.us.archive.org

Tracing route to ia310038.us.archive.org [208.70.30.22]
over a maximum of 30 hops:

1	<1 ms	<1 ms	<1 ms	cerberus-csug.cs.cornell.edu [132.236.227.1]
2	<1 ms	<1 ms	<1 ms	rhodes1-msfc-vl339.net.cornell.edu [128.84.154.1]
3	<1 ms	<1 ms	<1 ms	core2-msfc-vl8.net.cornell.edu [132.236.222.174]
4	1 ms	<1 ms	<1 ms	cornellnet2-dmz2.net.cornell.edu [132.236.222.2]
5	2 ms	2 ms	2 ms	199.109.9.25
6	5 ms	5 ms	5 ms	buf-7600-syr-7600.nysernet.net [199.109.7.5]
7	17 ms	17 ms	17 ms	abilene-chin-buf-7600.nysernet.net [199.109.2.2]
8	21 ms	21 ms	21 ms	iplsng-chinng.abilene.ucaid.edu [198.32.8.77]
9	31 ms	38 ms	52 ms	kscyng-iplsng.abilene.ucaid.edu [198.32.8.81]
10	41 ms	41 ms	43 ms	dnvrng-kscyng.abilene.ucaid.edu [198.32.8.13]
11	66 ms	66 ms	96 ms	snvang-dnvrng.abilene.ucaid.edu [198.32.8.1]
12	73 ms	74 ms	73 ms	losang-snvang.abilene.ucaid.edu [198.32.8.94]
13	77 ms	77 ms	77 ms	hpr-lax-gsr1--abilene-LA-10ge.cenic.net [137.164.25.2]
14	85 ms	85 ms	85 ms	svl-hpr--lax-hpr-10ge.cenic.net [137.164.25.13]
15	86 ms	86 ms	86 ms	internet-archive--svl-hpr-ge.cenic.net [137.164.27.174]
16	*	*	*	Request timed out.
17	87 ms	87 ms	87 ms	ia310038.us.archive.org [208.70.30.22]

Trace complete.

Appendix C

Traceroute to an Internet Archive node that went through the National Lambda Rail

October 26, 2006, 8:18 PM.

Z:\>tracert ia310038.us.archive.org

Tracing route to ia310038.us.archive.org [208.70.30.22]
over a maximum of 30 hops:

1	<1 ms	<1 ms	<1 ms	cerberus-csug.cs.cornell.edu [132.236.227.1]
2	<1 ms	<1 ms	<1 ms	rhodes1-msfc-vl339.net.cornell.edu [128.84.154.1]
3	<1 ms	<1 ms	<1 ms	core2-msfc-vl8.net.cornell.edu [132.236.222.174]
4	8 ms	8 ms	8 ms	nyc1-msfc-dmz2.net.cornell.edu [132.236.222.4]
5	12 ms	10 ms	13 ms	newy-nlr-vl4000.net.cornell.edu [192.35.82.129]
6	14 ms	14 ms	14 ms	wash-newy-98.layer3.nlr.net [216.24.186.23]
7	35 ms	35 ms	28 ms	atla-wash-64.layer3.nlr.net [216.24.186.20]
8	52 ms	51 ms	51 ms	hous-atla-70.layer3.nlr.net [216.24.186.8]
9	82 ms	90 ms	82 ms	losa-hous-87.layer3.nlr.net [216.24.186.30]
10	82 ms	81 ms	82 ms	hpr-lax-hpr--nlr-packetnet.cenic.net [137.164.26.130]
11	89 ms	89 ms	89 ms	svl-hpr--lax-hpr-10ge.cenic.net [137.164.25.13]
12	90 ms	90 ms	90 ms	internet-archive--svl-hpr-ge.cenic.net [137.164.27.174]
13	*	*	*	Request timed out.
14	92 ms	91 ms	91 ms	ia310038.us.archive.org [208.70.30.22]

Trace complete.