distCVS: A Distributed Peer-to-Peer Versioning File Storage System

Andrew Logan

May 13, 2005

Advised by:
Professor Robert Signorile
Professor Elizabeth Borowsky

Thanks to:

My family, for always encouraging me to follow my interests.
My thesis advisors, for first suggesting this problem and then devoting a year of their time to guiding me towards a solution.
All of the other faculty and staff members at BC who donated their time, energy and resources to this project.
And last but not least, my fellow Senior CS Thesis authors, without whom the Unix lab would not have been nearly as much fun at 2 am.

Contents

1 Abstract

The current layout of resources on the internet suffers from the fact that there is generally a single point of failure for data access. Peer-to-peer applications promise to change this by distributing resources, and therefore network routes and load, across the network itself. The problem is that the development and testing of such a system is often a hard and tedious chore, especially since the technology is still rapidly evolving. In this paper, I present distCVS, a peer-to-peer system for the storage of versioned data. distCVS is unique because it is an application built by using the Bamboo peer-to-peer networking framework to pass messages for an unmodified version of the CVS source control system. The development of this type of peer-to-peer system allows the designer to not only rapidly assemble a working system, it also allows him or her to take advantage of any future improvements that may be made to either Bamboo or CVS.

2 Introduction

The basic layout of the internet has changed very little since its inception in the late 1960's. In essence, the resources of the internet are laid out in a star topology. Client computers spread across the internet access resources contained on a central server. The advantages of this type of system, and part of the reason that it has been so long-lived, is that it is very easy to find files and resources in this type of layout. Every client knows that the resources they want to access are stored on one central server, and since the server's location in the network changes infrequently, clients know how to find it. In fact, there is generally a delay ranging from several hours to as long as one or two days when a server moves its network address until the clients are able to find it again.

There are drawbacks to laying out a network in a star topology, though, and these drawbacks stem from the fact that data eventually has to pass through the central server. The network does not scale well, and there is a single point of failure. Since once computer has to process all of the requests coming through the network, either it or its link to the rest of the world can easily become overwhelmed by the amount of data it has to process. If this server goes off of the network because of this overload, some other failure, or even routine maintenance, then none of the clients can access the resources stored on it until it is brought back online.

Despite these problems, high-traffic sites do exist on the internet. They

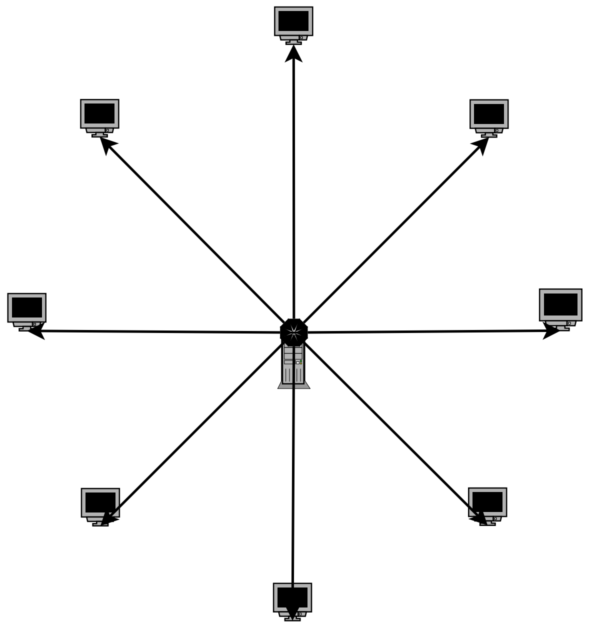

Figure 1: *An 8 node network laid out in a star topology.*

tend to work around these problems by having their resources served by networks of machines that are capable of handling the enormous traffic loads that they experience. A more elegant solution is required, however, since this method does not address the root causes of the scalability problems.

Over the last few years, much work has been put in to researching and developing peer-to-peer networks,[6, 3, 4, 8, 2, 7] which suddenly became popular in 1999 with the release and success of the Napster file sharing program. Peer-to-peer systems promise truly decentralized networks, but there are still many problems that have not yet been solved.

The simplest form of a peer-to-peer network would connect every computer on a network to every other one on the network. Although this scheme does provide complete connectivity, it is not practical for several reasons. First, there is a limit to how many network connections a computer can have active at one time, which places a relatively small upper bound on the maximum size of the network.[1] This topology was attempted with the first version of the Gnutella peer-to-peer network, and the designers of that system soon discovered that the information that needs to be shared to index the data on this network imposed an enormous amount of networking overhead, which severely limited the amount of bandwidth that could be used for sharing data.

The popularity of Napster was partly due to the fact that it solved the indexing problem by using a central server to keep track of the resources available on the network.[2] Although this is a clever solution to the indexing problem, the addition of a central server makes this system not truly peer-to-peer. Therefore, this network still has some of the scalability and reliability problems that star networks do. This napster topology is better than a star topology because the resources have been moved to the computers on the network, relieving much of the strain on the central server. However, nodes will not be able to locate anything on the network if the indexing server goes

[1] Valid TCP/IP port numbers are in the range 1-65535. Assuming that all ports could be used for a peer-to-peer network and that there were no conflicts in assigning port numbers, the network would be limited to 65535 computers at any given time. Although this number is more than sufficient for some applications, it is several orders of magnitude lower than estimated 43 million US households with internet access.[9] Clearly, something else has to be done to create a global peer-to-peer system.

[2] See Figure 3

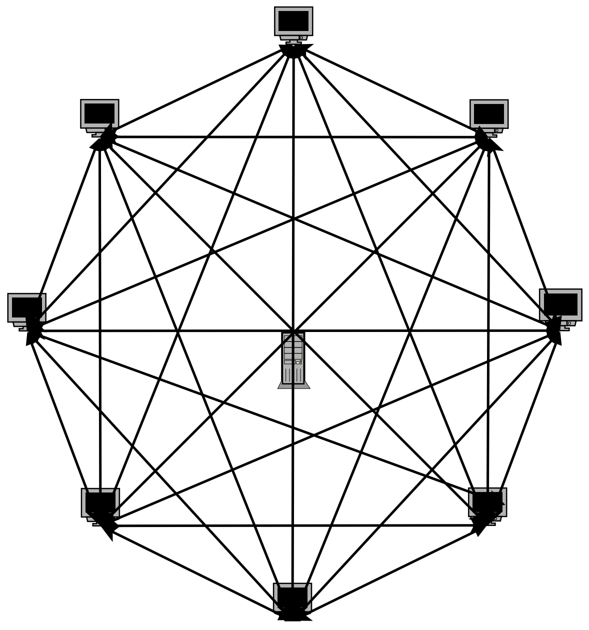

Figure 2: *An 8 peer network laid out in a simplistic peer-to-peer topology.*

offline, and it may still be possible to overwhelm it with location requests.

The most recent innovation in peer-to-peer networks is the use of distributed hash tables (DHT's). DHT's organize computers and resources on a network in such a way that an index server is not needed. In the Bamboo[5] implementation of this idea, each node assigns itself a random id number based upon the port number and ip address it has when it joins the Bamboo network.[3] Each piece of data is also assigned an id number in the same range as the ones assigned to the nodes, and then the numerically closest node in the network to the id number assigned to a specific piece of data becomes its

[3]See Figure 4

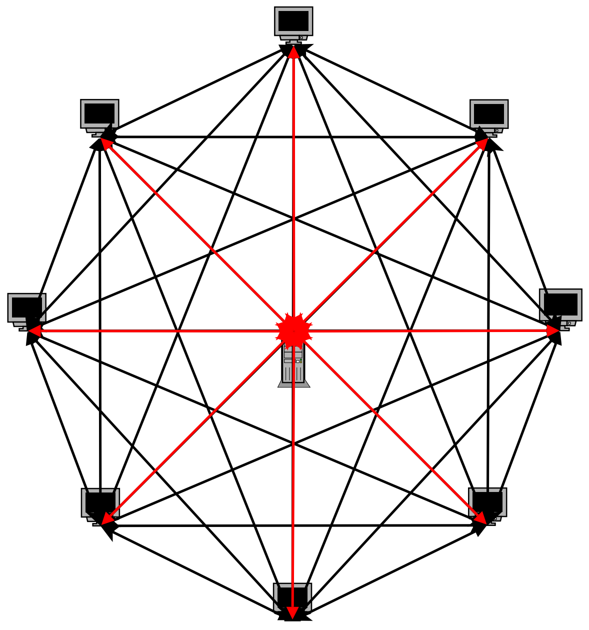

Figure 3: *An 8 node network laid out in the topology used by the original Napster network.*

primary owner, and is responsible for maintaining current copies of this piece of data on four nodes: the two closest nodes with greater id numbers, and the two closest nodes with lower id numbers.[4] Using random ID numbers for the nodes helps to ensure that nodes are not on nearby subnets, which helps prevent data from being lost if an entire subnet goes down. As nodes join and leave the network, data is moved around to ensure that these conditions are still met.

The Concurrent Versions System (CVS)[1] is a typical application that

[4]`MAX_INT` is assumed to be one less than `MIN_INT` and vice versa, which causes a graph of the network to be circular.

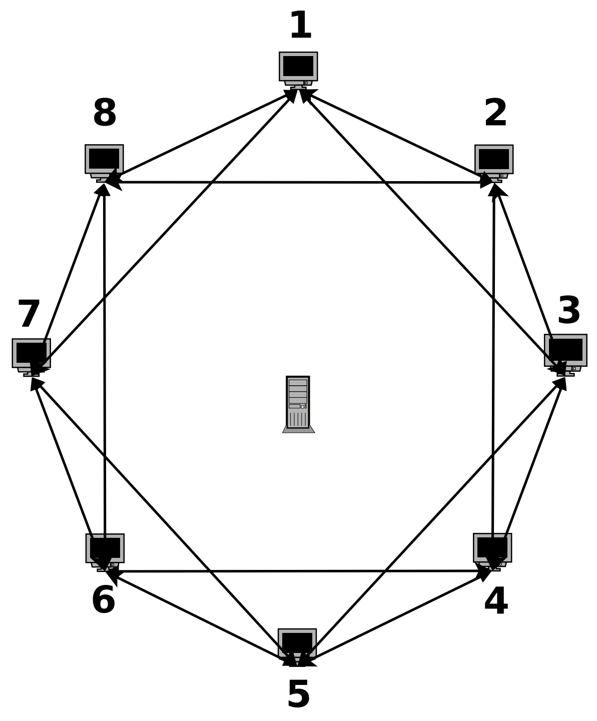

Figure 4: *An 8 node network laid out in the topology used by the Bamboo network. Numbers represent node id's.*

uses a star network topology to distribute data. A group of programmers who are potentially dispersed across the globe will log in to a central CVS server to retrieve and update changes to a shared codebase. The purpose of my thesis experiment was to see if this type of system could become a peer-to-peer system by combining it with a modular networks infrastructure, such as Bamboo. CVS was chosen as the test program for this experiment because, to the best of our knowledge, no such system already exists. CVS is also easy to work with, since it stores its file repositories in a way that is easily recreated. Finally, since Bamboo and CVS are both freely available on the internet as open-source projects, they are uniquely suited to this sort of experimentation.

3 System Overview

The distCVS system allows users to put, get and update files in a peer-to-peer network, and handles all versioning and concurrency enforcement in the background. The typical user of the system would provide it with the name of a project and a file that he or she wanted to work on, make changes to it, and then resubmit it into the network. If another user had submitted changes while this copy was being worked on, the system will notify the user that his or her file is out of date and needs to be updated before the changes will be accepted.

Each node in a distCVS network needs to run both the Bamboo software and the CVS software. This is done because Bamboo provides the network system that passes data to and from copies of CVS. Bamboo also manages a per-node database which stores copies of the cumulative version files that are produced by CVS. These files are the main pieces of data tracked by the network, and may be passed from node to node as the composition of the network changes. Finally, each node also requires a copy of CVS since all data retrieval and update operations must be done on the primary owner to ensure that the newest copy of the data is used as a reference. Due to these requirements, the system currently only runs on the Linux operating system, although it should be possible to port the components to other platforms.

4 Implementation

Each Bamboo node is a collection of smaller components, called stages, that provide specific functionality to that node. For example, there are stages that store data, route messages, and handle network connections. This modular structure therefore allowed me to extend the functionality of Bamboo by writing a stage that interacted with CVS. This stage, named DistCVSStage, sends and receives messages, places and retrieves data in the Bamboo network, and interacts with CVS primarily through the help of two perl scripts. This method was chosen because Perl is better suited for interaction with command-line resources than Java is, and therefore allows for cleaner code when automating tasks such as creating directories and moving files.

There are three main tasks that can be performed on a distCVS network: put, get and update. When a user requests that a file be put into storage in the network, his or her local copy of DistCVSStage reads the target file off of local storage, calculates its id number (GUID) and generates a BambooRouteInit message carrying a PutCVSReqPayload object. This message is routed by Bamboo to the primary owner for this piece of data, where it is received by the copy of DistCVSStage running on that node. This copy requests the current copy of the CVS version file from its local database by sending a GetByGuidReq message to the StorageManager stage running on the primary owner, which sends back a GetByGuidResp message containing the data. Since the version of Bamboo this project is based on does not

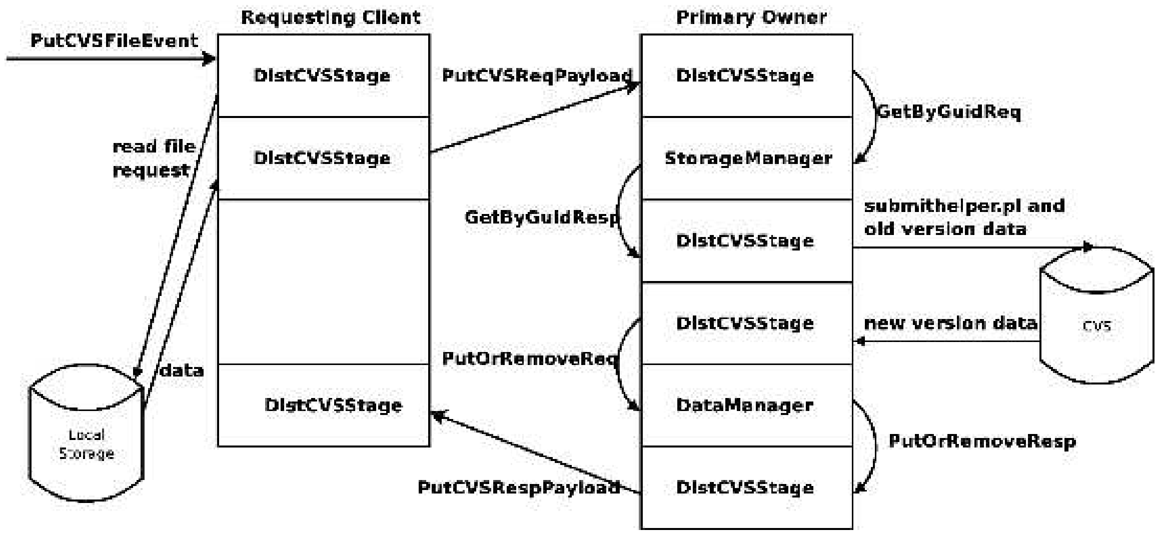

Figure 5: *An illustration of the distCVS file submission process.*

implement deletion of old resources, extra steps need to be taken to iterate through every file referenced by a given GUID, to ensure that the most recently stored copy is retrieved from the primary owner's database. If no matching data is found, then the system assumes that the submitted data is being added for the first time. Otherwise, it is treated as an update.

The retrieved version data (if it exists) is written to disk, along with the new file which was sent over the network. DistCVSStage then invokes the submithelper.pl perl script, which reconstructs a repository with version information and then invokes CVS, monitoring it for errors. If an error does occur, such as an attempted update from an obsolete version, then the error message is passed back up to DistCVSStage, which sends a message back across the network to notify the user. Otherwise, submithelper.pl copies the new version data from the CVS repository and indicates that the process completed successfully. DistCVSStage then retrieves this file from local storage and places it into the Bamboo network by passing a PutOrRemoveReq message to the DataManager stage running on the primary owner. The result of this operation is passed back to DistCVSStage in a PutOrRemoveResp message, which is used to indicate either success or failure in the PutCVSRespPayload that is wrapped in a BambooRouteInit message and sent back to the requesting user's copy of DistCVStage. The requesting user's copy of DistCVSStage then displays this message to the user, which tells him or her whether the operation succeeded or failed.

It should be noted that the file on to the requesting user's node is never altered during this process, so a failure to store the data will not result in any data loss. Also, since Bamboo is event-driven and all of its events are stored in a queue based on the order in which they arrived, it will not attempt to apply two updates to a file at the same time, which helps to ensure concurrency.

File retrieval works very similarly. When a user requests a file, a BambooRouteInit message containing a GetCVSReqPayload message is sent across the network to a primary owner. To prevent old data from being returned, a newly-joined node will delay the processing of this request for some reasonable amount of time[5] to ensure that it has the most recent copies of all of the data it needs. Since there is no central control in a peer-to-peer network,

[5]This was on the order of about 30 seconds during our tests.

there is no way to ensure that any given node has a globally correct piece of data. Therefore, we were forced to assume that the correct version would quickly propagate through the network.

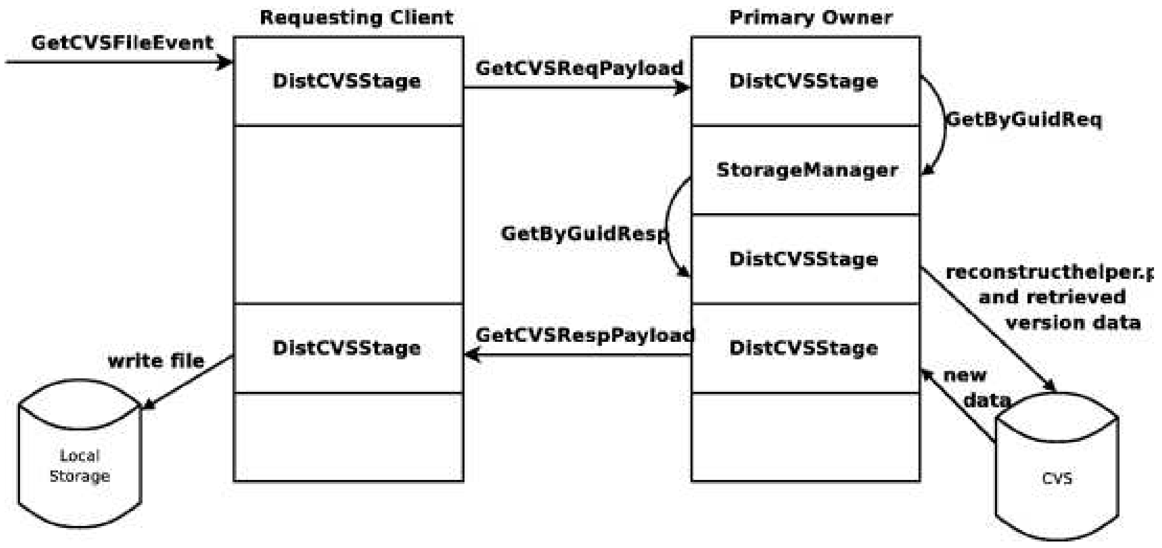

Figure 6: *An illustration of the distCVS file retrieval process*

Once this initial delay has passed, the primary owner's copy of DistCVSStage queries StorageManager for the highest version of the target file stored in the local database. If no data is found that matches the requested file's id number, a message is sent back to the requesting node informing the user that the file was not found. However, if it is found then this retrieved version data is written to local storage. DistCVSStage then invokes the reconstructhelper.pl perl script, which in turn creates a fake CVS repository and invokes CVS in such a way that it reconstructs a copy of the highest version of the data encoded within the file retrieved from the database. The DistCVSStage is alerted if this process fails, and sends a message across the network to inform the user what the problem is. Otherwise, DistCVSStage reads the reconstructed file from local storage, packages it and a version number into a GetCVSRespPayload object which is wrapped in a BambooRouteInit message and sent back across the network to the requesting user's computer. The id number of the node that made the original request is checked by the node that receives this packet, and it is discarded if they do not match. This is to prevent against users receiving data that they did not request if the node that made the request dropped from the network while the file was being reconstructed. If the GUID numbers match, however, then the reconstructed file and a file containing its version number are written to local storage and the user can then edit the file in any way he or she desires.

5 Testing

Our tests sought to answer several questions about the behavior of DistCVS. Specifically, we wanted to see how stable the system was under a high rate of file requests, or a high rate of nodes joining and dropping from the network. We also wanted to ensure that the correct data was always created or returned, even when two updates happened at once, or updates were attempted from outdated version. Finally, we wanted to see what would happen in the extreme case where the entire network disappeared and then came back online. In short, the system performed well under all of the conditions that it was tested under.

distCVS was tested on a collection of ten Intel Pentium 4 workstations running Red Hat Linux release 9.[6] The tests were automated using a series of perl scripts that prepared a unique directory for each node to run it, provided copies of all of the necessary files, created configuration files and monitored the data stored on each node for correctness. The DistCVSStage also has a test mode which will automatically create file requests, output the response to local storage, and repeat the process after a random interval of time has passed. DistCVSStage can also handle not sending any more requests after a specified time period has elapsed, which was used to flush pending messages out of the network before the nodes were shut down. Similarly, no requests are sent until the entire network is running, to avoid skewing the results data.

[6]kernel version: 2.4.20

In order to ensure that nodes received the correct copy of the data from the network, a master copy was kept on a file system shared by all of the test machines. Whenever a response was returned, it was compared against this master copy using the Linux diff utility. If the files were identical, a `GOOD_VERSION` response was recorded in the node's log. Otherwise, a `BAD_VERSION` was recorded if they differed, and a `NOT_FOUND` recorded if the data could not be retrieved from the network. Once the experiment run is complete these log files can be parsed to compile statistics for the experiment run.

The following tables contain the results from the experiments we ran, along with the conditions that they were run under. A typical experiment took around an hour to run, and consisted of a warmup period, twenty minutes of requests, the random removal of 20% of the nodes in the network, another twenty minutes of requests, the re-insertion of the removed nodes, a final twenty minutes of requests, and then a cooldown period where the requests are flushed from the network.

These results illustrates the fact that even under very adverse conditions, the network retains the ability to return either good data or a not found warning. It very rarely returns a bad version to a user. Even this bad version is likely to be caught when the user submits his or her changes since a newer version will always propagate through the network if it exits.

number of nodes	total requests	%good	%bad	%not found	%unanswered
100	2531	0.89	0	0.11	0
200	4995	0.97	0	0.03	0
300	6874	0.98	0	0.02	0.01

Table 1: 12-15 minute request interval, no updates

number of nodes	total requests	%good	%bad	%not found	%unanswered
100	11698	0.82	0	0.18	0
200					
300	9660	0.97	0	.02	0.01

Table 2: 0-30 second request interval, no updates

number of nodes	total requests	%good	%bad	%not found	%unanswered
100	1684	0.97	0	0.02	0
200					
300	7732	0.96	0	0.04	0

Table 3: 12-15 minute request interval, updates

number of nodes	total requests	%good	%bad	%not found	%unanswered
100	7981	0.71	0.03	0.25	0
200					
300	11161	0.8	0	0.2	0

Table 4: 0-30 second request interval, updates

6 Conclusions and Future Work

Due to the modular structure of Bamboo and the way that CVS interacts with data, distCVS demonstrates that it is possible to build a complex peer-to-peer system that can be rapidly upgraded to reflect any possible future advances in the technology. Although distCVS is a working system, there are still several areas where it could be improved.

distCVS is only a partial implementation of CVS, so although the framework has been built, there is still work to be done before the system supports all of the functions offered by CVS. Along a similar vein, distCVS should also be extended to retrieve entire projects at once from the network, which is closer to how CVS interacts with source code repositories.

A deeper issue with distCVS, and one that would likely hamper any large-scale deployment of the system, is the fact that all packets are sent as plain text with no sort of authentication. If a malicious user wished to eavesdrop on the files being transferred, create fake data, or even use a modified client to corrupt the data stored on it, there is currently no way to detect or stop this. Implementing some sort of encryption or node trust scheme would greatly improve the utility of this system on an untrusted network.

Finally, a major shortcoming of distributed hash tables is that there is no efficient way to search or index the contents stored on the network. Further research into this area could help solve one of the largest fundamental problems that remain in peer-to-peer networks.

7 Appendix A: distCVS Source Listings

Listing 1: DistCVSStage.java

```
/*
 * DistCVSStage.java - a Bamboo stage that interacts with CVS.
 *
 * Copyright (C) 2004,2005 Andrew Ian Logan
 * logana@bc.edu, andrewlogan@gmail.com
 *
 */

package thesis;
import bamboo.lss.ASyncCore;
import bamboo.lss.DustDevil;

import bamboo.util.StandardStage;
import bamboo.util.GuidTools;

import java.math.BigInteger;

import java.security.MessageDigest;

import java.util.Date;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Random;
import java.util.Set;
import java.util.LinkedList;
import java.util.StringTokenizer;

import ostore.util.ByteUtils; // for print_bytes function only
import ostore.util.SHA1Hash;
import ostore.util.SecureHash;
import ostore.util.QuickSerializable;
import ostore.util.InputBuffer;
import ostore.util.OutputBuffer;

import seda.sandStorm.api.StagesInitializedSignal;
import seda.sandStorm.api.ConfigDataIF;
import seda.sandStorm.api.QueueElementIF;
import seda.sandStorm.api.SingleThreadedEventHandlerIF;
import seda.sandStorm.api.EventHandlerException;

import java.nio.ByteBuffer;

import java.net.InetAddress;
import java.net.UnknownHostException;

import bamboo.db.StorageManager;

import bamboo.dht.GatewayClient;
import bamboo.dht.bamboo_put_args;
import bamboo.dht.bamboo_get_args;
import bamboo.dht.bamboo_value;
import bamboo.dht.bamboo_secret;
import bamboo.dht.bamboo_key;
import bamboo.dht.bamboo_stat;
import bamboo.dht.bamboo_placemark;
import bamboo.dht.bamboo_get_res;

import bamboo.dmgr.PutOrRemoveReq;
import bamboo.dmgr.PutOrRemoveResp;
import bamboo.dmgr.FetchDataReq;
import bamboo.dmgr.FetchDataResp;

import bamboo.dmgr.DataManager;

import java.awt.*;
import java.awt.event.*;

import javax.swing.*;

import java.io.*;
import bamboo.api.BambooRouteDeliver;
import bamboo.api.BambooRouteInit;
import bamboo.api.BambooRouterAppRegReq;
import bamboo.api.BambooRouterAppRegResp;

/**
 * This file contains the frontend and the logic for the DistCVS
 * system.
 *
 * @author Andrew Logan
 * @version $Id: DistCVSStage.java,v 1.11 2005/02/03 06:31:48 andrew Exp $
 */

public class DistCVSStage extends StandardStage
implements SingleThreadedEventHandlerIF {
    protected static final long app_id = bamboo.router.Router.app_id(DistCVSStage.class);

    protected static final int MAX_FILESIZE = 16000;

    protected static final boolean DEBUG = true;

    //how many seconds after we join the network do we start
    //processing requests?

    //TODO: Make this configurable... or even better, some function of
    //network lag & the amount of data that we're being sent.
    protected int init_delay_sec = 15;

    //testing flag - will perform reads/writes at random intervals
    protected boolean automatic_test = false;
    protected boolean updater_node = false;
    protected boolean initial_data = false;
    protected boolean show_gui = true;
    protected boolean automatic_data_submission_hack = false;
    protected String test_filename = null;
    protected String test_projectname = null;
    protected String logfile_name = null;
    protected int test_minwait = -1;
    protected int test_maxwait = -1;
    protected int test_warmup_time = -1;
    protected int test_run_time = -1;
    protected Date stopDate;

    protected static final long NO_TIME = -1;

    protected BigInteger my_guid;

    //Used for testing.  Filenames are fileprefix + random(num_files)
    protected String file_prefix = null;
    protected int num_files;

    //ttl constants (in seconds)
    protected static final int ONE_MINUTE =  60;
    protected static final int ONE_HOUR   =  60 * ONE_MINUTE;
    protected static final int ONE_DAY    =  24 * ONE_HOUR;
    protected static final int ONE_WEEK   =   7 * ONE_DAY;
    protected static final int ONE_MONTH  =  30 * ONE_DAY;
    protected static final int ONE_YEAR   = 365 * ONE_DAY;
    protected static final int MAX_INT    = Integer.MAX_VALUE;

    protected LinkedList wait_q = new LinkedList();

    protected JFrame mainFrame = null;
    protected JPanel panel = null;
    protected JTextField filenameField = null;
    protected JTextField projectnameField = null;
    protected JLabel filenameLabel = null;
    protected JLabel projectnameLabel = null;
    protected JButton submitFilenameButton = null;
    protected JRadioButton putFileRadioButton = null;
    protected JRadioButton getFileRadioButton = null;

    protected boolean retrieveFile = false;
    protected boolean initialized = false;
    ////////////////////////////////////////////////////////////////////
    // Action adapter for easy event-listener coding
    class ActionAdapter implements ActionListener
    {
        public void actionPerformed(ActionEvent e) {}
    }
    ////////////////////////////////////////////////////////////////////

    protected class GetCVSFileEvent implements QueueElementIF{
        public String filename;
        public String projectname;

        public GetCVSFileEvent(String filename, String projectname){
            this.filename = filename;
            this.projectname = projectname;
        }
    }

    protected class PutCVSFileEvent implements QueueElementIF{
        public String filename;
        public String projectname;

        public PutCVSFileEvent(String filename, String projectname){
            this.filename = filename;
            this.projectname = projectname;
        }
    }

    //NOTE!  In order to be QuickSerializable, the class has to be
    //static.  Don't know about protected or void, but probably those
    //as well.
    protected static class NullPayload implements QuickSerializable{
        public NullPayload(){};

        public NullPayload(InputBuffer b){}

        public void serialize(OutputBuffer b){}

        public String toString(){
            return "NullPayload";
        }
    }

    //Encapsulate a status message.
    protected static class StatusPayload implements QuickSerializable{
        public String msg;

        public StatusPayload(String msg){
            this.msg = msg;
        }

        public StatusPayload(InputBuffer b){
            msg = b.nextString();
        }

        public void serialize(OutputBuffer b){
            b.add(msg);
        }

        public String toString(){
            return msg;
        }
    }

    //Sent to primary owner to initiate placing data into the storage network.
    protected static class PutCVSReqPayload implements QuickSerializable{
        public String filename;
        public String projectname;
        public byte[] filedata;
        private int filesize;  //Just for the deserializer, others should get it from
filedata.length

        public PutCVSReqPayload (String filename, String projectname, byte[] filedata){
            this.filename = filename;
            this.projectname = projectname;
            this.filedata = new byte[filedata.length];
            this.filedata = filedata;
            this.filesize = filedata.length;
            System.out.println("PutCVSReqPayload: constructor");
            debug_disp_file(this.filename, this.projectname, this.filedata);
        }

        public PutCVSReqPayload (InputBuffer b){
            filename = b.nextString();
            projectname = b.nextString();
            filesize = b.nextInt();
            filedata = new byte[filesize];
            b.nextBytes(filedata, 0, filedata.length);
            System.out.println("PutCVSReqPayload: InputBuffer");
            debug_disp_file(this.filename, this.projectname, this.filedata);
        }

        public void serialize(OutputBuffer b){
            System.out.println("putcvsreqpayload:outputbuffer:start");
            b.add(filename);
            b.add(projectname);
            b.add(filesize);
            b.add(filedata);
            System.out.println("PutCVSReqPayload: serialize");
            debug_disp_file(this.filename, this.projectname, this.filedata);
            System.out.println("putcvsreqpayload:outputbuffer:start");
        }
        public String toString(){
            return "PutCVSReqPayload with filename: " + filename + " and projectname: " +
projectname;
        }
    }

    //Sent back by primary owner to indicate success or failure.  If
    //the put operation fails, message contains a description of why.
    protected static class PutCVSRespPayload implements QuickSerializable{
        public boolean error;
        public String message;
        public String filename;
        public String projectname;
        public byte[] version_filedata;
        private int filesize;

        public PutCVSRespPayload(boolean error, String message, String filename, String
projectname, byte[] version_filedata){
            this.error = error;
            this.message = message;
            this.filename = filename;
            this.projectname = projectname;
            try{
                this.version_filedata = new byte[version_filedata.length];
                this.version_filedata = version_filedata;
                this.filesize = version_filedata.length;
            }
            catch(NullPointerException npe){
                this.version_filedata = new byte[0];
                this.filesize = 0;
            }
        }

        public PutCVSRespPayload(InputBuffer b){
            error = b.nextBoolean();
            message = b.nextString();
            filename = b.nextString();
            projectname = b.nextString();
            filesize = b.nextInt();
            version_filedata = new byte[filesize];
            b.nextBytes(version_filedata, 0, version_filedata.length);
        }

        public void serialize(OutputBuffer b){
            b.add(error);
            b.add(message);
            b.add(filename);
            b.add(projectname);
            b.add(filesize);
            b.add(version_filedata);
        }
    }
    //Sent to primary owner to initiate retrieval of data from the
    //storage network.
    protected static class GetCVSReqPayload implements QuickSerializable{
        public String filename;
        public String projectname;
        public boolean testing;

        public GetCVSReqPayload (String filename, String projectname, boolean testing){
            this.filename = filename;
            this.projectname = projectname;
            this.testing = testing;
        }

        public GetCVSReqPayload (InputBuffer b){
            filename = b.nextString();
            projectname = b.nextString();
            testing = b.nextBoolean();
        }

        public void serialize (OutputBuffer b){
            b.add(filename);
            b.add(projectname);
            b.add(testing);
        }
    }

    //Sent to requesting node after retrieval operation occurs.  If
    //success is false, then filedata has length zero.
    protected static class GetCVSRespPayload implements QuickSerializable{
        public String filename;
        public String projectname;
        public boolean success;
        public byte[] filedata;
        public String msg;
        private int filesize;  //Just for the deserializer, others should get it from
filedata.length
        public boolean testing;
        public byte[] master_filedata;
        private int master_filesize;  //Just for the deserializer, others should get it from
filedata.length

        public GetCVSRespPayload (String filename, String projectname, boolean success, byte[]
filedata, String msg, boolean testing, byte[] master_filedata){
            this.filename = filename;
            this.projectname = projectname;
            this.success = success;

            //TODO: Do I really have to enforce this here?
            if(success){
                this.filedata = new byte[filedata.length];
                this.filedata = filedata;
                this.filesize = filedata.length;
            }
            else{
                //I guess...  At least it will have a .length, which
                //is absent when null. (?)
                this.filesize = 0;
                this.filedata = new byte[0];
            }

            this.msg = msg;
            this.testing = testing;

            try{
                this.master_filedata = new byte[master_filedata.length];
                this.master_filedata = master_filedata;
                this.master_filesize = master_filedata.length;
            }
            catch(NullPointerException npe){
                this.master_filedata = new byte[0];
                this.master_filesize = 0;
            }

        }

        public GetCVSRespPayload (InputBuffer b){
            //I keep track of the length here since I get an underflow
            //error if I try to take more data from the byte buffer
            //than it actually has.
            filename = b.nextString();
            projectname = b.nextString();
            success = b.nextBoolean();
            filesize = b.nextInt();
            filedata = new byte[filesize];
            b.nextBytes(filedata, 0, filedata.length);
            msg = b.nextString();
            testing = b.nextBoolean();
            master_filesize = b.nextInt();
            master_filedata = new byte[master_filesize];
            b.nextBytes(master_filedata, 0,master_filedata.length);
        }

        public void serialize (OutputBuffer b){
            b.add(filename);
            b.add(projectname);
            b.add(success);
            b.add(filesize);
            b.add(filedata);
            b.add(msg);
            b.add(testing);
            b.add(master_filesize);
            b.add(master_filedata);
        }
    }

    //Used on the primary owner side to attach data to requests through the system.
    protected class GuidReqInfoContainer{
        public String filename;
        public String projectname;
        public BigInteger src;
        public long time_usec;
        public byte[] filedata;
        public boolean testing;

        public GuidReqInfoContainer(String filename, String projectname, BigInteger src, boolean
testing){
            this.filename = filename;
            this.projectname = projectname;
            this.src = src;
            this.time_usec = NO_TIME;
            this.filedata = null;
            this.testing = testing;
        }
    }

    //Used on the primary owner side to attach data to requests through the system.
    protected class PutCVSReqInfoContainer{
        public String filename;
        public String projectname;
        public BigInteger src;
        public long time_usec;
        public byte[] filedata;
        public byte[] version_filedata;

        public PutCVSReqInfoContainer(String filename, String projectname, byte[] filedata,
BigInteger src){
            this.filename = filename;
            this.projectname = projectname;
            this.src = src;
            this.time_usec = NO_TIME;
            this.filedata = new byte[filedata.length];
            this.filedata = filedata;
            this.version_filedata = version_filedata;
        }
    }

    //Forces a delay between when the node integrates into the network
    //and when it begins sending responses.  This was added to avoid
    //returning the wrong data when a node just integrates into the
    //system.
    protected static class IntegrationDelayFinishedAlarm implements QueueElementIF{};

    //For testing purposes only.  Will retrieve, modify, & submit a
```

```
//specific file at random intervals.
protected static class RetrieveTestDataAlarm implements QueueElementIF{};
protected static class SubmitTestDataAlarm implements QueueElementIF{};
//    protected static class InitialDataAlarm implements QueueElementIF{};

//Alerts the system that the testing warmup time has been completed.
protected static class WarmupCompletedAlarm implements QueueElementIF{};

//I don't actually use this anymore.
public boolean die_on_get_failure;

//Constructor
public DistCVSStage() throws Exception{
    //DEBUG = true;

    //Register payloads
    ostore.util.TypeTable.register_type(NullPayload.class);
    ostore.util.TypeTable.register_type(StatusPayload.class);
    ostore.util.TypeTable.register_type(GetCVSReqPayload.class);
    ostore.util.TypeTable.register_type(GetCVSRespPayload.class);
    ostore.util.TypeTable.register_type(PutCVSReqPayload.class);
    ostore.util.TypeTable.register_type(PutCVSRespPayload.class);

    event_types = new Class[]{
        StagesInitializedSignal.class,
        PutCVSFileEvent.class,
        GetCVSFileEvent.class,
        PutOrRemoveResp.class,
        StorageManager.PutResp.class,
        StorageManager.GetByGuidResp.class,
        StorageManager.GetByTimeResp.class,
        RetrieveTestDataAlarm.class,
        SubmitTestDataAlarm.class,
        //      InitialDataAlarm.class,
        WarmupCompletedAlarm.class,
        IntegrationDelayFinishedAlarm.class
    };
}

protected Random rand;
protected Set puts = new HashSet ();
protected MessageDigest digest;
protected GatewayClient client;
protected bamboo.lss.ASyncCore acore;
protected double mean_get_period_ms, mean_put_period_ms;
protected int storage_goal;
protected int total_storage;

protected static double random_exponential(double mean, Random rand) {
    double u = rand.nextDouble();
    return (0 - (mean * Math.log(1.0 - u)));
}

protected int [] ttl_values = { 3600, 24*3600, 7*24*3600 };
protected int [] put_sizes = { 32, 64, 128, 256, 512, 1024 };
protected long next_seq;
protected HashMap pending_puts = new HashMap ();
protected HashMap pending_gets = new HashMap ();

//cvs-ify the file.
protected String make_cvs_file(String initialFilename) throws IOException, InterruptedException{
    System.out.println("make_cvs_file:");

    Runtime runtime = Runtime.getRuntime();
    String command, resultingFilename;

    command = "pwd";

    Process whereAreWeProc = runtime.exec(command);

    InputStream inputstream = whereAreWeProc.getInputStream();
    InputStreamReader inputstreamreader = new InputStreamReader(inputstream);
    BufferedReader bufferedreader = new BufferedReader(inputstreamreader);

    //read output
    String line;
    while ((line = bufferedreader.readLine()) != null){
            System.out.println(line);
        }

        command = "./storehelper.pl ";
        resultingFilename = initialFilename + ",v";

        Process proc = runtime.exec(command + initialFilename + " > storehelperoutput");

        //check for failure
        if(proc.waitFor() != 0){
            System.err.println("Exit value = " + proc.exitValue());
            throw new InterruptedException();
        }

        return resultingFilename;
    }

    //TODO: What to do?
    public void do_add(String filename, String projectname){}

    public void do_update(String filename, String projectname){}

    public void do_checkin(String filename, String projectname){}

    public void do_checkout(String filename, String projectname){}
    //TODO: end "what to do?"

    public void do_put(String filename, String projectname){
        System.out.println("DistCVSStage: do_put");

        try{
            byte[] filedata = read_file(filename, projectname);
            byte[] file_header = null;
            byte[] completed_filedata = null;
            //This will fail if it's an add, so we catch that here.
            try{
                file_header = read_file(filename+"_version", projectname);
                completed_filedata = ((new String(file_header)) + (new
String(filedata))).getBytes();
            }

            //Null or a newline?
            catch(FileNotFoundException fnfe){
                completed_filedata = (new String(filedata)).getBytes();
            }

            long now = now_ms() * 1000;
            int ttl_sec = 3600*24*7;
            /*
            String hashname = projectname + ":" + filename;
            SecureHash securehash = new SHA1Hash(digest.digest(hashname.getBytes()));
            BigInteger guid = GuidTools.secure_hash_to_big_integer(securehash);
            */
            BigInteger guid = make_hashname(filename, projectname);
            ByteBuffer bb = ByteBuffer.wrap(filedata);
            InetAddress clientAddress = InetAddress.getLocalHost();

            //<extra junk>
            //byte[] dh = data_hash(completed_filedata);
            //System.out.println("data_hash: 0x" + ByteUtils.print_bytes(dh, 0, 8));
            //</extra junk>

            //Place our data into the system
            //StorageManager only acts locally, DataManager wraps StorageManager globally.
            //classifier.dispatch_later(new StorageManager.PutReq (key, bb, my_sink, "teststring"),
0);
            //classifier.dispatch_later(new PutOrRemoveReq(now, ttl_sec, guid, bb, true,
clientAddress, my_sink, null), 0);
            //classifier.dispatch_later(new PutCVSReq(filename, projectname, filedata), 0);

            //Dispatch data to primary owner
            classifier.dispatch_later(new BambooRouteInit(guid, app_id, false, false, new
PutCVSReqPayload(filename, projectname, completed_filedata)), 0);

            debug_disp_file(filename, projectname, filedata);

            logger.info("Doing a put: filename: " + filename + " projectname: " + projectname);
        }
        catch(Exception e){
            e.printStackTrace();
            System.err.println("Error!: " + e);
        }
    }

    //TODO: Try to spread the load across all of the nodes that store data, not just the primary
owner.
    public void do_get(String filename, String projectname){
        System.out.println("DistCVSStage: do_get");
        boolean errorCondition = false;

        String resultingFilename = new String();

        try{
            //TODO: Why do I still do this?  I don't even use
            //resultingFilename anymore.
            resultingFilename = make_cvs_file(filename);
        }
        catch(Exception e){
            System.out.println("Caught: " + e);
            errorCondition = true;
        }

        if(!errorCondition){
            try{
                //byte[] filearray = new byte[MAX_FILESIZE];

                //FileInputStream fis = new FileInputStream(resultingFilename);

                /*
                String hashname = projectname + ":" + filename;

                //Make guid
                SecureHash securehash = new SHA1Hash(digest.digest(hashname.getBytes()));
                BigInteger guid = GuidTools.secure_hash_to_big_integer(securehash);
                */
                BigInteger guid = make_hashname(filename, projectname);

                //Send a message to primary owner (closest to guid).
                //classifier.dispatch_later(new BambooRouteInit(guid, app_id, false, false, new
GUIDReqPayload(filename, projectname)), 0);
                classifier.dispatch_later(new BambooRouteInit(guid, app_id, false, false, new
GetCVSReqPayload(filename, projectname, automatic_test)), 0);

                logger.info("Doing a get:");
            }
            catch(Exception e){
                e.printStackTrace();
                System.err.println("Error!: " + e);
            }
        }
    }

    public void init (ConfigDataIF config) throws Exception {
        super.init (config);
        die_on_get_failure = config_get_boolean (config, "die_on_get_failure");
        mean_put_period_ms = config_get_int (config, "mean_put_period_ms");
        if (mean_put_period_ms == -1.0)
            mean_put_period_ms = 60.0*1000.0;
        mean_get_period_ms = config_get_int (config, "mean_get_period_ms");
        if (mean_get_period_ms == -1.0)
            mean_get_period_ms = 60.0*1000.0;
        storage_goal = config_get_int (config, "storage_goal");
        if (storage_goal == -1)
            storage_goal = 10*1024;
        int seed = config_get_int (config, "seed");
        if (seed == -1)
            seed = ((int) now_ms ()) ^ my_node_id.hashCode();
        rand = new Random (seed);
        try { digest = MessageDigest.getInstance("SHA"); }
        catch (Exception e) { assert false; }
        acore = DustDevil.acore_instance ();
        String client_stg_name = config_get_string (config, "client_stage_name");
        logfile_name = config_get_string (config, "logfile_name");
        System.out.println("logfile_name: " + logfile_name);
        client = (GatewayClient) lookup_stage (config, client_stg_name);

        automatic_test = config_get_boolean(config, "automatic_test");
        updater_node = config_get_boolean(config, "updater_node");
        initial_data = config_get_boolean(config, "initial_data");
        show_gui = config_get_boolean(config, "show_gui");
        automatic_data_submission_hack = config_get_boolean(config,
"automatic_data_submission_hack");
        System.out.println("DistCVSStage: initial_data: " + initial_data);
        test_maxwait = config_get_int(config, "test_maxwait");
        if(test_maxwait == -1){
            //milliseconds
            test_maxwait = 60 * 1000;
        }
        test_minwait = config_get_int(config, "test_minwait");
        if(test_minwait == -1){
            //milliseconds
            test_minwait = 30 * 1000;
        }

        System.out.println("Initialization: test_maxwait: " + test_maxwait + " test_minwait: " +
test_minwait);

        if(automatic_test){
            test_filename = config_get_string(config, "test_filename");
            if(test_filename == null){
                System.err.println("Need to pass a test_filename if automatic_test is true!");
                System.exit(1);
            }
            test_projectname = config_get_string(config, "test_projectname");
            if(test_filename == null){
                System.err.println("Need to pass a test_projectname if automatic_test is true!");
                System.exit(1);
            }
            test_warmup_time = config_get_int(config, "test_warmup_time");
            if(test_warmup_time == -1){
                //milliseconds
                test_warmup_time = 0;
            }

            //-1 (the default value) indicates that the program should
            //run forever, although routines checking for this should
            //first see if automatic_test is true.
            test_run_time = config_get_int(config, "test_run_time");

            if (test_run_time != -1){
                //stop test_run_time seconds from now
                stopDate = new Date(test_run_time + ((new Date()).getTime()));
            }
        }

        //Note!  Don't uncomment without removing the my_guid
        //reference in initGUI (throws a null pointer exception)
        //initGUI();
    }

    public void handleEvent (QueueElementIF elem) throws EventHandlerException{
        if (logger.isDebugEnabled()) logger.debug("got " + elem);
        System.out.println("DistCVSStage: initialized:" + initialized);

        if(!initialized){
            //sent after all stages have been initialized (on this node)
            if(elem instanceof StagesInitializedSignal){
                //Request registration
                dispatch(new BambooRouterAppRegReq(app_id,
                                                   false,
                                                   false,
                                                   false,
                                                   my_sink));
            }
            else if (elem instanceof BambooRouterAppRegResp){

                //TODO: Add in some sort of delay specified by the
                //perl script which will take into account the fact
                //that the network needs some time to get connected.
                //Then, copy these routines into the Finished
                //notification for that message.
                System.out.println("DistCVSStage: got BambooRouterAppRegResp");

                BambooRouterAppRegResp resp = (BambooRouterAppRegResp) elem;

                my_guid = resp.node_guid;

                try{append_to_logfile("Node started with stopDate=" + stopDate.toString() + "\n");}
                catch(IOException ioe){System.err.println("Error appending to logfile!");}

                //The delay on this message is how long we want to
                //wait before we begin processing messages.  It should
                //be set a bit longer than the average amount of time
                //it takes to give the right data to a node.
                classifier.dispatch_later(new IntegrationDelayFinishedAlarm(), init_delay_sec *
1000);
            }
            else if (elem instanceof IntegrationDelayFinishedAlarm){
                System.out.println("DistCVSStage: got IntegrationDelayFinishedAlarm");

                //Now that we're a member of the network, allow the
                //user to interact with the node.  This was done
                //because we won't have my_guid until here, but it
                //does make the GUI not show up until we're integrated
                //into the network.

                //TODO: Think about if this is the best solution.
                //There must be a method to change the titlebar.
                if(show_gui){
                    initGUI();
                }

                //TODO: Cleanup!
                //Are we automating the submit process?
                if(automatic_data_submission_hack){
                    classifier.dispatch_later(new PutCVSFileEvent("foo1", "foo1"), 0);
                }

                //Are we testing the network?
                if(automatic_test){
                    if(test_warmup_time == -1){
                        classifier.dispatch_later(new WarmupCompletedAlarm(), 0);
                    }
                    else{
                        classifier.dispatch_later(new WarmupCompletedAlarm(), test_warmup_time);
                    }
                }

                else{
                    initialized = true;

                    try{append_to_logfile("Node processing messages\n");}
                    catch(IOException ioe){System.err.println("Error appending to logfile!");}

                    //Clean out the queue that may have built up.
                    while(!wait_q.isEmpty()){
                        handleEvent((QueueElementIF)wait_q.removeFirst());
                    }
                }
            }

            //BambooRouteDeliver messages are the only ones sent
            //across the network.  See what we've got here.
            else if (elem instanceof BambooRouteDeliver){
                //Don't want to lose it.
                wait_q.addLast(elem);

                BambooRouteDeliver info = (BambooRouteDeliver)elem;

                if(info.payload instanceof GetCVSReqPayload){
                    GetCVSReqPayload pay = (GetCVSReqPayload) info.payload;

                    classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false, false,
new StatusPayload("Initializing database - your request will be processed shortly.")), 0);
                }
                else if(info.payload instanceof PutCVSReqPayload){
                    PutCVSReqPayload pay = (PutCVSReqPayload) info.payload;
                    classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false, false,
new StatusPayload("Initializing database - your request will be processed shortly")), 0);
                }
            }

            //WarmupCompletedAlarm messages are only sent if the system is being tested.
            else if (elem instanceof WarmupCompletedAlarm){
                handle_warmup_completed_alarm();
            }

            else{
                wait_q.addLast(elem);
            }
        }
        else{
            if (elem instanceof GetCVSFileEvent){
                GetCVSFileEvent fileEvent = (GetCVSFileEvent)elem;
                System.out.println("calling do_get");
                do_get(fileEvent.filename, fileEvent.projectname);
            }
            else if (elem instanceof PutCVSFileEvent){
                PutCVSFileEvent fileEvent = (PutCVSFileEvent)elem;
                do_put(fileEvent.filename, fileEvent.projectname);
            }
            else if (elem instanceof StorageManager.PutResp){
                System.out.println("Got a PutResp: " + elem);
            }
            else if (elem instanceof StorageManager.GetByGuidResp){
                handle_get_by_guid_resp((StorageManager.GetByGuidResp) elem);
            }
            else if (elem instanceof PutOrRemoveResp){
                handle_put_or_remove_resp((PutOrRemoveResp) elem);
            }
            else if (elem instanceof BambooRouteDeliver){
                handle_bamboo_route_deliver((BambooRouteDeliver) elem);
            }
            //else if (elem instanceof InitialDataAlarm){
            //handle_initial_data_alarm();
            //}
            else if (elem instanceof RetrieveTestDataAlarm){
                handle_retrieve_test_data_alarm();
            }
            else if (elem instanceof SubmitTestDataAlarm){
                handle_submit_test_data_alarm();
            }
            else{
                BUG("Event " + elem +" unknown");
            }
        }
    }

    //TODO: Do something better than have 16K chunks of file (have to change this is to_put as well)
    protected void handle_get_by_guid_resp(StorageManager.GetByGuidResp elem){

        System.out.println("DistCVSStage: key: " + elem.key);
        System.out.println("DistCVSStage: data: " + elem.data);
        System.out.println("DistCVSStage: continuation: " + elem.continuation);
        //System.out.println("DistCVSStage: user_data: " + (String)elem.user_data);

        //Is there potentially more data to retrieve from the network?
        if(elem.continuation != null){
            if(elem.user_data instanceof GuidReqInfoContainer){
                if(elem.key != null){
                    GuidReqInfoContainer info = (GuidReqInfoContainer)elem.user_data;
                    //Is the data we're looking at newer than the data we have stored?
                    if(elem.key.time_usec > info.time_usec){
                        info.time_usec = elem.key.time_usec;
                        info.filedata = new byte[elem.data.remaining()];
                        elem.data.get(info.filedata);
                    }
                }
                else{
                    //I don't think this can happen, so throw an error telling me to implement it
if it does.
                    BUG("non-null continuation with null key in a GuidReqInfoContainer - have to
implement sending a file not found message.");
```

```
                    //Note -- this is essentially the implementation here, just make info valid
down here as well
                    //classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false, false,
new GetCVSRespPayload(info.filename, info.projectname, false, new byte[0], "File not found.")), 0);
                }
            }
            else if(elem.user_data instanceof PutCVSReqInfoContainer){
                if(elem.key != null){
                    PutCVSReqInfoContainer info = (PutCVSReqInfoContainer)elem.user_data;
                    //Are we looking at newer data than what we have stored?
                    if(elem.key.time_usec > info.time_usec){
                        info.time_usec = elem.key.time_usec;
                        info.version_filedata = new byte[elem.data.remaining()];
                        elem.data.get(info.version_filedata);
                    }
                }
                else{
                    //This would be the case where you're adding data
                    //to the network for the first time.  I really
                    //don't think this can happen, so I'll throw an
                    //error if it does.

                    BUG("non-null continuation with null key in a PutCVSReqInfoContainer - have to
implement adding data in this case.");
                }
            }
            else{
                BUG("invalid user_data");
            }
            //Whatever the user_data is, the continuation message is the same
            classifier.dispatch_later(new StorageManager.GetByGuidCont(elem.continuation, false,
my_sink, elem.user_data), 0);
        }

        //continuation is null, so we've looked through all of the data we need to.
        else{
            /* (primary owner)
             * get data and return it to a requesting node.
             */
            if(elem.user_data instanceof GuidReqInfoContainer){
                GuidReqInfoContainer info = (GuidReqInfoContainer)elem.user_data;

                //do we have data?
                if((info.time_usec != NO_TIME) && (info.filedata != null)){
                    byte[] retrieved_filedata = info.filedata;

                    try{
                        //If we're testing, output that this is the
                        //primary owner node.  This information will
                        //be used to pick up where the primary owner
                        //is.

                        if(automatic_test){
                            Runtime runtime = Runtime.getRuntime();
                            Process proc = runtime.exec("./appendfiledata.pl " + "PRIMARY_OWNER:
serving request");

                            if(proc.waitFor() != 0){
                                System.err.println("Exit value = " + proc.exitValue());
                                throw new InterruptedException();
                            }
                        }

                        //Alright, we know that everything up to the
                        //first newline is a header.  Strip this out
                        //and write it to a separate file.

                        byte[] version_header = (((new String(retrieved_filedata)).split("\\n"))[0]
+ "\n").getBytes();

                        byte[] version_filedata = ((new
String(retrieved_filedata)).substring(version_header.length)).getBytes();


                        System.out.println("Grabbed from the database:");
                        debug_disp_file(info.filename + "_version", info.projectname,
version_header);
```

```
                        debug_disp_file(info.filename +",v", info.projectname, version_filedata);

                        //I don't need this, since I can just append it to the reconstructed data.
                        //version file
                        //write_file(info.filename + "_version", info.projectname, version_header);

                        //,v file
                        write_file(info.filename + ",v", info.projectname, version_filedata);

                        Runtime runtime = Runtime.getRuntime();

                        //does the requisite CVS commands

                        String command = "./reconstructhelper.pl ";

                        Process proc = runtime.exec(command + info.filename + ",v" + " >
helperoutput" );

                        //check for failure
                        if(proc.waitFor() != 0){
                            System.err.println("reconstructhelper exit value = " +
proc.exitValue());

                            //TODO: write new exception that passes error code.
                            throw new InterruptedException();
                        }

                        //So far so good.  Read the newly reconstructed file off of the disk.
                        byte[] new_filedata = read_file(info.filename, info.projectname);
                        debug_disp_file(info.filename, info.projectname, new_filedata);

                        //Place the header onto the new data.
                        byte[] completed_filedata = ((new String(version_header)) + (new
String(new_filedata))).getBytes();

                        //TODO: should clean up temporary data.

                        byte[] master_filedata = null;
                        //Grab the master filedata
                        if(automatic_test){
                            master_filedata = read_file("../master_" + info.filename,
info.projectname);
                        }

                        //Send the data back to the requesting node
                        //classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false,
false, new GUIDRespPayload(info.filename, info.projectname, filedata)), 0);
                        classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false,
false, new GetCVSRespPayload(info.filename, info.projectname, true, completed_filedata, "OK",
info.testing, master_filedata)), 0);
                    }
                    catch(IOException ioe){
                        //I/O Error!
                        System.err.println("I/O error in handle_get_by_guid_resp\n" + ioe);
                        classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false,
false, new GetCVSRespPayload(info.filename, info.projectname, false, new byte[0], "I/O error on
parent node.", info.testing, new byte[0])), 0);
                    }
                    catch(InterruptedException ie){
                        //submithelper error
                        System.err.println("reconstructhelper error\n" + ie);
                        classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false,
false, new GetCVSRespPayload(info.filename, info.projectname, false, new byte[0],
"reconstructhelper.pl error on parent node.", info.testing, new byte[0])), 0);
                    }
                }
                else{
                    //If we can't find it, then it doesn't exist.  Let the client know this.
                    //classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false, false,
new NullPayload()), 0);
                    classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false, false,
new GetCVSRespPayload(info.filename, info.projectname, false, new byte[0], "File not found.",
info.testing, new byte[0])), 0);
                }
            }
            /* (primary owner)
             * update a diff already in the database.
            */
            else if(elem.user_data instanceof PutCVSReqInfoContainer){
                PutCVSReqInfoContainer info = (PutCVSReqInfoContainer)elem.user_data;
                //do we have data?
                if((info.time_usec != NO_TIME) && (info.version_filedata != null)){
                    //we retrieved the ,v file from the database, so use it to base the new version
on.
                    byte[] retrieved_filedata = info.version_filedata;
                    try{
                        //Alright, we know that everything up to the
                        //first newline is a header.  Strip this out
                        //and write it to a separate file.

                        byte[] retrieved_version_header = (((new
String(retrieved_filedata)).split("\\n"))[0] + "\n").getBytes();
                        byte[] retrieved_version_filedata = ((new
String(retrieved_filedata)).substring(retrieved_version_header.length)).getBytes();

                        byte[] submitted_version_header = (((new
String(info.filedata)).split("\\n"))[0] + "\n").getBytes();
                        byte[] submitted_version_filedata = ((new
String(info.filedata)).substring(retrieved_version_header.length)).getBytes();

                        //retrieved version file
                        write_file(info.filename + ",v_version", info.projectname,
retrieved_version_header);

                        //,v file
                        write_file(info.filename + ",v", info.projectname,
retrieved_version_filedata);

                        //Submitted version.
                        write_file(info.filename + "_version", info.projectname,
submitted_version_header);

                        //Submitted update.
                        write_file(info.filename, info.projectname, submitted_version_filedata);

                        System.out.println("reconstruction stage:");
                        debug_disp_file(info.filename + ",v_version", info.projectname,
retrieved_version_header);
                        debug_disp_file(info.filename + ",v", info.projectname,
retrieved_version_filedata);
                        debug_disp_file(info.filename + "_version", info.projectname,
submitted_version_header);
                        debug_disp_file(info.filename, info.projectname,
submitted_version_filedata);

                        Runtime runtime = Runtime.getRuntime();

                        //does the requisite CVS commands

                        String command = "./submithelper.pl ";

                        Process proc = runtime.exec(command + info.filename +" >
submithelperoutput");

                        //check for failure
                        //TODO: Will a nonzero return value pass over
                        //the rest of the code?
                        if(proc.waitFor() != 0){
                            System.err.println("submithelper exit value = " + proc.exitValue());

                            //TODO: write new exception that passes error code.
                            throw new InterruptedException();
                        }

                        //So far so good.  Read the new version and ,v
                        //files off of the disk, put them together,
                        //and then insert them into the database.

                        byte[] new_version_header = read_file(info.filename + ",v_version",
info.projectname);
                        byte[] new_version_data = read_file(info.filename + ",v", info.projectname);
                        byte[] new_filedata = ((new String(new_version_header)) + (new
String(new_version_data))).getBytes();
```

```
                        debug_disp_file(info.filename +",v_version",  info.projectname ,
new_version_header);
                        debug_disp_file(info.filename +",v",  info.projectname , new_version_data);
                        debug_disp_file(info.filename +",v",  info.projectname ,
submitted_version_filedata);

                        //Place the version data into info
                        info.version_filedata = new_version_header;

                        long now = now_ms() * 1000;

                        //Note!  I don't have elem.key anymore.

                        //TODO: Make the client pass ttl_sec along
                        //with file write requests.
                        int ttl_sec = ONE_WEEK;
                        /*
                        String hashname = info.projectname + ":" + info.filename;
                        SecureHash securehash = new SHA1Hash(digest.digest(hashname.getBytes()));
                        BigInteger guid = GuidTools.secure_hash_to_big_integer(securehash);
                        */
                        BigInteger guid = make_hashname(info.filename, info.projectname);
                        //guid doesn't change, so why recalculate it? -- because now we don't have
it anymore.
                        //BigInteger guid = elem.key.guid;
                        ByteBuffer bb = ByteBuffer.wrap(new_filedata);
                        InetAddress clientAddress = InetAddress.getLocalHost();

                        //TODO: Make this work.
                        //delete (recycle?) the old one
                        //classifier.dispatch_later(new PutOrRemoveReq(/*elem.key.time_usec*/now,
elem.key.ttl_sec, elem.key.guid, ByteBuffer.wrap(version_filedata), false, clientAddress, my_sink,
null), 0);

                        //submit the new one.
                        classifier.dispatch_later(new PutOrRemoveReq(now, ttl_sec, guid, bb, true,
clientAddress, my_sink, info), 0);
                    }

                    catch(IOException ioe){
                        System.err.println("I/O error!:\n" + ioe);
                        classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false,
false, new PutCVSRespPayload(true, "I/O error on parent node", info.filename, info.projectname,
null)), 0);
                    }

                    catch(InterruptedException ie){
                        System.err.println("Problem with submithelper.pl!\n" + ie);
                        classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false,
false, new PutCVSRespPayload(true, "submithelper.pl problem on parent node", info.filename,
info.projectname, null)), 0);
                    }
                }
                else{
                    //NOTE: This is also the case if the data is being
                    //added for the first time.

                    //TODO: This is an ugly implementation - we should
                    //have the client specifically ask for an add, not
                    //just assume that one is wanted when the file is not
                    //found.

                    //classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false, false,
new PutCVSRespPayload(true, "File not found in database - attempting to add", info.filename,
info.projectname, null)), 0);
                    classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false, false,
new StatusPayload("File not found in database, performing an add\n")), 0);
                    try{
                        debug_disp_file(info.filename, info.projectname, info.filedata);

                        //Submitted file.
                        write_file(info.filename, info.projectname, info.filedata);

                        //Turn it into a ,v file.
                        String new_filename = make_cvs_file(info.filename);

                        //grab the resulting file
                        byte[] new_filedata = read_file(new_filename, info.projectname);
                        byte[] new_filedata_version = read_file(new_filename + "_version",
info.projectname);

                        info.version_filedata = new_filedata_version;

                        debug_disp_file(new_filename, info.projectname, new_filedata);
                        debug_disp_file(new_filename + "_version (adding file)", info.projectname,
new_filedata_version);

                        //Place the header onto the new data.
                        byte[] completed_filedata = ((new String(new_filedata_version)) + (new
String(new_filedata))).getBytes();

                        debug_disp_file("completed_filedata", info.projectname, completed_filedata);

                        //If we're testing, write the master copy of
                        //the data to the filesystem (we're being run
                        //in our own directory, so keep one central
                        //copy of this).
                        /*
                        if(automatic_test){
                            //I want the most recent update, not the most recent ,v file.
                            try{write_file("../master_"+info.filename, info.projectname,
info.filedata);}
                            catch(IOException ioe){
                                BUG(ioe);
                            }
                        }
                        */
                        //place it into the database
                        long now = now_ms() * 1000;
                        int ttl_sec = ONE_WEEK;

                        /*
                        String hashname = info.projectname + ":" + info.filename;
                        SecureHash securehash = new SHA1Hash(digest.digest(hashname.getBytes()));
                        BigInteger guid = GuidTools.secure_hash_to_big_integer(securehash);
                        */
                        BigInteger guid = make_hashname(info.filename, info.projectname);
                        ByteBuffer bb = ByteBuffer.wrap(completed_filedata);
                        InetAddress clientAddress = InetAddress.getLocalHost();

                        classifier.dispatch_later(new PutOrRemoveReq(now, ttl_sec, guid, bb, true,
clientAddress, my_sink, info), 0);
                    }

                    catch(IOException ioe){
                        System.err.println("I/O Error!\n" + ioe);
                        classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false,
false, new PutCVSRespPayload(true, "Add operation failed - I/O error on primary owner.",
info.filename, info.projectname, null)), 0);
                    }
                    catch(InterruptedException ie){
                        System.err.println("Error running storehelper.pl!\n" + ie);
                    }
                }
            }
            else{
                BUG("invalid user_data");
            }
        }
    }

    //We've recieved notification that our PutOrRemoveReq was
    //processed, so send a message back to the requesting node telling
    //it that the submit is complete.
    protected void handle_put_or_remove_resp(PutOrRemoveResp elem){

        if(elem.user_data instanceof PutCVSReqInfoContainer){
            System.out.println("Recieved known PutOrRemoveResp: " + elem);
            PutCVSReqInfoContainer info = (PutCVSReqInfoContainer)elem.user_data;

            //If we're testing, write the master copy of the data to
            //the filesystem (we're being run in our own directory, so
            //keep one central copy of this).
            if(automatic_test){
                //Strip out the header that's included in info.filedata
                byte[] master_version_header = (((new String(info.filedata)).split("\\n"))[0] +
"\n").getBytes();
                byte[] master_version_filedata = ((new
String(info.filedata)).substring(master_version_header.length)).getBytes();

                //I want the most recent update, not the most recent ,v file.

                //TODO: Clean this up

                //TODO: Fix the fact that there is no header on an add.

                //$line =~ s/\/$inputfile\/[0-9]\.[0-9]\/(.*)/\/$inputfile\/$versionnum\/$1/gi;

                //If we're version 1.1.1.1, we're an add.
                String temp = (new String(info.version_filedata)).substring(0,
10+info.filename.length());

                System.out.println("****temp: " + temp);

                if(temp.equals(new String("/" + info.filename + "/1.1.1.1/"))){
                    System.out.println("*****ADD VERSION DETECTED*****");
                    master_version_filedata = info.filedata;
                }

                try{write_file("../master_"+info.filename, info.projectname,
master_version_filedata);}
                catch(IOException ioe){
                    BUG(ioe);
                }
            }

            //TODO:  This is a hack.  Clean this up.

            debug_disp_file(info.filename + "_version (handle_put_or_remove_resp)",
info.projectname, info.version_filedata);

            classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false, false, new
PutCVSRespPayload(false, "OK", info.filename, info.projectname, info.version_filedata)), 0);
        }
        else{
            System.out.println("Recieved unknown PutOrRemoveResp: " + elem);
        }
    }

    protected void handle_bamboo_route_deliver(BambooRouteDeliver elem){
        System.out.println("Recieved BambooRouteDeliver from: " + elem.src);

        //We're the primary owner of a piece of data, and someone wants it.
        if (elem.payload instanceof GetCVSReqPayload){
            GetCVSReqPayload pay = (GetCVSReqPayload) elem.payload;

            //Destination GUID is set earlier to the GUID of the file we want.
            classifier.dispatch_later(new StorageManager.GetByGuidReq (elem.dest, true, null,
my_sink, new GuidReqInfoContainer(pay.filename, pay.projectname, elem.src, pay.testing)), 0);
        }

        //We're the requesting node, and we'd requested some data (earlier)

        else if (elem.payload instanceof GetCVSRespPayload){
            GetCVSRespPayload pay = (GetCVSRespPayload) elem.payload;

            //If the dest guid is equal to our guid, then this must be
            //a response to a message we sent out earlier.
            if(elem.dest.equals(my_guid)){
                debug_println("Got a BambooRouteDeliver with matching guid!");

                if(pay.success){
                    //output to screen
                    System.out.println("filename: " + pay.filename + " projectname: " +
pay.projectname);
                    System.out.println("  -=-=-=contents-=-=-=");
                    for(int i = 0; i < pay.filedata.length; i++){
                        System.out.print((char)pay.filedata[i]);
                    }
                    System.out.println("-=-=-=end_contents-=-=-=");
                    //Now, write to a file
                    try{
                        byte[] version_header = (((new String(pay.filedata)).split("\\n"))[0] +
"\n").getBytes();
                        byte[] filedata = ((new
String(pay.filedata)).substring(version_header.length)).getBytes();

                        //Open a version file and overwrite whatever already exists.
                        FileOutputStream version_fos = new FileOutputStream(pay.filename +
"_version");

                        version_fos.write(version_header);

                        version_fos.close();

                        //Open a file and overwrite whatever already exists.
                        FileOutputStream fos = new FileOutputStream(pay.filename);

                        //write filedata.length bytes to filesystem
                        //TODO: Does the byte order of the submitting system make a difference
here?  How about Windows<->Linux?
                        fos.write(filedata);

                        //done!
                        fos.close();

                        //Are we testing the system?
                        if(pay.testing){
                            try{
                                //Output a local master
                                write_file("master_"+pay.filename, pay.projectname,
pay.master_filedata);

                                //See if the data is correct
                                Runtime runtime = Runtime.getRuntime();

                                //Changed to now check local master
                                Process proc = runtime.exec("./diff_with_master.pl " +
test_filename);

                                //Note!  Must wait for these processes to exit before continuing!
                                if(proc.waitFor() != 0){
                                    System.err.println("Exit value = " + proc.exitValue());
                                    throw new InterruptedException();
                                }

                                int next_request_time = make_random_int(test_minwait, test_maxwait);
                                Date next_request_date = new Date(((new Date()).getTime()) +
(long)next_request_time);

                                try{append_to_logfile("next_request_date (updater):" +
next_request_date + "\n");}
                                catch(IOException ioe){System.err.println("Error appending to
logfile!");}

                                //will this request occur before the cutoff time?
                                if(next_request_date.before(stopDate)){
                                    //Construct some new data after a random interval has passed.
                                    if(updater_node){
                                        debug_println("sending SubmitTestDataAlarm()");
                                        classifier.dispatch_later(new SubmitTestDataAlarm(), 0);
                                    }
                                    else{
                                        debug_println("queueing RetrieveTestDataAlarm()");
                                        classifier.dispatch_later(new RetrieveTestDataAlarm(),
next_request_time);
                                    }
                                }
                                else{
                                    try{append_to_logfile("stopDate reached.  Ending requests.\n");}
                                    catch(IOException ioe){System.err.println("Error appending to
logfile!");}
                                }
                            }
                            catch(IOException ioe){
                                System.out.println("Error calling diff_with_master.pl!");
                            }
                            catch(InterruptedException ie){
                                System.out.println("diff_with_master.pl had nonzero exit status!");
                            }
                        }
                    }
                    catch (IOException ioe){
                        System.err.println("Error writing retrieved file to filesystem: " + ioe);
                    }
                }
                else{
                    System.out.println("-=-=-=");
                    System.err.println(pay.msg);
                    System.out.println("-=-=-=");
                    try{
                        //Yes, I am getting lazy.
                        Runtime runtime = Runtime.getRuntime();
                        Process proc = runtime.exec("./append_not_found.pl");

                        //Note!  Must wait for these processes to exit before continuing!
                        if(proc.waitFor() != 0){
                            System.err.println("Exit value = " + proc.exitValue());
                            throw new InterruptedException();
                        }
                    }
                    catch(IOException ioe){
                        System.out.println("Error calling ./append_not_found.pl!");
                    }

                    catch(InterruptedException ie){
                        System.out.println("./append_not_found.pl exited with nonzero code!");
                    }
                    debug_println("queueing RetrieveTestDataAlarm()");
                    classifier.dispatch_later(new RetrieveTestDataAlarm(),
make_random_int(test_minwait, test_maxwait));
                }
            }

            //If the destination guid is not equal to ours, then the
            //real requesting node must have died before the response
            //got there and the network redirected it to us.  Ignore.
            else{
                debug_println("Got a BambooRouteDeliver with mismatched guid -- ignoring.");
            }
        }

        //We're the primary owner, and we've been asked to store some
        //data.  This entails grabbing the currently stored version
        //file, using CVS to check if there are any conflicts, and
        //storing it if not, or kicking it back if so.  How I process
        //a StorageManager.GetByGUIDReq needs to handle the bulk of
        //this logic, including the identification of what message is
        //calling it.  Yay user_data.
        else if (elem.payload instanceof PutCVSReqPayload){
            PutCVSReqPayload pay = (PutCVSReqPayload) elem.payload;

            debug_disp_file(pay.filename, pay.projectname, pay.filedata);

            //Destination GUID is set earlier to the GUID of the file we want.
            classifier.dispatch_later(new StorageManager.GetByGuidReq (elem.dest, true, null,
my_sink, new PutCVSReqInfoContainer(pay.filename, pay.projectname, pay.filedata, elem.src)), 0);
        }

        //We're the requesting node, so let's see if our file
        //placement request worked out.
        else if (elem.payload instanceof PutCVSRespPayload){
            //If the dest guid is equal to our guid, then this must be
            //a response to a message we sent out earlier.
            if(elem.dest.equals(my_guid)){
                debug_println("Got a BambooRouteDeliver with matching guid!");

                PutCVSRespPayload pay = (PutCVSRespPayload)elem.payload;

                if(pay.error)
                    System.err.println(pay.message);
                else{
                    System.out.println("Put operation completed successfully");
                    debug_disp_file(pay.filename + "_version", pay.projectname,
pay.version_filedata);

                    try{write_file(pay.filename + "_version", pay.projectname,
pay.version_filedata);}
                    catch(IOException ioe){System.err.println("Error writing version file!");}

                    //if we're testing
                    if(automatic_test){
                        int next_request_time = make_random_int(test_minwait, test_maxwait);
                        Date next_request_date = new Date(((new Date()).getTime()) +
(long)next_request_time);

                        try{append_to_logfile("next_request_date:" + next_request_date + "\n");}
                        catch(IOException ioe){System.err.println("Error appending to logfile!");}

                        if(next_request_date.before(stopDate)){
                            debug_println("queueing new RetrieveTestDataAlarm()");
                            classifier.dispatch_later(new RetrieveTestDataAlarm(),
next_request_time);
                        }
                        else{
                            try{append_to_logfile("stopDate reached.  Ending requests.\n");}
                            catch(IOException ioe){System.err.println("Error appending to
logfile!");}
                        }
                    }
                }
            }

            //Otherwise, this message isn't intended for us.  Ignore!
            else{
                debug_println("Got a BambooRouteDeliver with mismatched guid -- ignoring.");
            }
        }

        //We're the requesting node, and the primary owner has
        //something to tell us.
        else if(elem.payload instanceof StatusPayload){
            StatusPayload pay = (StatusPayload)elem.payload;

            //print it.
            System.out.println(pay.msg);
        }

        else{
            BUG("Payload of unknown type returned in BambooRouteDeliver");
        }
    }

    protected byte[] data_hash (byte[] dh) {
        MessageDigest md = null;
        try { md = MessageDigest.getInstance ("SHA"); } catch (Exception e) {}
        md.update (dh);
        return md.digest ();
    }

    protected byte[] make_hash (String input){
        MessageDigest md = null;
        try {md = MessageDigest.getInstance ("SHA");} catch (Exception e){};
        md.update(input.getBytes());
        return md.digest();
    }

    protected void write_file(String filename, String projectname, byte[] filedata) throws
IOException{
        //Open a file and append ,v to filename.  We want to overwrite whatever already exists.
        FileOutputStream fos = new FileOutputStream(filename);

        //write filedata.length bytes to filesystem
        //TODO: Does the byte order of the submitting system make a difference here?  How about
Windows<->Linux?
        fos.write(filedata);

        //can't hurt
        fos.flush();
        //done!
        fos.close();
    }

    protected byte[] read_file(String filename, String projectname) throws FileNotFoundException,
IOException{
        FileInputStream fis = new FileInputStream(filename);

        //String hashname = projectname + ":" + filename;

        byte[] filedata;

        //CVS doesn't like it when I have a big block of NULL at
        //the end of my files, so I have to only grab the contents
        //of the file.
        int filesize = fis.available();
        if(filesize > MAX_FILESIZE){
            filesize = MAX_FILESIZE;
        }

        filedata = new byte[filesize];

        //read up to filedata.size bytes into the array.
        fis.read(filedata);

        //Don't need it anymore.
        fis.close();

        //return to requestor.
        return filedata;
    }

    //Testing only.  Will handle the first placement of data into the network.
    /*
    protected void handle_initial_data_alarm(){
        try{
            Runtime runtime = Runtime.getRuntime();
            Process proc = runtime.exec("./appendfiledata.pl " + test_filename);

            debug_println("sending PutCVSFileEvent");
            classifier.dispatch_later(new PutCVSFileEvent(test_filename, test_projectname), 0);

            //Queue up a RetrieveTestDataAlarm
            debug_println("queueing RetrieveTestDataAlarm");
            classifier.dispatch_later(new RetrieveTestDataAlarm(), rand.nextInt(test_maxwait));
        }
        catch(IOException ioe){
            System.out.println("Error calling appendfiledata.pl!");
        }
    }
    */

    //Testing only.  Will get data from the network and call a script to
    //check if it's the correct data.
    protected void handle_retrieve_test_data_alarm(){

        debug_println("sending GetCVSFileEvent");
        classifier.dispatch_later(new GetCVSFileEvent(test_filename, test_projectname), 0);

        try{
            Runtime runtime = Runtime.getRuntime();
            Process proc = runtime.exec("./log_retrieve_request.pl");

            //Note!  Must wait for these processes to exit before continuing!
            if(proc.waitFor() != 0){
                System.err.println("Exit value = " + proc.exitValue());
                throw new InterruptedException();
            }
        }

        catch(IOException ioe){
            System.out.println("Error calling log_retrieve_request.pl!");
        }
        catch(InterruptedException ie){
            System.out.println("log_retrieve_request.pl had nonzero exit status!");
        }
    }
    //Testing only.  Will modify a specific file and then place the
    //updated data into the network.
    protected void handle_submit_test_data_alarm(){
        try{
            Runtime runtime = Runtime.getRuntime();
            Process proc = runtime.exec("./appendfiledata.pl " + test_filename);

            if(proc.waitFor() != 0){
                System.err.println("Exit value = " + proc.exitValue());
                throw new InterruptedException();
            }

            debug_println("sending PutCVSFileEvent");
            classifier.dispatch_later(new PutCVSFileEvent(test_filename, test_projectname), 0);

            /*
            debug_println("queueing RetrieveTestDataAlarm");
            classifier.dispatch_later(new RetrieveTestDataAlarm(), rand.nextInt(test_maxwait));
            */
        }
        catch(IOException ioe){
            System.err.println("Error calling appendfiledata.pl!");
        }
        catch(InterruptedException ie){
            System.out.println("appendfiledata.pl had nonzero exit status!");
        }
    }

    //Testing only.  Will start the test routines once the warmup period has ended.
    protected void handle_warmup_completed_alarm() throws EventHandlerException{
        try{
            append_to_logfile("Warmup time: " + test_warmup_time + "\n");
            append_to_logfile("Warmup finished at: " + (new Date()).toString() + "\n");
        }
        catch(IOException ioe){System.err.println("Error appending to logfile!");}

        //TODO: put all of this into one routine (repeated code)
        initialized = true;

        try{append_to_logfile("Node processing messages\n");}
        catch(IOException ioe){System.err.println("Error appending to logfile!");}

        //Clean out the queue that may have built up.
        while(!wait_q.isEmpty()){
            handleEvent((QueueElementIF)wait_q.removeFirst());
        }
        //end repeat

        //Now, start the testing.
        if(initial_data){
            classifier.dispatch_later(new PutCVSFileEvent(test_filename, test_projectname), 0);
        }
        else{
            classifier.dispatch_later(new RetrieveTestDataAlarm(), make_random_int(test_minwait,
test_maxwait));
        }
    }

    protected static void debug_disp_file(String filename, String projectname, byte[] filedata){
        if(DEBUG){
            //output to screen
            System.out.println("filename: " + filename + " projectname: " + projectname);
            System.out.println("  -=-=-=contents-=-=-=");
            for(int i = 0; i < filedata.length; i++){
                System.out.print((char)filedata[i]);
            }
            System.out.println("-=-=-=end_contents-=-=-=");
        }
    }

    protected static void debug_println(String s){
        if(DEBUG)
            System.out.println(s);
    }

    //Append a status message to the node's logfile.
    protected void append_to_logfile(String s) throws IOException{
        if(automatic_test){
            FileOutputStream fos = new FileOutputStream(logfile_name, true);
            String newString = (new Date()).toString() + ": " + s;
            fos.write(newString.getBytes());
            fos.close();
        }
    }

    protected BigInteger make_submit_hashname(String filename, String projectname){
        InetAddress ia = null;
        try{ia = InetAddress.getLocalHost();}
        catch(UnknownHostException uhe){BUG(uhe);}

        File infile = new File(filename);

        String hostIP = ia.getHostAddress();
        String absolutePath = infile.getAbsolutePath();

        return make_hashname(filename, projectname, hostIP, absolutePath);
    }

    //TODO: Convert hashnames to include the ip address of the node
    //that submitted the initial update and the path where the file
    //was initially stored.
    protected BigInteger make_hashname(String filename, String projectname, String
initial_ip_address, String initial_path){
        String hashname = initial_ip_address + ":" + initial_path + ":" + projectname + ":" +
filename;
        SecureHash securehash = new SHA1Hash(digest.digest(hashname.getBytes()));
        BigInteger guid = GuidTools.secure_hash_to_big_integer(securehash);

        return guid;
    }

    protected BigInteger make_hashname(String filename, String projectname){
        String hashname = projectname + ":" + filename;
        SecureHash securehash = new SHA1Hash(digest.digest(hashname.getBytes()));
        BigInteger guid = GuidTools.secure_hash_to_big_integer(securehash);

        return guid;
    }

    protected int make_random_int(int min_int, int max_int){
        System.out.println("max_int: " + max_int + " min_int: " + min_int + " max_int - min_int: "
+ (max_int - min_int));
        return min_int + rand.nextInt(max_int - min_int);
    }

    protected void initGUI(){
        //Project name panel
        JPanel panel = new JPanel(new FlowLayout(FlowLayout.LEFT));
        projectnameLabel = new JLabel("project: ");

        projectnameField = new JTextField(32);

        panel.add(projectnameLabel);
        panel.add(projectnameField);

        //add to main pane
        JPanel mainPane = new JPanel(new BorderLayout());
        mainPane.add(panel, BorderLayout.NORTH);

        panel = new JPanel(new FlowLayout(FlowLayout.LEFT));
        filenameLabel = new JLabel("filename: ");
        ActionAdapter buttonListener = null;

        filenameField = new JTextField(32);

        submitFilenameButton = new JButton("Submit");

        buttonListener = new ActionAdapter()
            {
                public void actionPerformed(ActionEvent e)
                {
                    String filename, projectname;
                    try
```

```
                        {
                            projectname = projectnameField.getText();
                            filename = filenameField.getText();
                            //If we're decvs-ifying the file, add the
                            //CVS suffix.  -- This messes up the hash, don't do it.
                            if(retrieveFile){
                                //filename = filename + ",v";
                                classifier.dispatch_later(new GetCVSFileEvent(filename,
projectname), 0);
                                //do_get(filename, projectname);
                            }
                            else{
                                classifier.dispatch_later(new PutCVSFileEvent(filename,
projectname), 0);
                                //do_put(filename, projectname);
                            }
                            System.out.println("filename: " + filename + "\tprojectname: " +
projectname);
                            //FilenameSubmittedEventAlarm fsea = new
FilenameSubmittedEventAlarm(filename, projectname, retrieveFile);

                            //classifier.dispatch is no good from here, but dispatch_later is fine.
                            //classifier.dispatch_later(fsea, 0);

                        }

                    catch (Exception e2)
                        {
                            System.err.println("e2");
                            mainFrame.repaint();
                        }
                }
            };

        submitFilenameButton.addActionListener(buttonListener);
        submitFilenameButton.setMnemonic('S');
        //submitFilenameButton.setEnabled(!automatic_test);

        panel.add(filenameLabel);
        panel.add(filenameField);
        panel.add(submitFilenameButton);

        //add to main pane
        mainPane.add(panel, BorderLayout.CENTER);

        //do the radio buttons
        putFileRadioButton = new JRadioButton("cvs-ify file", !retrieveFile);

        buttonListener = new ActionAdapter()
            {
                public void actionPerformed(ActionEvent e)
                {
                    retrieveFile = false;
                }
            };

        putFileRadioButton.addActionListener(buttonListener);
        putFileRadioButton.setMnemonic('C');
        //putFileRadioButton.setEnabled(!automatic_test);

        getFileRadioButton = new JRadioButton("decvs-ify file", retrieveFile);

        buttonListener = new ActionAdapter()
            {
                public void actionPerformed(ActionEvent e)
                {
                    retrieveFile = true;
                }
            };

        getFileRadioButton.addActionListener(buttonListener);
        getFileRadioButton.setMnemonic('D');
        //getFileRadioButton.setEnabled(!automatic_test);

        ButtonGroup group = new ButtonGroup();

        group.add(putFileRadioButton);
        group.add(getFileRadioButton);

        //make a new panel
        panel = new JPanel(new FlowLayout(FlowLayout.RIGHT));

        //add buttons to panel
        panel.add(putFileRadioButton);
        panel.add(getFileRadioButton);

        //add to the main pane again
        mainPane.add(panel, BorderLayout.SOUTH);

        /*
          mainPane.add(networkChatterPanel, BorderLayout.CENTER);
          mainPane.add(statusPanel, BorderLayout.WEST);
          mainPane.add(connectionInfoPanel, BorderLayout.SOUTH);
        */

        // Set up the main frame
        String titlestring = "DistCVSInterface at ";
        try{
            titlestring += InetAddress.getLocalHost();
        }
        catch(UnknownHostException uhe){
            titlestring += "unknown host";
        }

        titlestring += ", guid: 0x" + ((my_guid.toString(16)).substring(0,8)).toUpperCase();

        mainFrame = new JFrame(titlestring);
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainFrame.setContentPane(mainPane);
        mainFrame.setSize(mainFrame.getPreferredSize());
        mainFrame.setLocation(200, 200);
        mainFrame.pack();
        mainFrame.setVisible(true);
    }
}
```

Listing 2: PrintDataStage.java

```
/*
 * PrintDataStage.java - a Bamboo stage that displays the GUIDs of
 * data currently stored on the node.
 *
 * Copyright (C) 2004,2005 Andrew Ian Logan
 * logana@bc.edu, andrewlogan@gmail.com
 *
 */

package thesis;

import java.math.BigInteger;

import java.nio.ByteBuffer;

import java.security.MessageDigest;

import java.util.Iterator;
import java.util.Random;
import java.util.LinkedList;

import ostore.util.ByteUtils;
import ostore.util.ByteArrayOutputBuffer;
import ostore.util.Pair;
import ostore.util.CountBuffer;

import seda.sandStorm.api.ConfigDataIF;
import seda.sandStorm.api.QueueElementIF;
import seda.sandStorm.api.SingleThreadedEventHandlerIF;
import seda.sandStorm.api.StagesInitializedSignal;

import bamboo.db.StorageManager;

import bamboo.lss.DustDevil;

import bamboo.util.StandardStage;
import bamboo.util.GuidTools;

import bamboo.dmgr.PutOrRemoveReq;
import bamboo.dmgr.PutOrRemoveResp;

import bamboo.dht.GatewayClient;
import bamboo.dht.bamboo_put_args;
import bamboo.dht.bamboo_get_args;
import bamboo.dht.bamboo_value;
import bamboo.dht.bamboo_secret;
import bamboo.dht.bamboo_key;
import bamboo.dht.bamboo_stat;
import bamboo.dht.bamboo_placemark;
import bamboo.dht.bamboo_get_res;

import java.awt.*;
import java.awt.event.*;

import javax.swing.*;

import java.io.*;

public class PrintDataStage extends StandardStage implements SingleThreadedEventHandlerIF{

    protected boolean initialized = false;
    protected String inputFilename;
    protected boolean retrieveFile = false;

    //Constructor
    public PrintDataStage() throws Exception{
        DEBUG = false;
        event_types = new Class[]{
            StagesInitializedSignal.class,
            PrintDataAlarm.class
        };
    }

    //Get variables from config file, initialize anything else
    public void init (ConfigDataIF config) throws Exception{
        super.init(config);
        int debug_level = config.getInt("debug_level");
        if (debug_level > 0)
            DEBUG = true;
    }

    //Event handler
    public void handleEvent(QueueElementIF item){
        if(logger.isDebugEnabled()) logger.debug("got " + item);

        if (item instanceof StagesInitializedSignal){
            dispatch (new PrintDataAlarm());
        }

        else if (item instanceof PutOrRemoveResp){
        }

        else if (item instanceof PrintDataAlarm){
            print_data();
        }

        else if (item instanceof StorageManager.GetByTimeResp){
            handle_get_by_time_resp((StorageManager.GetByTimeResp) item);
        }

        else if(item instanceof StorageManager.GetByGuidResp){
            handle_get_by_guid_resp((StorageManager.GetByGuidResp) item);
        }

        else{
            throw new IllegalArgumentException (item.getClass().getName());
        }
    }

    protected static class PrintDataAlarm implements QueueElementIF{};

    protected void print_data(){
        //Start with the primary DB.

        dispatch (new StorageManager.GetByTimeReq( 0, Long.MAX_VALUE, my_sink, new LinkedList()));
    }

    protected void handle_get_by_time_resp (StorageManager.GetByTimeResp resp){
        if (resp.continuation == null){
            //Also scan the recycling DB, since we want to make sure
            //that no node is storing tuples that it shouldn't be.

            StorageManager.GetByGuidReq req = new StorageManager.GetByGuidReq (BigInteger.valueOf
(0), false, null, my_sink, resp.user_data);
            dispatch(req);
        }
        else{
            if (resp.key.put){
                LinkedList data = (LinkedList) resp.user_data;
                data.addLast (new Pair(resp.key, null /*resp.data*/));
            }

            dispatch (new StorageManager.GetByTimeCont (resp.continuation, my_sink,
resp.user_data));
        }
    }

    protected void handle_get_by_guid_resp (StorageManager.GetByGuidResp resp){
        if (resp.continuation == null){
            // All done.
            System.err.println ("DistCVSStage-"+ my_node_id + " stored data:");
            LinkedList data = (LinkedList) resp.user_data;
            for (Iterator i = data.iterator(); i.hasNext();){
                Pair p = (Pair) i.next();
                StorageManager.Key k = (StorageManager.Key) p.first;

                System.err.println("  0x" + GuidTools.guid_to_string(k.guid)+ " 0x" +
ByteUtils.print_bytes(k.data_hash, 0, 4));
            }

            classifier.dispatch_later( new PrintDataAlarm(), 10*1000);
        }
        else {
            if (resp.key.put){
                LinkedList data = (LinkedList) resp.user_data;
                data.addLast (new Pair (resp.key, resp.data));
            }

            dispatch(new StorageManager.GetByGuidCont(resp.continuation, false, my_sink,
resp.user_data));
        }
    }
}
```

Listing 3: submithelper.pl

```
#!/usr/bin/perl
# Andrew Logan
# 11/30/04

# Written because making a pile of system calls in Java is a hassle.
# This script assumes that it has been given an old(er) cvs diff file,
# and a newer file to be diffed against it.

# Here is what we need to do:
# Make a CVSROOT
# run "cvs init" in it
# make a repository directory within CVSROOT
# copy the version file into it
# make a temp directory outside of CVSROOT
# check out a copy of the new repository
# overwrite the checked out copy with the updated copy
# check in
# copy the new version file back to where we started


$length = $#ARGV + 1;

if($length < 1){
    print "please provide an input file\n";
    exit(1);
}

$inputfile = @ARGV[0];
$cvsrootdir = "cvsroot";
$cvssuffix = ",v";
$inputversionfile = "$inputfile$cvssuffix";
$tempdir = "tempdir";
$message = "message";
$repository = "repository";
$repositorydir = "$cvsrootdir/$repository";

#where are we?
$here = `pwd`;
chomp($here);

#this is useful later
$cvsrootpath = "$here/$cvsrootdir";

#make the cvs root directory
#mkdir("$cvsrootdir", 0777);

#open the new entries file
if(open (INFILE, "$inputfile\_version")){
    @infilearray = <INFILE>;
    close(INFILE);
    if(open(OUTFILE, "> temp_entries")){
        print OUTFILE @infilearray;
        #add footer to Entries data
        print OUTFILE "D\n";
        close(OUTFILE);
    }
    else{
        print("can't open file for output!\n");
        exit(1);
    }
}
else{
    print("can't open version file!\n");
    exit(1);
}

#set up our CVSROOT (also makes directory)
`cvs -d$cvsrootpath init`;

#make the repository directory
mkdir("$repositorydir", 0777);

#force copy version file there
`cp -f $inputversionfile $repositorydir`;

#make a temp directory
```

```
mkdir("$tempdir", 0777);

#now go to the temp directory
if (!(chdir("$tempdir"))){
    print("failed to change directory.\n");
    cleanup();
    exit(1);
}

#check out a copy of our repository
`cvs -d$cvsrootpath co $repository`;

#go there
#now go to the temp directory
if (!(chdir("$repository"))){
    print("failed to change directory.\n");
    cleanup(); exit(1);
}

#remove the reconstructed file
`rm $inputfile`;

#copy the new version here
`cp $here/$inputfile .`;

#force copy our parsed Entries file over the existing one
`cp -f $here/temp_entries CVS/Entries`;

#check it in and redirect STDERR to a log file
`cvs ci -m$message 2> checkinerrors`;

#did it work?
if(open (INFILE, checkinerrors)){
    @infilearray = <INFILE>;
    close(INFILE);

    foreach $line (@infilearray){
        if($line =~ m/commit aborted/i){
            print("commit failed!\n");
            #cleanup();
            exit(1);
        }
    }
}
else{
    print("can't open checkinerrors.\n");
    cleanup();
    exit(1);
}

#parse the CVS/Entries file
if(open (INFILE, "CVS/Entries")){
    @infilearray = <INFILE>;
    close(INFILE);
    if(open (OUTFILE, "> $here/$inputfile$cvssuffix\_version")){
        foreach $line (@infilearray){
            if($line =~ $inputfile){
                print OUTFILE $line;
            }
        }
        close(OUTFILE);
    }
    else{
        print("can't open file for output!\n");
        exit(1);
    }
}
else{
    print("can't open CVS/Entries!\n");
    exit(1);
}

#now, let's go retrieve the new version file
if (!(chdir("$cvsrootpath/$repository"))){
    print("failed to change to repository directory.\n");
    cleanup();
    exit(1);
}

#force copy cvs' diff of testinput back to where we started
`cp -f $inputfile$cvssuffix $here/`;


#finally, delete everything to prevent old versions from building up
if (!(chdir("$here"))){
    print("failed to change to starting directory.\n");
    cleanup();
    exit(1);
}

sub cleanup(){
    if (!(chdir("$here"))){
        print("failed to change to starting directory.\n");
        exit(1);
    }
    #print "rm -rf $here/$cvsrootdir\n";
    `rm -rf $here/$cvsrootdir`;
    #print "rm -rf $here/$tempdir\n";
    `rm -rf $here/$tempdir`;

     #`rm $here/$inputfile`;
}

#done!
```

Listing 4: reconstructhelper.pl

```
#!/usr/bin/perl
# Andrew Logan
# gethelper.pl
# 10/22/04
#
# Written because making a pile of system calls in Java is a hassle.
# This script will reconstruct a CVS repository, and then extract a
# file from it.  It leaves <filename> in the directory it was
# called from.

# Here is what we need to do:
# Make a CVSROOT
# run "cvs init" in it
# make a repository directory in it
# copy our input ,v files there (CVS only reconstructs files ending in ,v)
# run "cvs co repository"

# TODO: it would be nice to have this file share variables with
# storehelper.pl, since they're two halves of the same thing.

# Revision History:

# 1.3 : 11/19/04
# I now allow for the script to be stored in a directory separate from
# the one that it is being called from.

# $Id #

$length = $#ARGV + 1;

if($length == 0){
    print "please provide an input file to reconstruct\n";
    exit();
}

$inputfile = @ARGV[0];
$cvsrootdir = "cvsroot";
$tempdir = "tempdir";
$repository = "repository";

#where are we?
$here = `pwd`;
chomp($here);
# print "here: $here\n";

#this is useful later
$cvsrootpath = "$here/$cvsrootdir";

#make the cvs root directory
mkdir("$here/$cvsrootdir", 0777);

#set the CVSROOT variable (only for the execution of the script).
#local $ENV{CVSROOT} = "$cvsrootpath";

#automatically inits CVSROOT, but the -d forces it to work in
#CVS-controlled directories
`cvs -d $cvsrootpath init`;

#move into CVSROOT
if (!(chdir("$cvsrootpath"))){
    print("failed to change to repository directory.\n");
    exit();
}

#make the temp repository
mkdir("$repository", 0777);

#force copy the input file there
`cp -f $here/$inputfile $repository/$inputfile`;

#now go back to where we started
if (!(chdir("$here"))){
    print("failed to change directory.\n");
    exit();
}
```

```
#check out the repository
`cvs -d $cvsrootpath co $repository`;

#now, let's go retrieve the file
if (!(chdir("$here/$repository"))){
    print("failed to change to repository directory.\n");
    exit();
}

#copy the undiffed file(s) back to where we started
# FIXME: this won't handle subdirectories.

@filenames = `ls`;

foreach $filename (@filenames){

    #ignore the CVS directory, the rest should be source files.  This
    #is a bit of a hack, since it's actually just matching files with
    #"CVS" at the end of their names.  For now, this is not a big
    #deal.  It will, however, copy everything back on every run.  I
    #should fix that.

    if(!($filename =~ /CVS$/)){
        open(infile, "$filename");
        open(outfile, ">$here/$filename");

        @infilelines = <infile>;

        close(infile);

        foreach $line (@infilelines){
            print outfile $line;
        }

        close(outfile);
    }
}

#I'm not worrying about cleaning up the directories, since it's
#probably faster to not have to create them all the time.

#done!
```

8 Appendix B: Test Scripts

Listing 5: go_big

```
#!/usr/bin/perl

# go_big
# Andrew Logan
# logana@bc.edu, andrewlogan@gmail.com
# 3/14/05

# Written to automate the process of doing multiple tests of the
# DistCVS system.

$hostname = `hostname`;
chomp($hostname);

$start_experiment = 15;
$end_experiment = 15;

for ($i = $start_experiment; $i < $end_experiment+1; $i++){
    if ($i > $start_experiment){
        print "*****WAITING TO BEGIN EXPERIMENT $i*****\n";

        sleep(10 * 60);
    }

    print "*****BEGINNING EXPERIMENT $i*****\n";

    #clear the old results
    if($hostname eq 'cslabgw1'){
        system "./setup_test.sh";
    }
    else{
        #deleting a pile of directories also takes some amount of
        #time... and it's _BAD_ to start again before this has
        #finished.  Wait until master_foo1 is gone to proceed.  This
        #also helps synchronize the rest of the nodes to the gateway.
        #I think it is unlikely that we will overshoot this period and
        #catch the creation of the next one.

        $master_foo1_status = `find -iname master_foo1`;
        chomp($master_foo1_status);

        while($master_foo1_status eq './master_foo1'){
            print "master_foo1 exists\n";
            sleep(1);

            #check every second
            $master_foo1_status = `find -iname master_foo1`;
            chomp($master_foo1_status);
        }
    }

    #start the correct test
    $command_line = "./go_$i";
    system $command_line;

    #generate results
    $command_line = "./make_global_experiment_stats.pl > test_$i\_results";
    system $command_line;

    #store everything for future reference
    if($hostname eq 'cslabgw1'){
        $command_line = "tar cvf auto_experiment_$i\_data.tar cslab*\:*/";
        system $command_line;
    }
}

print "*****AUTOMATIC EXPERIMENTS COMPLETE******\n";

#e-mail!
#`./process_and_send_results.pl`;
```

Listing 6: go

```
#!/bin/bash


# go
# Andrew Logan
# logana@bc.edu, andrewlogan@gmail.com

# Starts a single test of the Bamboo system.


./run_local_nodes.pl -nodes=10 -with_updater -with_gateway
```

Listing 7: run_local_nodes.pl

```
#!/usr/bin/perl

# run_local_nodes.pl
# Andrew Logan
# logana@bc.edu, andrewlogan@gmail.com
# 12/6/04

# This script will start a specified number of bamboo nodes on a local
# machine (with or without a gateway), wait some amount of time for
# things to settle down, and then start churning the network (killing
# nodes and then restarting them after a random amount of time).  This
# script keeps track of the current pids of the client processes, and
# also keeps track of the currently assigned port numbers to prevent
# conflicts.

use Getopt::Long;

#init random number generator
srand();

#get number of nodes to run.
&GetOptions("nodes=i"           => \$num_nodes,
            "with_gateway"      => \$with_gateway,
            "with_updater"      => \$with_updater,
            "with_initial_data" => \$with_initial_data,
            "cfgfile=s"         => \$cfg_filename,
            "kill_percent=f"    => \$kill_percent,
            "no_warmup"         => \$no_warmup,
            "no_run_time"       => \$no_run_time);

if(not defined $num_nodes){
    print "@ARGS[0] -nodes=<int> [-with_gateway] [-with_updater]\n";
    exit;
}

if(defined $cfg_filename){
    $cfg_flag = "-cfgfile $cfg_filename ";
}

if(not defined $kill_percent){
    $kill_percent = .2;
}

print "*****cfg_flag: $cfg_flag****\n";

$hostname = `hostname`;
chomp($hostname);

$port = 20000;
$gatewayport = $port;
$updaterport = $gatewayport;
$livenodes = 0;
$deadnodes = 0;
$logfile = "$hostname:$liveport/diff_results.log";

if($with_updater){
    $updater = "-updater";
}

if($with_initial_data){
    $initial_data = "-initial_data";
}

$before_killed_runtime = 20 * 60;
$after_killed_runtime =  20 * 60;
$after_raised_runtime =  20 * 60;

#seconds to sleep before beginning experiment run.
$stabilize_time = 5 * 60;
#$stabilize_time = 0;

#Java is expecting this in milliseconds
$experiment_runtime = ($before_killed_runtime + $after_killed_runtime + $after_raised_runtime) *
1000;

#give me time to seed data into the network
```

```
#$gateway_pause = 1 * 60;
$gateway_pause = 15;
#$gateway_pause = 0;

#$inter_node_pause = 30;
$inter_node_pause = 15;
#$inter_node_pause = 0;

#in seconds
$killnode_sleep_interval = 10;
$killnode_sleep_interval = 10;


#maximum amount of time to sleep between operation attempts (seconds);
#TODO: take this into account with the node timing
$max_sleep_time = 1 * 60;
#$max_sleep_time = 5;

#random max_sleep_time?
$random_sleep = 0;

#amount of time after the experiment run ends to allow all of the messages in the system to be
dealt with. (in seconds)
$cooldown_time = 2 * 60;

#only display the GUI for cslabgw1:20000 (gateway where the data is seeded)
$only_display_essential_gui = true;

#AutomaticDataSubmissionHack!  Causes the gateway on cslabgw1 to
#automatically submit foo1/foo1 into the network.  This gets rid of
#the last GUI, and allows testing to be entirely autonomous.
#$adsh = true;

if ($only_display_essential_gui){
    $gui = "-nogui";
}

@pidlist;
@liveportlist;
@deadportlist;

#init deadportlist
for($i = 0; $i < $num_nodes; $i++){
    $deadportlist[$i] = -1;
}

#start the gateway node if we want one (as an updater if we want that too)
if($with_gateway){
    $pid = fork();

    #are we the child process?
    if($pid == 0){
        my $date = `date`;
        chomp($date);
        my $hostname = `hostname`;
        chomp($hostname);
        if ($only_display_essential_gui && ($hostname eq 'cslabgw1') && !($adsh)){
            $gui = "";
        }

        #make cslabgw1 automatically submit data into the network.  Note that it is only
appropriate to do so as the network is coming up.
        if($adsh && ($hostname eq 'cslabgw1')){
            $adsh_flag = "-adsh";
        }

        print "****GUI: $gui****\n";

        #Java is expecting this in milliseconds
        if(not defined $no_warmup){
            $test_warmup_time = ((($num_nodes - 2) * $inter_node_pause) + $gateway_pause) * 1000;
        }
        if(not defined $no_run_time){
            $test_run_time = $experiment_runtime + $test_warmup_time + $stabilize_time;
        }
        &init_log_file($port);
        &log_to_live_nodes("\"\#node: $hostname:$port online at $date\"");
        print "child gateway with port: $port\n";
        exec("./parse_cfg_and_run.pl $cfg_flag -port=$port -gateway $updater $initial_data
-warmup_time=$test_warmup_time -run_time=$test_run_time $gui $adsh_flag&");
        exit;
    }
    #we're the parent
    else{
        @liveportlist[$livenodes] = $port;
        $livenodes++;
        $port+=2;
        print "parent: sleeping to allow gateway to establish itself\n";
        sleep($gateway_pause);
        print "parent: awake after gateway sleep\n";
    }
}
else{
    #sleep to let me seed the network
    sleep($gateway_pause);
}

for($i = $livenodes; $i < $num_nodes; $i++){
    my $pid = fork();
    #if child:
    if($pid == 0){
        my $date = `date`;
        chomp($date);
        #Java is expecting this in milliseconds
        if(not defined $no_warmup){
            $test_warmup_time = (($num_nodes - $i - 1) * $inter_node_pause) * 1000;
        }
        if(not defined $no_run_time){
            $test_run_time = $experiment_runtime + $test_warmup_time + $stabilize_time;
        }
        if($port == $updaterport){
            &init_log_file($port);
            &log_to_live_nodes("\"\#node: $hostname:$port online at $date\"");
            print "child updater with port: $port\n";
            exec("./parse_cfg_and_run.pl $cfg_flag -port=$port $updater $initial_data
-warmup_time=$test_warmup_time -run_time=$test_run_time $gui&");
        }
        else{
            #Java is expecting this in milliseconds
            #$test_warmup_time = ((($num_nodes - $i - 1) * $inter_node_pause) + $stabilize_time) *
1000;
            #$test_run_time = $experiment_runtime + $test_warmup_time;
            &init_log_file($port);
            &log_to_live_nodes("\"\#node: $hostname:$port online at $date\"");
            print "child node with port: $port\n";
            exec("./parse_cfg_and_run.pl $cfg_flag -port=$port -warmup_time=$test_warmup_time
-run_time=$test_run_time $gui&");
        }
        exit;
    }
    #if parent
    else{
        @liveportlist[$livenodes] = $port;
        $livenodes++;
        $port+=2;
        sleep($inter_node_pause);
    }
}

#Note!  Array lengths in perl are one less than the conventional idea of length.
print "pidlist: @pidlist\nlength: $#pidlist\n";
print "liveportlist: @liveportlist\nlength: $#liveportlist\n";
print "deadportlist: @deadportlist\nlength: $#deadportlist\n";

#initial sleep to stabilize network

$date = `date`;
chomp($date);
&log_to_live_nodes("\"\#sleeping for $stabilize_time seconds to stabilize network beginning at
$date\"");

print "parent: sleeping to stabilize network\n";
sleep ($stabilize_time);
print "parent: awake - beginning experiment\n";
```

```
sleep ($before_killed_runtime);
print "parent: awake - beginning network churn\n";

#not perfect, but will cut off all of the beginning data.
#print "resetting statistics\n";
#&clean_slate;

$date = `date`;
chomp($date);
&log_to_live_nodes("\"\#network churn beginning at $date\"");

#in seconds
$timeslept = 0;
$raise_nodes = false;

$time_to_sleep = rand($max_sleep_time);
print "parent: going to sleep\n";
#sleep for some amount of time so all nodes don't die at once.
sleep($time_to_sleep);
print "parent: awake\n";

#kill 20% by default
for($i = 0; $i < $kill_percent*$num_nodes; $i++){
    print "parent: calling killnode\n";
    &killnode;
    sleep($killnode_sleep_interval);  #wait a little bit before doing it again.
}

print "parent: going to sleep\n";
#sleep for $max_sleep_time - $time_to_sleep to ensure that all scripts end at roughly the same time
sleep ($max_sleep_time - $time_to_sleep);

#see how the network reacts now.
sleep($after_killed_runtime);
print "parent: awake\n";

if($raise_nodes){
    #raise the nodes we killed above
    for($i = 0; $i < $kill_percent*$num_nodes; $i++){
        print "parent: calling resurrectnode\n";
        &resurrectnode((($kill_percent*$num_nodes) - $i)* $resurrectnode_sleep_interval);
        sleep($resurrectnode_sleep_interval);  #wait a little bit before doing it again.
    }

    print "parent: going to sleep\n";
    #see if the network can figure things out again.
    sleep($after_raised_runtime);
    print "parent: awake\n";
}

print "parent: cooling down experiment\n";
sleep($cooldown_time);

print "parent: terminating experiment\n";

#drastic but quick.
`killall java`;

$date = `date`;
chomp($date);

&log_to_live_nodes("\"\#local experiment terminated at $date\"");


#while($timeslept < $runtime){
#    #pick a random interval to sleep?
#    if($random_sleep){
#       $time_to_sleep = rand($max_sleep_time);
#    }
#    else{
#       $time_to_sleep = $max_sleep_time;
#    }
#    $timeslept += $time_to_sleep;
#    print "parent: going to sleep\n";
#    #sleep for some amount of time to make the network stable.
#    sleep($time_to_sleep);
#    print "parent: awake\n";
```

```
#
#    #flip a weighted coin to kill node
#    if(rand(1) < 1){
#    for($i = 0; $i < 2; $i++){
#       print "parent: calling killnode\n";
#       &killnode;
#    }

#    #flip a weighted coin to resurrect node
#    if(rand(1) < 0){
#       print "parent: calling resurrectnode\n";
#       &resurrectnode;
#    }
#}

sub killnode{
    print "live nodes: $livenodes\n";
    #are there any nodes to kill?
    if($livenodes <= 0){
        print "returning to parent\n";
        return;
    }

    #-1 (in this case) is my signal that the node is already alive
    do{
        $node_to_kill = rand @liveportlist;
    }while($liveportlist[$node_to_kill] == -1);

    @output = `ps aux | grep java | grep $hostname-$liveportlist[$node_to_kill]`;

    $java_pid_line = @output[0];

    #print "java_pid_line: $java_pid_line\n";
    #print "output: @output";

    #I want the second element in this space-delimited string.
    @temparray = split(' ', $java_pid_line);
    $java_pid = @temparray[1];

    print "parent: killing node\n";
    #kill the targeted node
    `kill $java_pid`;
    my $date = `date`;
    chomp $date;

    &log_to_node("\"\#node killed: $date\"", $liveportlist[$node_to_kill]);
    &log_to_live_nodes("\"\#node: $hostname:$liveportlist[$node_to_kill] killed at $date\"");

    $livenodes--;
    $deadnodes++;

    #move the port number to the inactive list
    @deadportlist[$node_to_kill] = @liveportlist[$node_to_kill];
    @liveportlist[$node_to_kill] = -1;
    print "liveportlist: @liveportlist\n";
    print "deadportlist: @deadportlist\n";
    return;
}

sub resurrectnode{
    $resurrectnode_remaining_sleep_interval = $_[0];
    print "dead nodes: $deadnodes\n";
    #are there any dead nodes?
    if($deadnodes <= 0){
        print "returning to parent\n";
        return;
    }

    #-1 is my signal that the node is already dead
    do{
        $node_to_res = rand @deadportlist;
    }while(@deadportlist[$node_to_res] == -1);

    $resurrect_port = @deadportlist[$node_to_res];

    #restart node
    #TODO: figure out gateway issues
    print "parent: resurrecting node\n";
    my $pid = fork();
    if($pid == 0){
        my $date = `date`;
        chomp $date;

        &log_to_node("\"\#node resurrected: $date\"", $resurrect_port);
        &log_to_live_nodes("\"\#node: $hostname:$resurrect_port resurrected at $date\"");

        #TODO: is this a good idea?
        #TODO: you never return from an exec().
        #TODO: add in some more delays here?
        if(not defined $no_run_time){
            $test_run_time = $after_raised_runtime + $resurrectnode_remaining_sleep_interval;
        }
        if($with_gateway && ($resurrect_port == $gatewayport)){
            exec("./parse_cfg_and_run.pl $cfg_flag -port=$resurrect_port -gateway $updater
-run_time=$test_run_time $gui&");
            print "starting gateway ";
        }elsif($with_updater && ($resurrect_port == $updaterport)){
            exec("./parse_cfg_and_run.pl $cfg_flag -port=$resurrect_port $updater
-run_time=$test_run_time $gui&");
            print "starting updater ";
        }else{
            exec("./parse_cfg_and_run.pl $cfg_flag -port=$resurrect_port -run_time=$test_run_time
$gui&");
            print "starting child ";
        }
        print "with port: $resurrect_port\n";

        exit;
    }
    #if parent
    else{
        $livenodes++;
        $deadnodes--;

        #move the port number to the active list
        @liveportlist[$node_to_res] = @deadportlist[$node_to_res];
        @deadportlist[$node_to_res] = -1;
        print "liveportlist: @liveportlist\n";
        print "deadportlist: @deadportlist\n";
        return;
    }
}

#remove accumulated statistics information from before the churn began.
sub clean_slate{
    foreach $liveport (@liveportlist){
        &init_log_file($liveport);
        &log_to_node("\"\#churn beginning\n\"", $liveport);

        #kill the startup parts
        #`rm -rf $hostname:$liveport/diff_results.log`;

        #tag the new one
        #`echo $hostname:$liveport: *begin churn* max_sleep_time: $max_sleep_time random_sleep:
$random_sleep num_nodes: $num_nodes >> $logfile`;
    }
}

sub init_log_file{
    my ($portnum) = @_;
    my $date = `date`;
    chomp($date);
    my $fileheader = "\"\#$hostname:$portnum: max_sleep_time: $max_sleep_time random_sleep:
$random_sleep num_nodes: $num_nodes test_warmup_time: $test_warmup_time test_run_time:
$test_run_time now: $date\"";
    #print $fileheader;
    `rm -rf $hostname:$portnum/diff_results.log`;
    #print "echo $fileheader > $hostname:$portnum/diff_results.log\n";
    `echo $fileheader > $hostname:$portnum/diff_results.log\n`;
}

sub log_to_live_nodes{
    my ($message) = @_;
    foreach $liveport (@liveportlist){
        if($liveport != -1){
            #`echo $message >> $hostname:$liveport/diff_results.log`;
            log_to_node($message , $liveport);
        }
    }
}

sub log_to_node{
    my ($message , $portnum) = @_;
    #print "log_to_node: message: $message portnum: $portnum\n";
    `echo $message >> $hostname:$portnum/diff_results.log`;
}
```

Listing 8: parse_cfg_and_run.pl

```
#!/usr/bin/perl

# parse_cfg_and_run.pl
# Andrew Logan
# logana@bc.edu, andrewlogan@gmail.com
# 12/6/04

# This perl script will parse through a Bamboo configuration file,
# replace the variables within with the correct information, and then
# start a node using that file.  Based off of location-test-menu.pl,
# found in the bamboo distribution.

use Getopt::Long;

#get port to run node on, doesn't count when running a gateway node
$result = GetOptions("cfgfile=s"      => \$cfginfile,
                     "port=i"         => \$port,
                     "gateway"        => \$is_gateway,
                     "to_put=i"       => \$to_put,
                     "updater"        => \$is_updater,
                     "put_size=i"     => \$put_size,
                     "initial_data"   => \$does_initial_data,
                     "warmup_time=i"  => \$warmup_time,
                     "run_time=i"     => \$run_time,
                     "nogui"          => \$hide_gui,
                     "adsh"           => \$do_adsh);

if(not defined $port){
    print "usage: parse_cfg_and_run.pl [-cfgfile <filename>] -port <port> [-gateway] [-to_put
<to_put>] [-put_size <put_size>] [initial_data] [warmup_time <warmup>] [run_time <run>]\n";
    exit;
}

#defaults

if(not defined $cfginfile){
    $cfginfile = "base.cfg";
}

$gateway_port = $port;

$hostname = `hostname`;
chomp($hostname);   #remove trailing newline from `hostname`

#TODO: Make this more general.
if($is_gateway){
    $gateway = "$hostname:$gateway_port";
    $gateway_count = -1;
}
else{
    $gateway = "$hostname:20000";
    $gateway_count = 11;
}

if($is_updater){
    $updater_var = "true";
}
else{
    $updater_var = "false";
}

if($does_initial_data){
    $initial_data_var = "true";
}
else{
    $initial_data_var = "false";
}

if(not defined $to_put){
    $to_put = -1;
}

if(not defined $put_size){
    $put_size = -1;
}
```

```
if($hide_gui){
    $show_gui_var = false;
}
else{
    $show_gui_var = true;
}

if($do_adsh){
    $automatic_data_submission_hack = true;
}
else{
    $automatic_data_submission_hack = false;
}

$data_manager_debug_level = 1;

$dirname = "$hostname:$port";
mkdir($dirname);

$filename_prefix = "$hostname-$port";

my $cfg = "$dirname/$filename_prefix.cfg";
my $log = "$dirname/$filename_prefix.log";
my $blocksdir = "$dirname/$filename_prefix-blocks";

#print "Starting $machine:$port with gateway $gateway.\n";
#print "cfg=$cfg\n";
#print "log=$log\n";

#open (CFG_IN, "location-test-node.cfg.copy")
open (CFG_IN, $cfginfile)
    or die "Could not open $cfginfile";
open (CFG_OUT, ">$cfg") or die "Could not open $cfg";

my %variable_map = ("NodeID"                => "$hostname:$port",
                    "GatewayID"             => $gateway,
                    "GatewayCount"          => $gateway_count,
                    "CacheDir"              => $blocksdir,
                    "GatewayPort"           => $port,
                    "DataManagerDebugLevel" => $$data_manager_debug_level,
                    "ToPut"                 => $to_put,
                    "PutSize"               => $put_size,
                    "IsUpdater"             => $updater_var,
                    "InitialData"           => $initial_data_var,
                    "WarmupTime"            => $warmup_time,
                    "RunTime"               => $run_time,
                    "ShowGUI"               => $show_gui_var,
                    "ADSH"                  => $automatic_data_submission_hack);

while (<CFG_IN>) {
    if (m/\$\{([^}]+)\}/) {
         my $value = $variable_map{$1};
         if (defined $value) {
              s/\$\{([^}]+)\}/$value/;
         }
    }
    print CFG_OUT $_;
}

close (CFG_IN);
close (CFG_OUT);

print "config file: $cfg\n";
print "dirname: $dirname\n";

#set up the links we need
`ln -sf ../storehelper.pl $dirname/storehelper.pl`;
`ln -sf ../submithelper.pl $dirname/submithelper.pl`;
`ln -sf ../reconstructhelper.pl $dirname/reconstructhelper.pl`;
`ln -sf ../diff_with_master.pl $dirname/diff_with_master.pl`;
`ln -sf ../append_not_found.pl $dirname/append_not_found.pl`;
`ln -sf ../log_retrieve_request.pl $dirname/log_retrieve_request.pl`;
`ln -sf ../appendfiledata.pl $dirname/appendfiledata.pl`;

#now go to this node's directory
if (!(chdir("$dirname"))){
    print("failed to change directory.\n");
    cleanup();
    exit(1);
}

#clear the temp files from the experiments
#`rm diff_results.log`;

#-a is append
exec("../modified_run_cfg.pl $filename_prefix.cfg | tee $filename_prefix.log");
```

Listing 9: modified_run_cfg.pl

```
#!/usr/bin/perl

# run_cfg.pl
# Andrew Logan
# logana@bc.edu, andrewlogan@gmail.com
# 10/25/04

# This file will run a bamboo .cfg file passed to it.  Written to save
# me some typing.  Used for automatic testing since I'm running it
# from a directory in.

$length = $#ARGV + 1;

if ($length != 1){
    print "Please enter a configuration file.\n";
    exit(1);
}

$cfgfile = @ARGV[0];

#this line actually starts the node
exec("../../bin/run-java bamboo.lss.DustDevil $cfgfile");
```

Listing 10: base.cfg

```
# $Id: base.cfg,v 1.2 2004/12/15 17:19:26 andrew Exp $

# Sample node configuration file.

<sandstorm>
    <global>
        <initargs>
            node_id ${NodeID}
        </initargs>
    </global>

    <stages>
        <Network>
            class bamboo.network.Network
            <initargs>
                udpcc_debug_level 0
            </initargs>
        </Network>

        <Router>
            class bamboo.router.Router
            <initargs>
                debug_level             0
                #If gateway_count is -1, only the value of gateway is
                # used.  Otherwise, the other gateways are used.  The
                # router code removes our node from the gateway_# list,
                # so we have to do it this way.
                gateway_count           ${GatewayCount}
                gateway_0               cslabgw1:20000
                gateway_1               cslabgw1:20002
                gateway_2               cslabgw1:20004
                gateway_3               cslabgw1:20006
                gateway_4               cslabgw1:20008
                gateway_5               cslabgw1:20010
                gateway_6               cslabgw1:20012
                gateway_7               cslabgw1:20014
                gateway_8               cslabgw1:20016
                gateway_9               cslabgw1:20018
                gateway_10              cslabgw1:20020
                gateway                 ${GatewayID}
                periodic_ping_period    20
                ls_alarm_period         4
                near_rt_alarm_period    0
                far_rt_alarm_period     10
                leaf_set_size           2
                digit_values            2
                ignore_proximity        false
                location_cache_size     0
            </initargs>
        </Router>

        <RouterCallbackInterface>
            class bamboo.router.RouterCallbackInterface
            <initargs>
            </initargs>
        </RouterCallbackInterface>

        <DataManager>
            class bamboo.dmgr.DataManager
            <initargs>
                debug_level            0
                merkle_tree_expansion  2
            </initargs>
        </DataManager>

        <StorageManager>
            class bamboo.db.StorageManager
            <initargs>
                debug_level             0
                homedir                 ${CacheDir}
            </initargs>
        </StorageManager>

        <DistCVSStage>
            class thesis.DistCVSStage
            <initargs>
```

```
                debug_level 0
                logfile_name           diff_results.log
                automatic_test         true
                updater_node           ${IsUpdater}
                initial_data           ${InitialData}
                show_gui               ${ShowGUI}
                automatic_data_submission_hack ${ADSH}
                test_filename          foo1
                test_projectname       foo1
                #15 minutes
                test_maxwait           900000
                #12 minutes
                test_minwait           720000
                mean_put_period_ms     10000
                mean_get_period_ms     10000
                storage_goal           10240
                test_warmup_time       ${WarmupTime}
                test_run_time         ${RunTime}
                client_stage_name  GatewayClient
            </initargs>
        </DistCVSStage>

        <Dht>
            class bamboo.dht.Dht
            <initargs>
                debug_level 0
                storage_manager_stage StorageManager

            </initargs>
        </Dht>

        <Gateway>
            class bamboo.dht.Gateway
            <initargs>
                debug_level 0
                port ${GatewayPort}
            </initargs>
        </Gateway>

        <GatewayClient>
            class bamboo.dht.GatewayClient
            <initargs>
                debug_level 0
                gateway localhost:${GatewayPort}
            </initargs>
        </GatewayClient>

        <PrintDataStage>
             class thesis.PrintDataStage
        </PrintDataStage>

        <Vivaldi>
            class bamboo.vivaldi.Vivaldi
            <initargs>
              vc_type           2.5d
              generate_pings    true
              eavesdrop_pings   false
              use_reverse_ping  true
              ping_period       10000
              version           1
            </initargs>
        </Vivaldi>

        <WebInterface>
            class bamboo.www.WebInterface
            <initargs>
                storage_manager_stage StorageManager
            </initargs>
        </WebInterface>

    </stages>
</sandstorm>
```

Listing 11: make_global_experiment_stats.pl

```
#!/usr/bin/perl
#
# make_global_experiment_stats.pl
# Andrew Logan
# logana@bc.edu, andrewlogan@gmail.com
# 12/13/04
#
# This perl script scans through the logfiles produced by experiment
# runs and outputs some raw statistics.

$num_requests = `grep DATA_REQUEST cslab*\:*/diff_results.log | wc -l`;
chomp($num_requests);

$num_good_version = `grep GOOD_VERSION cslab*\:*/diff_results.log | wc -l`;
chomp($num_good_version);

$num_bad_version = `grep BAD_VERSION cslab*\:*/diff_results.log | wc -l`;
chomp($num_bad_version);

$num_not_found = `grep NOT_FOUND cslab*\:*/diff_results.log | wc -l`;
chomp($num_not_found);

$num_responses = $num_good_version + $num_bad_version + $num_not_found;

$num_dangling = $num_requests - $num_responses;

$percent_responded  = $num_responses    / $num_requests;
$percent_good       = $num_good_version / $num_responses;
$percent_bad        = $num_bad_version  / $num_responses;
$percent_not_found  = $num_not_found    / $num_responses;
$percent_unanswered = $num_dangling     / $num_requests;

print "=====RESULTS=====\n";
print "number of requests:            $num_requests\n";
print "number of responses:           $num_responses\n";
print "number of good versions:       $num_good_version\n";
print "number of bad versions:        $num_bad_version\n";
print "number of not founds:          $num_not_found\n";
print "number of unanswered requests: $num_dangling\n";
print "====PERCENTS=====\n";
printf "percent responded : %.3g\n",$percent_responded;
printf "percent good      : %.3g\n",$percent_good;
printf "percent bad       : %.3g\n",$percent_bad;
printf "percent not found : %.3g\n",$percent_not_found;
printf "percent unanswered: %.3g\n",$percent_unanswered;
```

Listing 12: process_and_send_results.pl

```
#!/usr/bin/perl

# process_and_send_results.pl
# Andrew Logan
# logana@bc.edu, andrewlogan@gmail.com
# 3/15/05

# Written to automagically format and e-mail the results of a long
# night's analysis.

$hostname = `hostname`;
chomp($hostname);

if($hostname eq 'cslabgw1'){

    $outputfile_name = "total_results";

    $header_1 = "\n100 nodes, moderate rate:";
    $header_2 = "\n200 nodes, moderate rate:";
    $header_3 = "\n300 nodes, moderate rate:";
    $header_4 = "\n100 nodes, extreme rate:";
    $header_5 = "\n200 nodes, extreme rate:";
    $header_6 = "\n300 nodes, extreme rate:";
    $header_7 = "\n100 nodes, moderate rate, updates enabled:";
    $header_8 = "\n300 nodes, moderate rate, updates enabled:";

    `echo "This is an automated e-mail message, sent at the completion of the 8 experiment runs.
--Andrew" > $outputfile_name`;

    `echo \"$header_1\" >> $outputfile_name`;
    `cat test_1_results >> $outputfile_name`;

    `echo \"$header_2\" >> $outputfile_name`;
    `cat test_2_results >> $outputfile_name`;

    `echo \"$header_3\" >> $outputfile_name`;
    `cat test_3_results >> $outputfile_name`;

    `echo \"$header_4\" >> $outputfile_name`;
    `cat test_4_results >> $outputfile_name`;

    `echo \"$header_5\" >> $outputfile_name`;
    `cat test_5_results >> $outputfile_name`;

    `echo \"$header_6\" >> $outputfile_name`;
    `cat test_6_results >> $outputfile_name`;

    `echo \"$header_7\" >> $outputfile_name`;
    `cat test_7_results >> $outputfile_name`;

    `echo \"$header_8\" >> $outputfile_name`;
    `cat test_8_results >> $outputfile_name`;

    `mail -s \"Automatic Results Report!\" logana\@bc.edu,signoril\@bc.edu,borowsky\@bc.edu <
$outputfile_name`;
}
else{
    print "skipping mail phase\n";
}
```

9 Appendix C: Bibliography

References

[1] B. Berliner. CVS, The Concurrent Versioning System . http://www.gnu.org/manual/cvs1.9/html_chapter/cvs_2.html.

[2] I. Clarke, T. W. Hong, S. G. Miller, O. Sandberg, and B. Wiley. Protecting free expression online with freenet. *IEEE Internet Computing*, 6(1):40--49, 2002.

[3] P. Druschel and A. Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. In *HotOS VIII*, May 2001.

[4] S. P. Ratnasamy. *A Scalable Content-Addressable Network*. PhD thesis, Dept. Comp. Sci., University of California at Berkeley, 2002.

[5] S. Rhea, D. Oppenheimer, S. Czerwinski, and B. Karp. The Bamboo Distributed Hash Table: A Robust, Open-Source DHT. http://www.bamboo-dht.org, 2004.

[6] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329--350, 2001.

[7] A. Sotira. Step 1: What is Gnutella? http://www.gnutella.com/news/4210, December 2001.

[8] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 149--160. ACM Press, 2001.

[9] U.S. Department of Commerce: Economics and Statistics Administration: U.S. Census Bureau. Home Computers

and Internet Use in the United States: August 2000.
http://www.census.gov/prod/2001pubs/p23-207.pdf, 2001.