Suffield Academy Network Documentation

Jason Healy, Director of Networks and Systems

Last Updated Mar 18, 2008 (Revision 1083)

This document reflects the ongoing work at Suffield Academy to maintain complete and up-to-date documentation describing the setup and maintenance of the campus network, servers, and software.

This documentation primarily serves as a written record of the knowledge and experience of the Network Administrator. It came about to help solve the "hit by a bus" scenario, where the transfer of knowledge from the Network Administrator to other staff members is necessary. Therefore, the documentation is primarily written about the application of technology at Suffield Academy.

That said, much of the information is generic enough in nature that we felt it would be beneficial to put it online to share with the general community. We hope that others may find useful information in this documentation, and we welcome comments and suggestions.

As of this writing, the document is a work in progress. As we add new sections, we shall make them available online at the following address:

http://web.suffieldacademy.org/ils/netadmin/docs/

That page will always have the most recent version of the documentation, along with a date of publication to distinguish different versions.

Contents

I HOWTOs **5**

1 Building NetRestore Images **9**

1.1 Introduction 9

1.1.1 Prerequisites 9

1.2 Preparing the Operating System 10

1.2.1 Initial Setup 10

1.2.2 Boot Images 10

1.2.3 Software Updates 11

1.2.4 System Preferences 11

1.2.5 Certificate Trust 12

1.3 Installing Applications 12

1.3.1 Common Applications 12

1.3.2 Network Workstation Software 14

1.3.3 Printers 15

1.3.4 Faculty Applications 15

1.4 Final Preparations 16

1.4.1 Customize the Dock 16

1.4.2 Hard Drive Cleanup 16

1.4.3 Reset Safari 16

1.4.4 User Profile Customization 16

1.4.5 Deleting All Users 18
1.4.6 Networked Users 18
1.4.7 Local Admin Group 19
1.4.8 Printer Group Changes 19
1.5 Building the Image 19
1.6 NetRestore Configuration 20
1.6.1 Updating an Existing Image 20
1.6.2 Adding a New Image 20
1.7 Resources 21

2 Installing Windows XP **23**

2.1 Introduction 23
2.2 Installation 23
2.2.1 Before You Start 23
2.2.2 BIOS Settings 24
2.2.3 Booting 24
2.2.4 Partitioning 25
2.2.5 Initializing the Disk 27
2.2.6 Waiting for Installation 29
2.2.7 Regional and Language Options 31
2.2.8 Personalize Your Software 32
2.2.9 Product Activation Key 33
2.2.10 Computer Name and Administrator Password 33
2.2.11 Date and Time Settings 34
2.2.12 Networking Settings 34
2.2.13 Workgroup or Computer Domain 35
2.2.14 Finishing Initialization 36
2.2.15 Network Identification Wizard 36

2.2.16 Logging In 38
2.3 Setting Up the OS 39
2.3.1 Eject the Installation CD 39
2.3.2 Display Settings 39
2.3.3 Control Panel Settings 41
2.3.4 Install Pending Updates 46
2.4 Installing Standard Applications 47
2.4.1 Connecting to the Fileserver 47
2.4.2 First Class 49
2.4.3 Mozilla Firefox 50
2.4.4 Mozilla Sunbird 52
2.4.5 Microsoft Office 2003 53
2.4.6 FileMaker 6 53
2.4.7 Acrobat Reader 7 53
2.4.8 Ultr@VNC Server 53
2.4.9 Sophos Anti-Virus 55
2.4.10 Printing 55
2.4.11 Macromedia Studio 8 (optional) 55

3 Installing ODIN **57**
3.1 Introduction 57
3.2 Prerequisites 57
3.3 Installation 58
3.3.1 Logging In to the Server 58
3.3.2 Installing Odin 60

4 Installing ODIN **69**
4.1 Introduction 69

4.2 Mac OS X 69
4.3 Windows XP 70
4.3.1 Configuring the Client 70
4.3.2 Using the VPN 74

II Software **77**

5 Netboot Services **81**
5.1 Introduction 81
5.2 Configuring Netboot Services 82
5.2.1 Initial Setup 82
5.2.2 AFP Settings 82
5.2.3 NFS Settings 82
5.2.4 NetBoot Settings 83
5.2.5 Network Settings 83
5.3 Building a Rescue Image 84
5.3.1 Selecting a Machine 84
5.3.2 Base Installation 84
5.3.3 System Configuration 86
5.3.4 System Preferences 89
5.3.5 Software Installation 91
5.3.6 Performance Tweaks 93
5.3.7 Building the Image 94
5.3.8 Installing the Image 94
5.3.9 Testing the Image 94

6 OS 9 Remote Shutdown **97**
6.1 Introduction 97

6.2 Requirements . . . 97
6.3 Configuring the Target . . . 98
6.3.1 Installation of Scripts . . . 98
6.3.2 Receive AppleEvents . . . 99
6.4 Configuring the Source . . . 100
6.5 Configuring NUT as a Trigger . . . 100
6.5.1 Setting Up NUT . . . 100
6.5.2 Sample Configuration Files . . . 101
6.5.3 Creating a Remote Shutdown Script . . . 102
6.5.4 Calling the Remote Shutdown Script . . . 103

7 FirstClass Scripts **105**

7.1 Introduction . . . 105
7.2 Preflight Scripts . . . 105
7.2.1 The Preflight Scripts . . . 106
7.3 Postflight Scripts . . . 107
7.3.1 The Postflight Scripts . . . 107
7.4 FirstClass Network Store Backup . . . 108
7.4.1 Background . . . 108
7.4.2 Requirements . . . 108
7.4.3 Design . . . 108
7.4.4 Usage . . . 109
7.4.5 Recovering from Zombie Processes . . . 111
7.4.6 Restoration . . . 112
7.5 Internet Services Log Rotate . . . 113
7.5.1 Background . . . 113
7.5.2 Requirements . . . 113
7.5.3 Design . . . 113

7.5.4 Usage . . . 114
7.5.5 Viewing Old Logs . . . 114

8 DNS **115**

8.1 Introduction . . . 115
8.1.1 Suffield's Topology . . . 116
8.2 Master (Adonis) Configuration . . . 116
8.2.1 Adonis Notes . . . 117
8.2.2 Updating the Zones . . . 117
8.3 Slave Server Configuration . . . 118
8.3.1 Master Configuration . . . 118
8.3.2 Slave Configuration . . . 119
8.4 DNS with Mac OS X Server . . . 120
8.4.1 Custom Changes . . . 121
8.4.2 Configuration Files . . . 121
8.5 Suffield DNS Config Template . . . 121
8.5.1 Differences . . . 121
8.5.2 Using the Configuration . . . 122
8.6 Disaster Recovery . . . 124

9 DHCP **125**

9.1 Introduction . . . 125
9.2 Mac OS X Server Setup . . . 125
9.2.1 Getting dhcpd from DarwinPorts . . . 126
9.3 Suffield DNS Config Template . . . 126
9.3.1 Differences . . . 127
9.3.2 Using the Configuration . . . 127
9.4 Using DHCP . . . 129

9.4.1 Normal Operation Indicators . . . 129
9.4.2 Finding a Client's Address . . . 130
9.4.3 Starting the Daemon . . . 130
9.4.4 Stoping the Daemon . . . 130
9.4.5 Loading Configuration Changes . . . 130
9.4.6 Adding Fixed Hosts . . . 131
9.4.7 Adding Registered Hosts . . . 132
9.4.8 Failover . . . 132

10 Printing **135**

10.1 Introduction . . . 135
10.2 Printing Overview . . . 136
10.2.1 Printer Hardware . . . 136
10.2.2 Client Machines . . . 136
10.2.3 Older Clients . . . 137
10.3 DNS Service Discovery (Bonjour) . . . 137
10.3.1 Publishing a Service Zone . . . 138
10.3.2 Publishing Service Records . . . 138
10.3.3 Final Notes . . . 140
10.4 CUPS Configuration . . . 140
10.5 DHCP Configuration . . . 140
10.6 TFTP Configuration of Printers . . . 141
10.7 Our Helper Script . . . 142

11 Syslog (Centralized Logging and Analysis) **145**

11.1 Introduction . . . 145
11.1.1 Logging Software . . . 145
11.1.2 Analysis Software . . . 146

11.1.3 Intended Audience . . . 146
11.2 syslog-ng . . . 146
11.2.1 General Concepts . . . 147
11.2.2 Installation . . . 147
11.2.3 Configuration . . . 147
11.3 Syslog Client Configuration . . . 151
11.4 Logcheck . . . 151
11.4.1 Obtaining the Software . . . 152
11.4.2 Usage . . . 152
11.4.3 Syslog-NG and Logcheck . . . 152
11.4.4 Basic Configuration . . . 153
11.4.5 Custom Rules . . . 154

12 OpenDirectory (LDAP/Kerberos) 157

13 Name of Project Here 159
13.1 Introduction . . . 159
13.2 Initial Setup . . . 159
13.2.1 Open Directory Integration . . . 160
13.2.2 Joining Kerberos . . . 161
13.3 AFP Sharepoints (Macintosh Clients) . . . 161
13.3.1 Server Admin Settings . . . 162
13.3.2 Workgroup Manager Settings . . . 162
13.4 CIFS Sharepoints (Windows Clients) . . . 162
13.4.1 Server Admin Settings . . . 163
13.4.2 Workgroup Manager Settings . . . 163
13.5 Macintosh Network Home Directories . . . 164
13.5.1 Enabling Local Home Directories . . . 164

13.6 Windows Roaming Profiles . . . 165
13.7 NFS Exports . . . 165
13.7.1 Server Admin Settings . . . 165
13.7.2 Workgroup Manager Settings . . . 166

14 Squid (WWW Proxy Server) 167

14.1 Introduction . . . 167
14.2 Suffield's Requirements . . . 167
14.3 Hardware . . . 168
14.4 Custom Compilation . . . 169
14.4.1 Preparing The System . . . 169
14.4.2 Kernel Compilation . . . 170
14.4.3 IPTables Compilation . . . 171
14.4.4 Squid Compilation . . . 173
14.5 Configuration . . . 175
14.5.1 Bridge Setup . . . 175
14.5.2 Disk Setup . . . 176
14.5.3 Squid Configuration . . . 177
14.5.4 TPROXY Rules . . . 177
14.5.5 Watching Squid . . . 179
14.5.6 Initialization Scripts . . . 180
14.6 Cache Manager . . . 181
14.6.1 Configuration . . . 181
14.6.2 Security . . . 181
14.7 Squid Statistics . . . 181
14.7.1 Downloading and Installing Calamaris . . . 182
14.7.2 Configuring Calamaris . . . 182
14.7.3 Running Calamaris . . . 183

15 Network UPS Tools **185**

15.1 Introduction . . . 185

15.2 Suffield's Setup . . . 185

15.3 Server Setup . . . 186

15.3.1 Install the Software . . . 186

15.3.2 Port Identification . . . 186

15.3.3 Configuring the UPSes . . . 187

15.3.4 Configuration Access Controls . . . 187

15.3.5 Configuring Monitoring . . . 189

15.3.6 Enable NUT . . . 192

15.3.7 NUT CGI . . . 192

15.4 Compiling NUT . . . 192

15.4.1 Getting the Sources . . . 193

15.4.2 Add a User . . . 193

15.4.3 Build the Software . . . 193

15.4.4 Configure the Software . . . 194

15.4.5 Launch the Software . . . 194

15.5 Replacing Batteries . . . 195

15.5.1 When to Replace? . . . 195

15.5.2 Replacing the Battery . . . 195

15.5.3 Updating the Battery Date Variables . . . 195

15.5.4 Refreshing UPS Battery Data . . . 195

16 Incremental Backup Script **199**

16.1 Introduction . . . 199

16.2 Design Issues . . . 200

16.2.1 HFS Metadata . . . 201

16.2.2 Directory Services Integration . . . 201

16.2.3 Remote Backups . . . 201
16.3 Script Usage . . . 202
16.3.1 Organization . . . 203
16.3.2 Configuration Files . . . 204
16.4 Recovery . . . 207

17 Subversion **209**
17.1 Introduction . . . 209
17.2 Mac OS X Client Setup . . . 209
17.2.1 Precompiled Versions . . . 209
17.2.2 Fink . . . 210
17.2.3 DarwinPorts . . . 210

18 PostgreSQL **213**
18.1 Introduction . . . 213
18.2 Installation . . . 214
18.2.1 Debian Linux . . . 214
18.2.2 Mac OS X . . . 215
18.3 Configuration . . . 219
18.3.1 Schemas . . . 220
18.3.2 Tablespaces . . . 221
18.3.3 Host-based Authentication . . . 221
18.3.4 Allowing Remote Access . . . 222
18.3.5 Autovacuum . . . 223
18.3.6 Logging . . . 223
18.3.7 Tweaking Kernel Memory Parameters . . . 223
18.3.8 PostgreSQL Shared Memory Parameters . . . 224
18.3.9 Speeding Up Disk Access . . . 225

18.4 PostgreSQL Usage . . . 226
18.4.1 Server Management . . . 226
18.4.2 Manual Interaction . . . 227
18.4.3 Programatic Access . . . 228
18.5 Backing Up . . . 229
18.5.1 Suffield Overview . . . 229
18.5.2 The “postgresql_archive“ Script . . . 230
18.6 Restoring From Backup . . . 231
18.6.1 Point-In-Time Recovery (PITR) . . . 231
18.6.2 SQL Dump Recovery . . . 233

19 Host-based Firewalls with IPTables 237
19.1 Introduction . . . 237
19.2 Concepts . . . 238
19.2.1 The Suffield Firewall Scripts . . . 238
19.3 Installation of the Scripts . . . 239
19.3.1 File Overview . . . 239
19.3.2 Installing the Scripts . . . 239
19.3.3 Configuration . . . 239
19.3.4 Configuration Syntax . . . 240
19.3.5 Variable Types . . . 240
19.4 Protocol Rules . . . 241
19.5 Usage . . . 244

20 Exile 245
20.1 Introduction . . . 245
20.1.1 Synopsis . . . 245
20.2 History . . . 247

20.3 Design . . . 248
20.3.1 Requirements . . . 249
20.4 Theory of Operation . . . 250
20.5 Installing OpenBSD . . . 251
20.5.1 Pre-Installation Checklist . . . 252
20.5.2 OpenBSD Installation . . . 252
20.5.3 Basic System Configuration . . . 255
20.5.4 Perl Libraries . . . 257
20.5.5 Wrapping Up . . . 257
20.6 Installation . . . 257
20.6.1 Downloading . . . 257
20.6.2 Configuring . . . 258
20.6.3 Running . . . 259
20.7 Performance Graphing . . . 262
20.7.1 Warning . . . 262
20.7.2 Overview . . . 262
20.7.3 Installation . . . 262
20.7.4 Customization . . . 263

21 Failover and High Availability **265**

21.1 Introduction . . . 265
21.2 Terms and Definitions . . . 265
21.3 Mac OS X IPFailover . . . 266
21.3.1 Configuring Failover . . . 267
21.3.2 Failover Transition Scripting . . . 269
21.3.3 Peering Failover . . . 270
21.3.4 Using the Suffield Configs . . . 270

22 NFS (Network File System) 271

22.1 Introduction . . . 271

22.2 NFS Exports . . . 271

22.2.1 Adding an Export . . . 272

22.3 NFS Clients . . . 272

23 AimSniff 273

23.1 Introduction . . . 273

23.2 Dependencies . . . 273

23.2.1 Network Setup . . . 274

23.2.2 AimSniff Dependencies . . . 274

23.2.3 Testing It Out . . . 276

23.3 Database Setup . . . 277

23.3.1 Selecting a DB Server . . . 277

23.3.2 Using MySQL as the Backend . . . 278

23.3.3 Using PostgreSQL as the Backend . . . 279

23.4 Configuring AimSniff . . . 281

23.4.1 AimSniff Configuration File . . . 281

23.4.2 AimSniff Startup Script . . . 281

23.4.3 Configuring Network Equipment . . . 281

23.4.4 Testing . . . 282

24 SMDR (Phone Switch Call Accounting) 283

24.1 Introduction . . . 283

24.2 Overview . . . 284

24.2.1 Basic Operation . . . 284

24.2.2 Requirements . . . 284

24.3 Running the Code . . . 285

24.3.1 Setting Up the Database . . . 285
24.3.2 Setting the Script to Run . . . 286

25 Dance-A-Thon Scripts **289**

25.1 Introduction . . . 289
25.2 Dance-A-Thon Overview . . . 289
25.2.1 Hardware . . . 290
25.2.2 Software . . . 292
25.3 Usage . . . 293
25.3.1 Hardware Preparation . . . 293
25.3.2 Software Preparation . . . 294
25.3.3 Dance-A-Thon Preparation . . . 295
25.3.4 After the Dance-A-Thon . . . 295

III Servers **297**

26 Firewall (OpenBSD PF) **301**

26.1 Introduction . . . 301
26.1.1 Network Diagram . . . 301
26.2 Installation . . . 303
26.2.1 Hardware Requirements . . . 303
26.2.2 Install OpenBSD . . . 304
26.2.3 Installing Packages . . . 304
26.2.4 OS Tweaks . . . 305
26.2.5 Network Configuration . . . 306
26.2.6 Finishing Up . . . 307
26.3 Firewall Settings . . . 307
26.3.1 Normalization . . . 307

26.3.2 Policy queueing using tags . . . 308
26.3.3 Static NAT for servers . . . 308
26.3.4 Round-robin NAT for users . . . 309
26.4 Reloading the Firewall Ruleset . . . 309
26.5 Tips and Tricks . . . 309
26.5.1 pftop . . . 310
26.5.2 tcpdump . . . 310
26.5.3 Label Statistics . . . 310
26.5.4 Inspecting PF Config . . . 311
26.5.5 Killing States . . . 311

27 NEAX-2400 (PBX / Phone System) 313

27.1 Introduction . . . 313
27.1.1 Network Setup . . . 313
27.1.2 PBX Setup . . . 315
27.1.3 System Interfaces . . . 317
27.2 The Frame . . . 318
27.2.1 Server Room . . . 318
27.2.2 Connecting Station Equipment . . . 318
27.3 Programming . . . 319
27.3.1 Terminology . . . 320
27.3.2 Commands . . . 322
27.4 Voicemail . . . 330
27.5 User Commands . . . 330
27.5.1 Forwarding Calls (all phones) . . . 330
27.5.2 Voicemail (all phones) . . . 331
27.5.3 Telephone Setup Functions (digital phones only) . . . 331
27.5.4 System Features . . . 332

28 Catalyst 3550 Common **335**

28.1 Introduction . . . 335

28.1.1 Intended Audience . . . 335

28.1.2 Shortcut to Configs . . . 336

28.2 Banner (MOTD and Login) . . . 336

28.2.1 Suffield Config Files . . . 336

28.2.2 Setting the MOTD Banner . . . 336

28.2.3 Setting the Login Banner . . . 337

28.2.4 Testing It Out . . . 338

28.3 DNS . . . 338

28.3.1 Suffield Config Files . . . 338

28.3.2 Setting the Hostname . . . 338

28.3.3 Setting the DNS Name Servers . . . 339

28.3.4 Setting the Domain Suffix Search . . . 339

28.3.5 Testing It Out . . . 340

28.4 NTP . . . 340

28.4.1 Suffield Config Files . . . 341

28.4.2 Setting the Clock Options . . . 341

28.4.3 Setting the NTP Servers . . . 342

28.4.4 Testing It Out . . . 342

28.5 Kerberos . . . 343

28.5.1 Suffield Config Files . . . 343

28.6 Syslog . . . 343

28.6.1 Suffield Config Files . . . 343

28.6.2 Basic Logging Setup . . . 343

28.6.3 Local Logging Options . . . 344

28.6.4 Remote Syslog Options . . . 345

28.6.5 Testing It Out . . . 345

28.7 Edge Filtering 346
28.7.1 Suffield Config Files 346
28.7.2 Types of Unwanted Traffic 346
28.7.3 Quick Background: ACLs and VLAN Access Maps 347
28.7.4 ACLs 347
28.7.5 VLAN Access Maps 352
28.7.6 Applying Access Maps and Testing 353
28.8 Switch Installation / Replacement 353
28.8.1 Hardware Setup 353
28.8.2 Initial Configuration 354
28.8.3 VLAN Configuration 354
28.8.4 Upgrading the IOS Software 355
28.8.5 Restoring the Configuration 357

IV Diagrams **359**

Part I

HOWTOs

This section contains documentation of general procedures that do not apply to any particular computer. Examples include: basic configuration of client machines, disaster recovery, emergency shutdowns, and short user guides for frequently asked questions.

Chapter 1

Building NetRestore Images

Last updated 2008/03/18

1.1 Introduction

We repair and set up hundreds of machines each year at Suffield Academy. To help us with this process, we have set up several pre-made **system images** that we use to reformat computers. Each image contains an operating system and common applications.

This document describes how to build a new image for computers running Mac OS X. It is intended for someone who has experience installing applications on Macs, and who has a general familiarity with the applications used at Suffield. No system administration experience is necessary.

Note: this document describes how to create images that will be installed on user machines (erasing and replacing whatever is there). We also use a special *rescue image* for booting machines and running diagnostics. If you need to know how to create a rescue image, please see our NetBoot documentation on rescue image creation.

1.1.1 Prerequisites

When building an image, always start with a clean machine. If possible, restore the computer using the restore CDs (or DVD) that came with it. Otherwise, erase the hard drive and perform a full install of the operating system from installation media.

Use the newest model of computer available when you build your image. Try to find one with as many "extra" features in it (such as DVD burners, large screens, etc). Images built on the best machines tend to work well on all other machines. Images built on "average" machines, however, do not tend to work well on better machines.

Try to build images for "classes" of machines. For example, if you're building an image that will primarily be used for laptop computers, build it on the best laptop you can find. If you're building an image for a group of machines in a lab that are all the same, choose the best machine from the lab. And if you're building an image for desktop machines, choose the best desktop machine available.

1.2 Preparing the Operating System

1.2.1 Initial Setup

We're assuming that you're starting with a machine that has a fresh install of the OS on it. The Apple Registration program should launch on start, and ask you the basic configuration questions.

Register the computer to `Suffield Academy`, and provide the school's address and phone number for the registration form.

When asked to create a user account, use `Suffield Academy` as the name, and `suffieldacademy` as the short name. Use `1833` as the password if this is a personal machine (*e.g.*, a laptop), or the proper master password for shared machines. If you do not know which password to use, consult the Network Administrator.

Continue through the rest of the setup program, setting up the network, date and time, and other settings.

When you're done with the registration program, the computer should boot to a default desktop, and be ready to use.

1.2.2 Boot Images

Suffield uses a custom background picture so we can tell which computers have been restored with our system image. These images reside on our file server. To get them, connect to the fileserver and mount the **Groups** partition. Then, open the **Tech Repair** folder. There is a folder called **Boot Pictures** that contains the pictures we need.

The **Desktop** folder contains a series of background images for use on the machines. Choose the one for the type of machine you're setting up. In general, `Laptop Aqua Blue.jpg` is used for personal machines, and `Suffield Aqua Blue.jpg` is used for machines that the school owns (such as office machines). Select the correct image and copy it to your desktop.

This copies the default background. Open Terminal and type the following commands (when prompted, enter your administrative password):

```
cd /System/Library/CoreServices/
sudo cp DefaultDesktop.jpg DefaultDesktop_original.jpg
sudo cp ~/Desktop/Laptop\ Aqua\ Blue.jpg DefaultDesktop.jpg
```

When you're done, you may quit `Terminal.app` and remove the images from your desktop.

1.2.3 Software Updates

Run Software Updates on the computer until there are no updates left to run. This may require multiple installations and reboots.

1.2.4 System Preferences

Now enter the System Preferences application. For each preference pane listed below, follow the instructions given.

Sharing

In the **Computer Name** field at the top of the screen, enter `unregistered`.

Date & Time

Find the **Set Date & Time Automatically** checkbox, and make sure it is *selected*. In the input box, type `ntp.suffieldacademy.org`.

Click on the **Time Zone** tab and confirm that the computer's time zone is set to an appropriate time zone (*e.g.*, **Boston**).

Network

We need to teach the laptop about the wireless networks available on campus. Click on the **AirPort** icon, and then click on the **Advanced** button.

Under **Preferred Networks**, click the plus sign to add a network. Choose "Suffield Auth" as the network. Do not enter any name or password; just add the network.

1.2.5 Certificate Trust

Connect to the **Installers** and move into the **Suffield Installers** folder. Double-click the *Suffield Academy Network Certificates* file. Enter your administrator password if prompted.

1.3 Installing Applications

Suffield has licenses for several commonly-used applications. You must install these applications before building the image so that they are immediately available when a computer is re-imaged.

1.3.1 Common Applications

The applications in this section should be installed on all computers at Suffield Academy. We have unlimited licenses for them, and they are used by nearly everyone on campus.

FirstClass 9

Install the FirstClass 9 client from the file server (you can find it in the **Suffield Installers** folder).

Once installed, FirstClass automatically launches. Quit the program immediately.

Run the "Install FirstClass Settings" script from the same folder.

Re-start FirstClass and ensure that the settings are now correct (*e.g.*, the server listed is `fc.suffieldacademy.org`).

Add the FirstClass client to the dock.

Sophos Anti-Virus Install

Run the Sophos Anti-Virus Installer off of the file server (you can find it in the **Suffield Installers** folder).

Microsoft Office

Run the Microsoft Office installer from the file server (you can find it in the **Suffield Installers** folder). When prompted to register the program, use `Suffield Academy` as the name, and leave all other fields blank. Click **Install** to complete the installation.

After installation, Office will run its auto-update tool. Install any pending updates before quitting.

iWork '09

Install from the server, and register with the given serial number.

iLife '09

Install from the server if iLife was not included as part of the default OS install.

Adobe Creative Suite Design Premium

Run the Adobe installer from the file server (you can find it in the **Suffield Installers** folder). Perform a default install of the application, but **skip** the "Version Cue" server.

When installation is complete, launch **Photoshop** and complete the product registration process (enter the serial number if necessary).

Run the Adobe Updater (in **Utilities**) and all updates (may need to run multiple times to get all updates).

Launch Acrobat, and make sure to say "don't ask again" when prompted for file-opening preferences.

For multimedia lab machines, download and install the Canon CanoScan drivers for our scanners (see the **Scanners** folder on the server).

Fetch (FTP Client)

Install from the **Network** folder on the server. Register with the name "Suffield Academy" and the serial number in the name of the disk image.

FireFox

Download the latest off the web and install.

Video Players

On the file server, in the **Multimedia** folder, find and open the folder called **Video**.

Install **VLC**.

Install **Perian** by double-clicking the Perian prefPane. Choose **Install for all users of this computer**. When the preference pane loads, *uncheck* **Automatically Update**.

Fonts

On the file server, in the **Multimedia** folder, find and open the folder called **Fonts**. Copy the contents of this folder into the `/Library/Fonts/` folder on the computer's hard drive.

1.3.2 Network Workstation Software

The following software should be installed on machines that are owned by Suffield (network workstations)

Remote Desktop

In the **Network** folder, install the **Suffield Remote Desktop 3** package. This allows us to connect to computers and remotely manage them.

Network Settings

Networked machines should force the user to authenticate before they can use the machine.

Open **System Preferences** and click on **Accounts**.

Click the lock to make changes (if necessary) and authenticate.

Click the **Login Options** button.

In the preference pane that appears, make sure **Automatically log in as:** is **deselected**. Also, set **Display login window as:** is set to **Name and password**. Finally, make sure **Enable fast user switching** is **deselected**.

1.3.3 Printers

On the **Installers** section of the file server, open the **Printing** folder.

Canon Copier Software

Run the **PS Installer** in the **Faculty Room Copier** folder.

Run the **UFR II Installer** in the **Library Copier** folder.

1.3.4 Faculty Applications

The following applications should only be installed on computers for Faculty and Staff.

FileMaker 9

Open the **Faculty and Staff** folder on the file server. Then find and open the **FileMaker** folder.

Run the FileMaker Pro 9 installer. If asked to register, choose **Already Registered**.

Once installation is complete, you'll need to copy a few more files onto the computer. Copy one or more of the FileMaker launch scripts (*e.g.*, `Open-o-Rama` or `Portal`) onto the desktop.

GradeKeeper

Open the **Faculty and Staff** folder on the file server. Then find and open the **Gradekeeper** folder.

Run the installer with the default options. When the installer is done, launch the application.

The application will prompt you to register the product. Click the **Enter Code** button. The registration information is contained in a text file in the same folder as the application.

Once the application is registered, you can quit it.

1.4 Final Preparations

Before building an image out of this computer, we need to make sure and "tidy up" any other aspects of the system.

1.4.1 Customize the Dock

Make sure the dock has all of our standard applications on it. You may wish to remove unused applications (such as Mail and Address Book) to create more space.

1.4.2 Hard Drive Cleanup

Look at the root level of the hard drive and delete any temporarly log files left over from the installation of software.

1.4.3 Reset Safari

Open Safari and choose **Reset Safari...** from the **Safari** menu. Check all the boxes and reset, then quit.

1.4.4 User Profile Customization

Any changes we've made to the User's desktop must now be saved so that when the computer is re-registered the user gets the same settings.

Open `Terminal.app` (in `/Applications/Utilities`) and type the following:

```
sudo -s
```

You will be asked for the administrator's password. Once you have correctly authenticated, your prompt will begin with a hash (#).

If you changed the background image and want that to stick for new users, copy the background preferences to the global prefs:

```
cp "/Users/suffieldacademy/Library/Preferences/com.apple.desktop.plist" \
"/Library/Preferences/"
```

We need to provide some default settings for new user accounts:

```
cd "/System/Library/User Template/"
```

That moves you to the folder where the user settings are kept. Before we do anything else, make a copy of the existing settings:

```
cp -pR Non_localized Original_Non_localized
```

Now we're ready to copy settings from our current user into the default settings for the machine. Each of the commands below has been split into two lines. You may enter each command on two lines (as shown), hitting the **return** key after the backslash. Alternately, you may omit the backslash entirely and type the commands all on a single line.

Since we modified the dock to hold our new applications, we'll move that over as well:

```
cp "/Users/suffieldacademy/Library/Preferences/com.apple.dock.plist" \
"Non_localized/Library/Preferences/"
```

We'll copy the modified FirstClass settings:

```
cp -R "/Users/suffieldacademy/Library/firstclass" \
"Non_localized/Library/"
```

If you've installed Gradekeeper on this computer, you'll need to copy the registration preferences over:

```
cp "/Users/suffieldacademy/Library/Preferences/Gradekeeper.plist" \
"Non_localized/Library/Preferences/"
```

If you've installed FileMaker on this computer, and you've copied the Suffield Opener script to the desktop, you'll also want to add those files to the default. Note that the scripts must be on the user's desktop for this line to work:

```
cp "/Users/suffieldacademy/Desktop/"*.fp? \
"Non_localized/Desktop/"
```

1.4.5 Deleting All Users

If you wish to return the computer to a "factory default" state, where the user must register the machine and create a new admin user, you can do so. We recommend copying the machine to a disk image, and then opening the disk image and making the changes. In the example below, the disk image has been mounted at `/Volumes/Macintosh HD`.

```
sudo /usr/bin/dscl -f \
/Volumes/Macintosh\ HD/private/var/db/dslocal/nodes/Default \
localonly -delete /Local/Target/Users/suffieldacademy

cd /Volumes/Macintosh\ HD/private/var/db/dslocal/nodes/Default/groups

for g in *.plist; do group=${g%.plist}; echo $group
sudo /usr/bin/dscl -f \
/Volumes/Macintosh\ HD/private/var/db/dslocal/nodes/Default \
localonly -delete \
/Local/Target/Groups/$group GroupMembership suffieldacademy
done

sudo rm \
/Volumes/Macintosh\ HD/private/private/var/db/dslocal/nodes/Default/config/SharePoints/Suffield*.plist

sudo rm /Volumes/Macintosh\ HD/private/var/db/.AppleSetupDone
```

Those lines remove the user from the directory, remove it from all the groups, drop any shares associated with the user, and finally, remove the AppleSetup flag (which forces re-registration).

1.4.6 Networked Users

For **networked workstations** (not laptops), we tell the machine to allow anyone with a valid network name and password to log on.

Open **System Preferences** and click on the **Accounts** icon.

Unlock the preference pane, if necessary.

Click on the **Login Options** icon at the bottom of the screen.

There will be a button to **Join** a network directory system. Click this button.

If not already filled in, use "ldap.suffieldacademy.org" as the server.

1.4.7 Local Admin Group

For **networked workstations** (not laptops), we add a special group so that we can easily administrate the computers with our own usernames and passwords.

In the terminal, type the following:

```
dscl /Search -read /Groups/helpdesk GeneratedUID
```

That will give you the UID of the group you'd like to nest (use something else instead of "helpdesk" for your group name).

Now:

```
dscl . -append /Groups/admin NestedGroups GENERATED-UID-OF-NETWORK-GROUP
```

Substitute the UID you got from the first step.

That adds our OpenDirectory "helpdesk" group to the local administrators group, granting us the rights to make administrative changes on the machine.

1.4.8 Printer Group Changes

On **networked workstations** (not laptops), we add our networked groups to a special group so that any user can add or delete printers on the system.

In the terminal, type the following (all on one line):

```
dseditgroup -o edit -n /Local/DEfault -u suffieldacademy -p -a
suffield -t group lpadmin
```

This authenticates as user "suffieldacademy" and adds our "suffield" network group to the "lpadmin" group, effectively allowing all our networked users to modify printer settings.

1.5 Building the Image

Netboot the master machine.

Start DeployStudio.

Choose the "Create master from hard drive" option.

1.6 NetRestore Configuration

Once you've built a NetRestore image, you have to make NetRestore aware that it exists. A quick configuration file change allows NetRestore to "see" the new image and make it available for restoring.

Note: The reader is assumed to have some general experience with NetRestore; we do not provide an in-depth discussion of what NetRestore is or how it works. For more information on NetRestore (including documentation), please see the NetRestore web site.

1.6.1 Updating an Existing Image

If you're building an image that replaces an existing one, you simply need to copy the new image onto the NetBoot server and replace the existing image. Currently, all images on the NetBoot server reside on the **Images** drive, filed away by image type.

You may wish to move the existing image to a temporary location before deleting it, in case the new image does not work as expected.

Once the image is copied, make sure it has the proper permissions. You can easily do this from the command line by running the following commands:

```
sudo chown netrestore_access:netrestore_admin ImageFile.dmg
sudo chmod 464 ImageFile.dmg
```

Replace `ImageFile.dmg` with the full path to the image file you copied.

Once this is done, the change should take effect immediately. NetRestore should begin using the new image without needing any further configuration.

1.6.2 Adding a New Image

In some cases, you may want to add a new image type to NetRestore. First, follow the instructions above for adding an image to the server and setting its permissions. In this case, however, do not replace an existing image; rather, pick a new name for the image that reflects what it does.

Next, you will need to update the `netboot-configurations.plist` file on the server. Currently, that file lives on the **Images** volume, in a folder called **WebConfig**. When our NetBoot image runs NetRestore, it loads this folder over the network and reads the configuration file found there.

The easiest way to add an image is to copy an existing stanza from this file and customize it for a new image. Here is a skeleton stanza you might want to use:

```
<key>Image Name</key>
<array>
    <string>afp</string>
    <string>veronica.suffieldacademy.org</string>
    <string>Images</string>
    <string>Path/To/ImageFile.dmg</string>
    <string>netrestore_access</string>
    <string>password</string>
    <string>Description of Image</string>
</array>
```

Assuming you store the image on the **Images** volume, you only need to customize the `Image Name` to give a short title to the image, the `Path/To/ImageFile.dmg` field to include the relative path to your image (relative to the **Images** volume), the `password` line to include the password to the server (ask the Network Administrator if you don't know it), and the `Description` field to define what the image does.

Once this file has been saved, the changes take effect immediately for clients accessing the settings via the web. Launch **NetRestore** and verify that the information is correct.

1.7 Resources

Mike Bombich is the creator of Carbon Copy Cloner (to create disk images) and NetRestore (to reimage the machines). His main web site is www.bombich.com, and it contains a wealth of information on building images and managing large numbers of computers.

Chapter 2

Installing Windows XP

Last updated 2008/03/18

2.1 Introduction

Suffield Academy has standardized on Windows XP Professional edition for use with all Windows-based workstations on the campus network.

This document describes how to install and configure Windows XP Professional with Service Pack 2 (the latest version, as of this writing). Later sections of the document also discuss installing the standard set of software used at Suffield Academy.

This document assumes a general familiarity with Windows concepts (such as the running installers, and using the Start Menu and System Tray). However, it assumes no system administration knowledge, so even first-time installers should have no trouble following the instructions.

2.2 Installation

2.2.1 Before You Start

These instructions assume installation onto a PC that has nothing of value on it. **Following these instructions will erase everything that is currently installed on the PC.** Do not perform the installation until you are sure everything of value has been backed up on the machine.

To complete the installation, you must have a keyboard, mouse, monitor, and network connection. The network should be connected during the entire installation.

You'll need to be a domain administrator to correctly join the machine to the network. If you do not have these priviledges, you will be able to install Windows, but you cannot join the machine to the network properly. Complete the installation and contact the Network Administrator for further assistance.

2.2.2 BIOS Settings

To install Windows XP, we must boot the PC off of the built-in CD-ROM drive. You should first try booting the machine with the installation CD in the drive. If the PC fails to boot off of the CD, you may need to modify the BIOS settings.

The BIOS settings can only be accessed immediately after the computer has been turned on. Usually, the computer prints a message telling you how to enter the BIOS setup utility. Often, this is done by pressing a special key: `F1`, `F2`, `F12`, and `delete` are common keys for this feature.

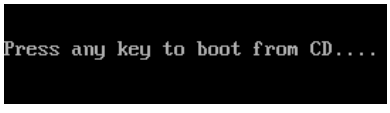
Once inside the BIOS setup, check for a setting related to **boot order**. Ensure that the computer is set to boot off CD-ROM first, and then off the internal hard drive.

Save your settings and follow the instructions below for booting the machine off of CD.

2.2.3 Booting

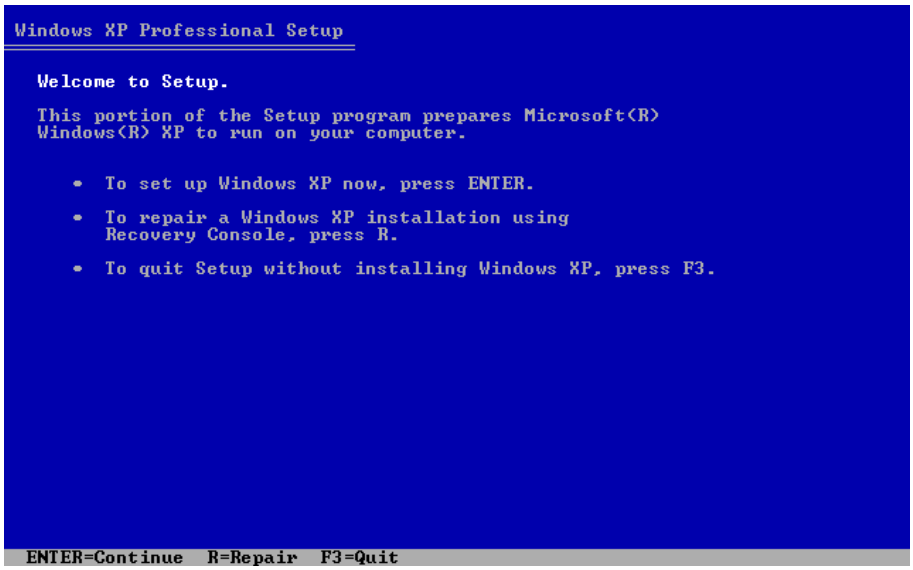
Put the Windows XP Professional CD into the CD-ROM drive and start up the computer.

The computer will print `Press any key to boot from CD` on the screen:



We want to do this, so strike a key on the keyboard as soon as you see this message (it will give up after a short time, so don't wait too long).

The screen will turn blue, and the Windows XP installer will start up. It takes a minute or so to load all of its files and bring up the main installer. You'll see a screen that says `Welcome to Setup` at the top.



Hit `enter` to continue to the next screen.

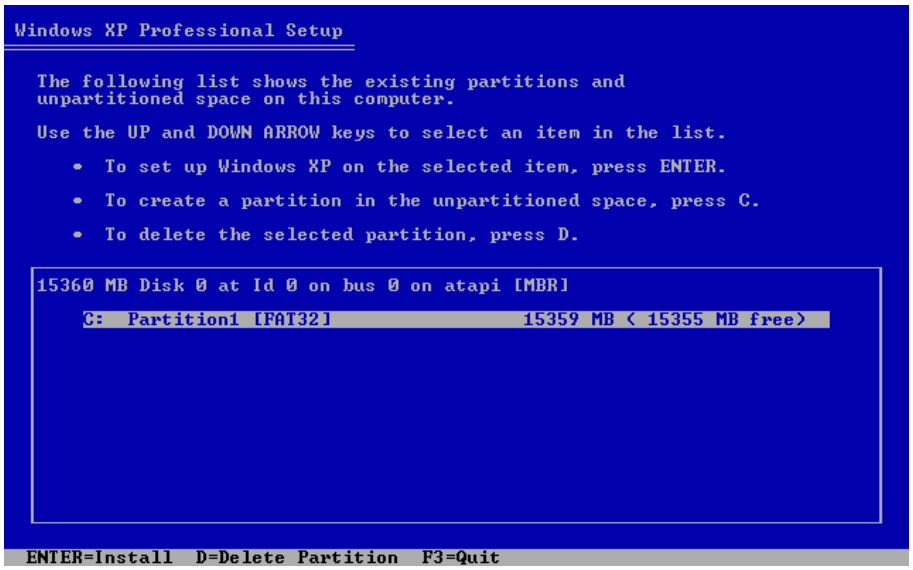
You'll be presented with a License Agreement. Sign your life away and press `F8`.

2.2.4 Partitioning

The setup program searches your hard drive for exising data. Because we've backed everything up on this computer (you **did** back everything up, right?), we're just going to erase the drive and start from scratch.

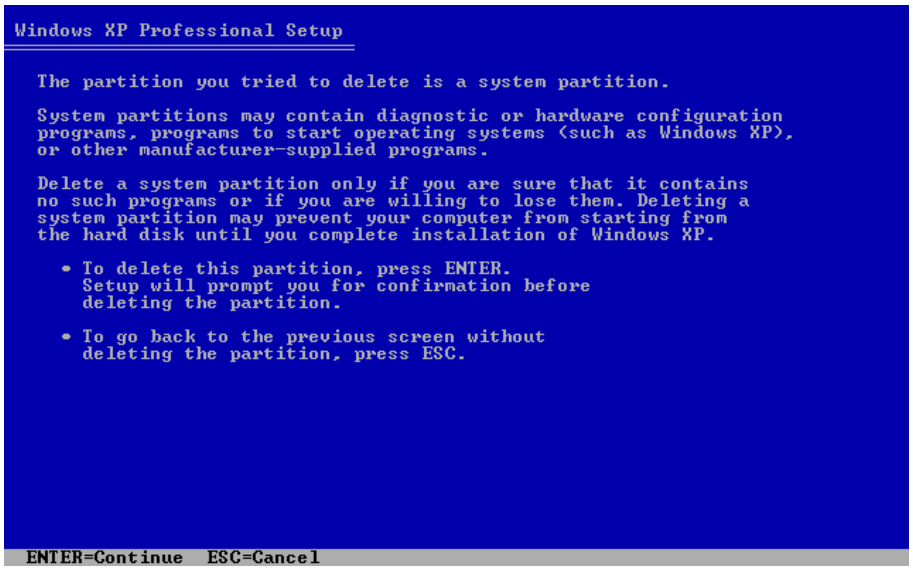
You'll see a listing of partitions on your computer. If all you see is the line `Unpartitioned space`, with no drive letters (such as `C:`), you may skip to the next step.

Otherwise, for each partition listed, you must highlight it and delete it. Use the arrow keys to highlight a drive letter (**e.g.**, `C:`, `D:`, and so on):



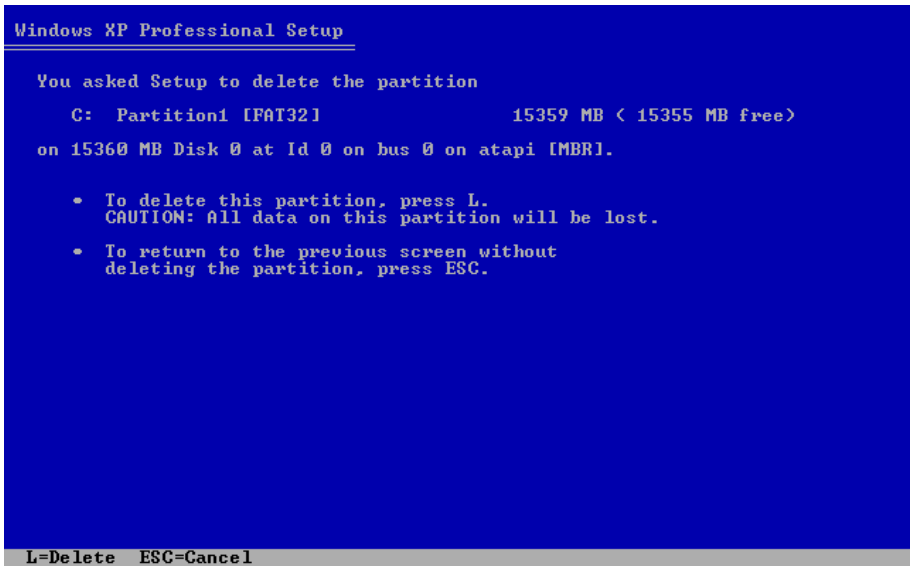
Once you've highlighted a partition, press the `D` key.

The installer reminds you that there is already data on this parition.



Press `enter` to confirm.

The installer then asks you to confirm that you really want to delete this partition.

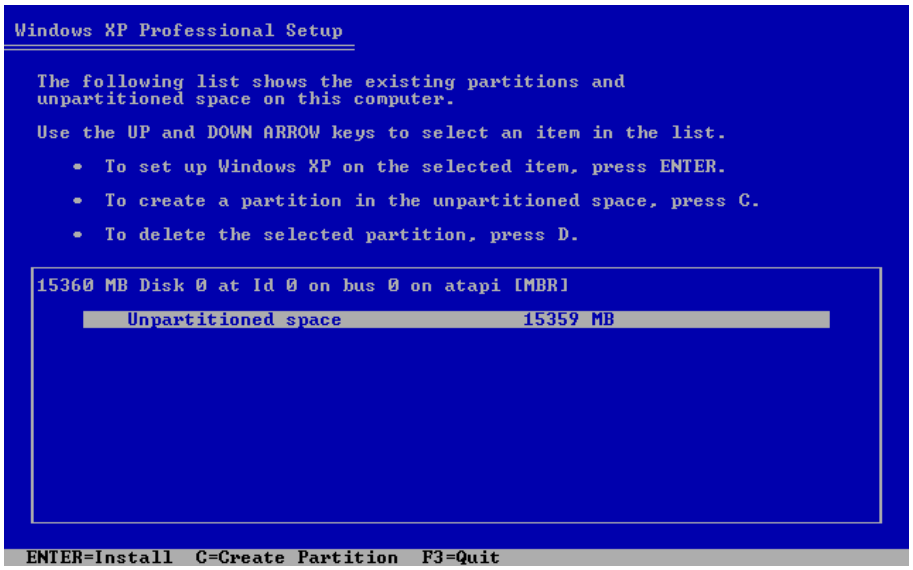


Press `L` to confirm. **All data on this partition will be erased after this step, so make sure you've backed up all important files.**

Repeat this procedure until there are no partitions remaining (the menu should only show a line that says `Unpartitioned space`.

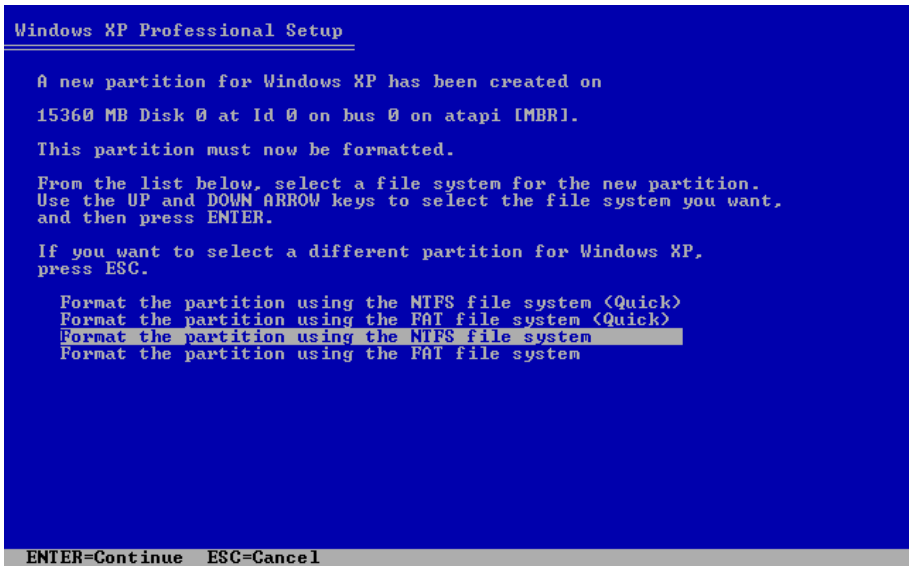
2.2.5 Initializing the Disk

You should now have a screen that has one choice: the `Unpartitioned space` line on the partition menu:



Hit the `enter` key to use this entire drive for Windows XP.

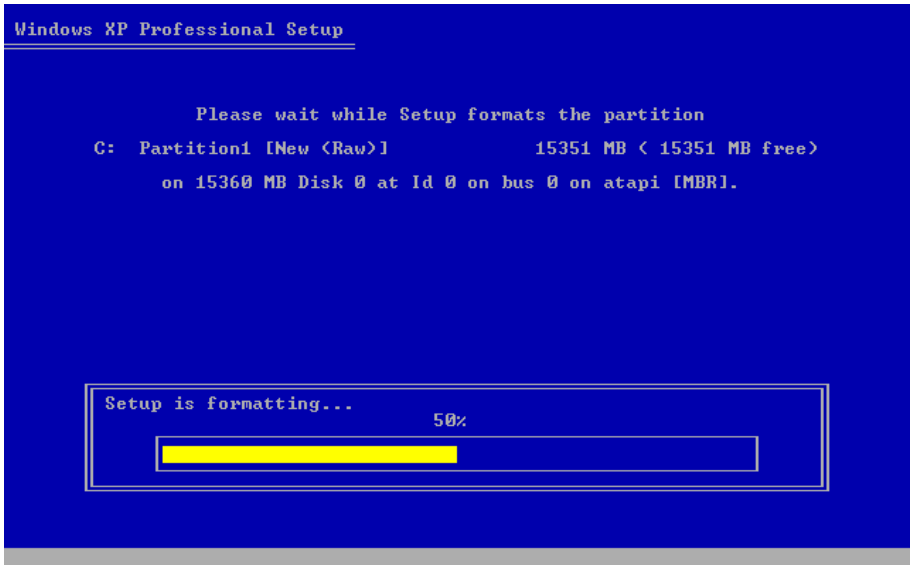
The installer will ask you how you want to format the drive. You should the `NTFS` filesystem. If the computer is brand new, you may use the "Quick" version of the format. For older computers, or if you're unsure, you should use the regular formatting to ensure that the drive has no errors:



Select your choice and hit `enter`.

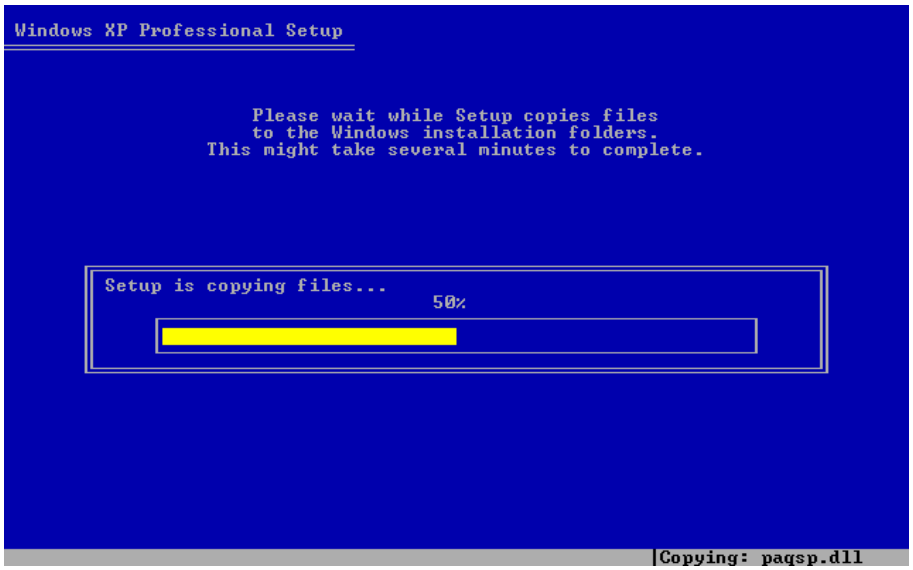
The computer will now format the drive in your computer. Depending on the

size of the drive, the speed of your computer, and whether you opted for the "quick" format, this can take several minutes.

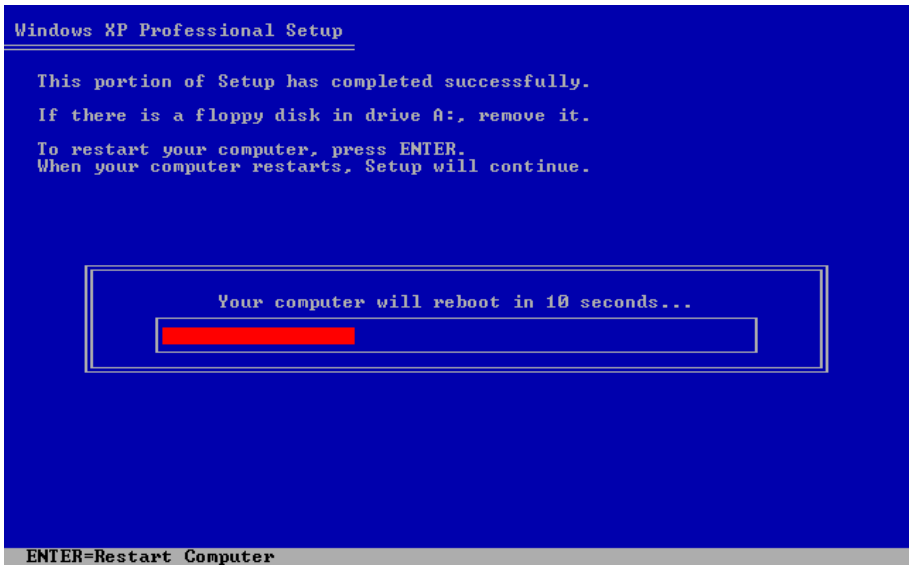


2.2.6 Waiting for Installation

Once the drive is initiallized, the installer will automatically begin copying files from the CD onto your computer. Again, depending on the speed of your computer and CD-ROM drive, this can take several minutes.



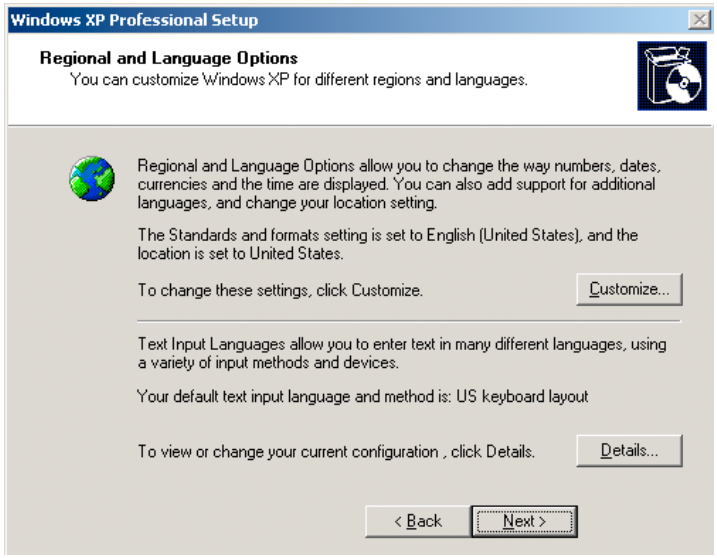
When the files have copied successfully, the computer will automatically restart.



You should leave the installation CD in the CD-ROM drive, however, you should take care **not** to boot off of the CD (do not press a key when the computer prompts you); let the computer start up off the hard drive.

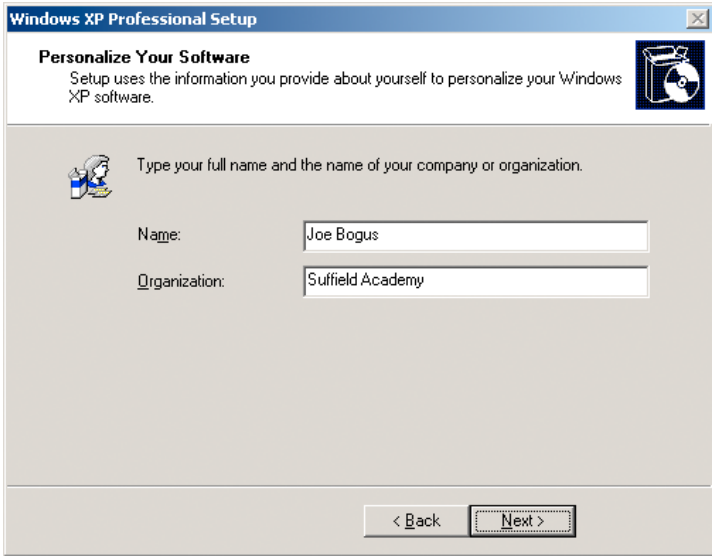
The computer will start up Windows XP, and will start the second phase of installation. For the most part, you can simply sit back and let the installation complete on its own. We describe below the parts that require action by you.

2.2.7 Regional and Language Options



No customization is required, hit **Next**.

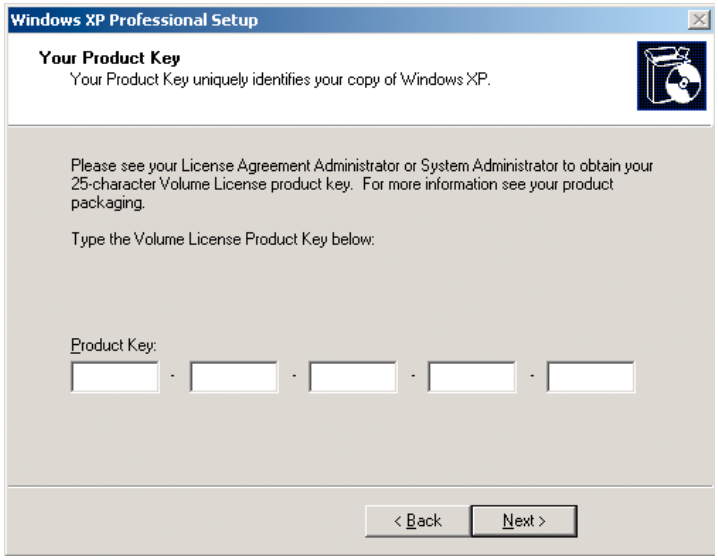
2.2.8 Personalize Your Software



Fill in the name of the person who will be using this computer. If it is a public machine that will be shared by several people, use `Suffield Academy` as the name.

You should enter `Suffield Academy` as the organization.

2.2.9 Product Activation Key



Here you must enter the 25-character product key for Windows. The key can be found in the software database, or on the installation media.

2.2.10 Computer Name and Administrator Password

For the computer name, you should enter a DNS hostname that describes this

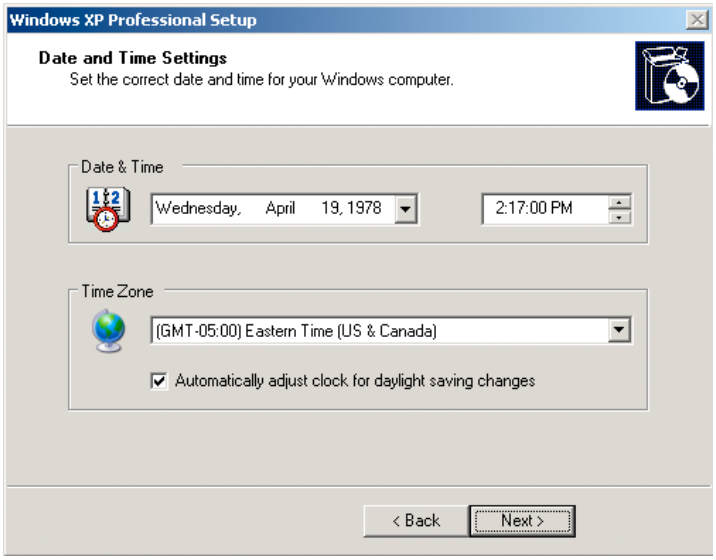
machine. For machines used by a single person, we use the `<username>-desktop` format (*e.g.*, `jbogus-desktop`). For shared machines, use a name that describes the machine (*e.g.*, `bookstore` or `art-lab-1`).

Do **not** use spaces, underscores, or other punctuation in the name. Only use numbers, leters, and dashes for the name.

The installer may force you to shorten the name of the machine. That's fine; just make note of the new shortened name.

For the Administrator password, use one of the standard system passwords for the department or lab where the computer is being installed. Consult with the Network Administrator if you do not know which password to use.

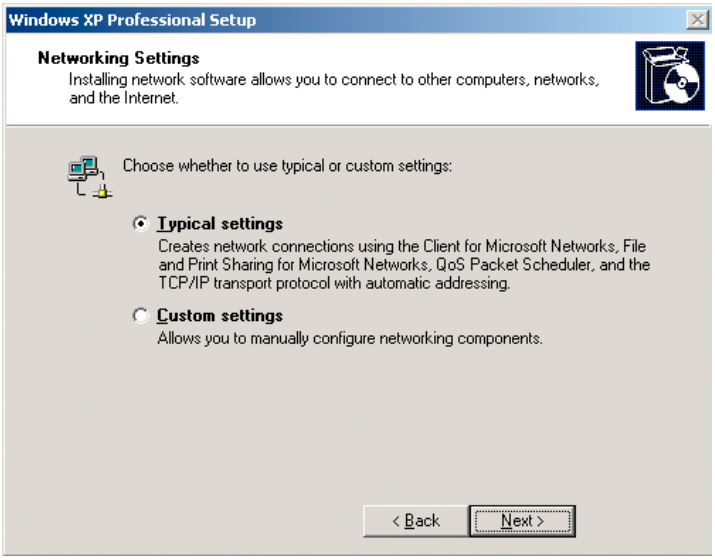
2.2.11 Date and Time Settings



Set the proper date and time. Use `(GMT-05:00) Eastern Time (US & Canada)` as the Time Zone. The **Automatically adjust clock for daylight saving changes** checkbox should be *selected*.

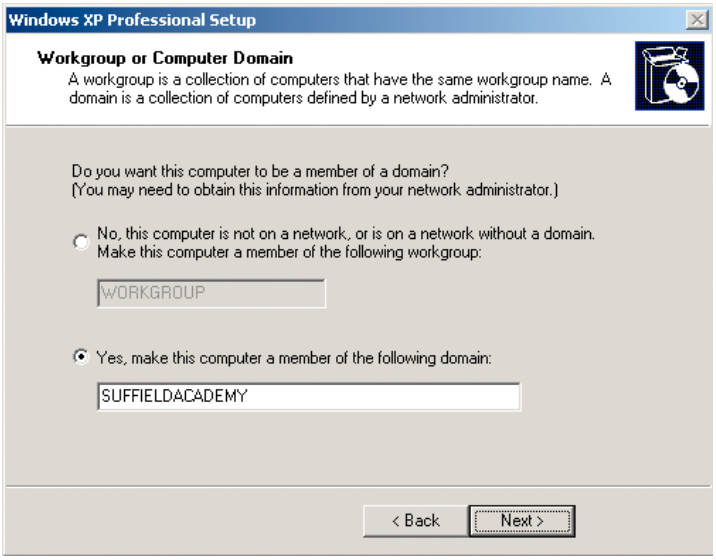
2.2.12 Networking Settings

Ensure that the computer has a network cable plugged into it, so it is ready to connect to the network.



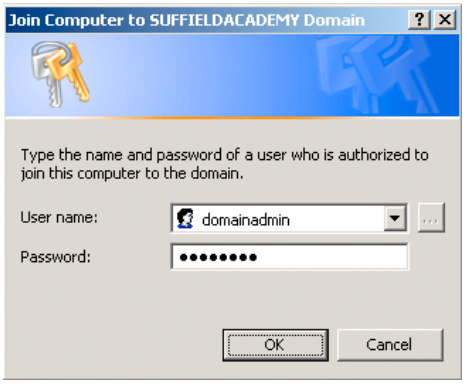
Leave the **Typical Settings** radio button *selected* and click the **Next** button to proceed.

2.2.13 Workgroup or Computer Domain



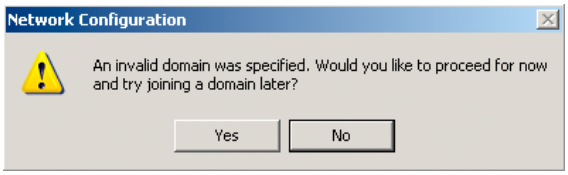
On this screen, *select* the radio button titled **Yes, make this computer a member of the following domain**. In the text area below the label, enter `SUFFIELDACADEMY`. Click on the **Next** button.

A dialog box will appear asking for the username and password of a domain administrator.



Enter your network username and password.

If you do not have domain privileges, you will be told that you cannot join the computer to the domain:



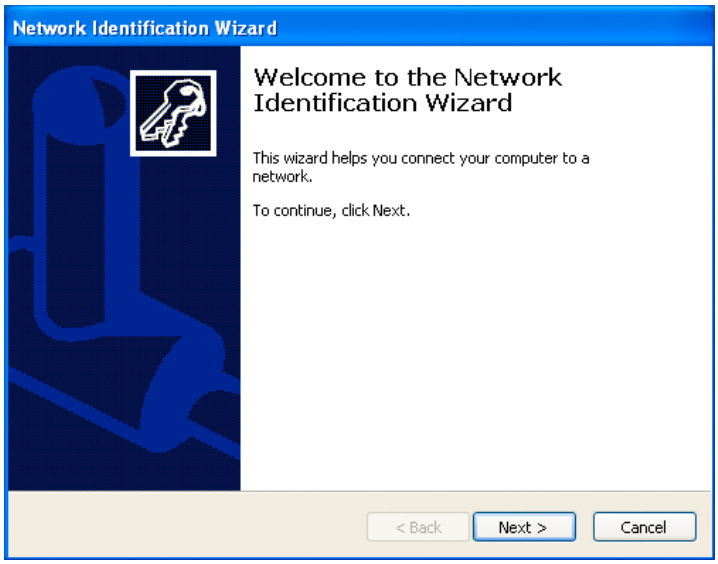
Click **No** and try to add the computer to the domain again. Make sure that the computer is properly networked, and that the username and password have been typed correctly. Finally, make sure the username provided has Domain Admin rights. If you're not sure, contact the Network Administrator before proceeding.

2.2.14 Finishing Initialization

After the Workgroup Settings are complete, the computer will continue to initialize Windows, which may take 20 minutes to an hour. After the process is complete, the machine will automatically reboot. Once again, do **not** boot off the installation CD; let the computer start up off of the hard drive.

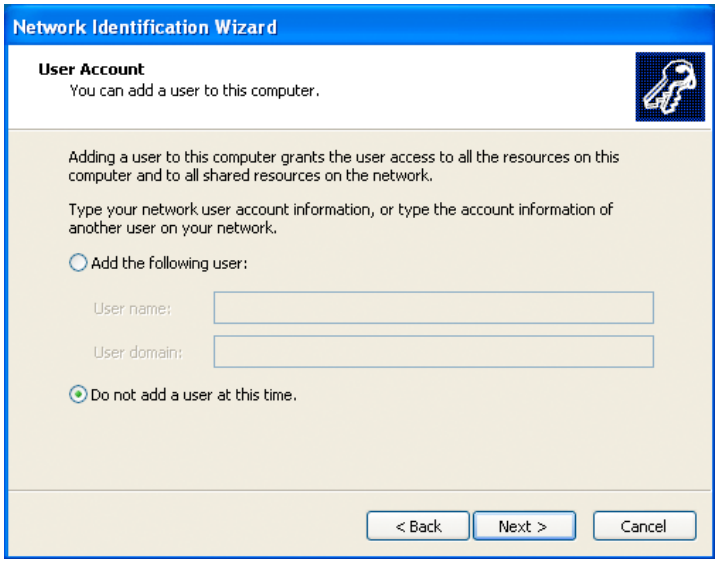
2.2.15 Network Identification Wizard

When the computer starts up again, a screen will appear that says **Network Identification Wizard**:



Click **Next** on the main screen.

A screen will appear asking you if you wish to add a user to this computer:



Find the radio button labeled **Do not add a user at this time** and *select* it. Click the **Next** button to continue.

Click the **Finish** button to complete the wizard.

2.2.16 Logging In

At this point, you should see the Windows XP login screen, which says **Press Ctrl-Alt-Delete to begin**:

You are now ready to log in and begin to set up the computer. Press `Control`, `Alt`, and `Delete` together to bring up the login screen.

Click the **Options** button to show full login options. Users will need the options later when they log in as a networked user.

You should log in as `Administrator` with the password you set during the installation procedure.

The computer will log in the Administrator user, and bring up the standard Windows XP desktop. You're now ready to begin setting the computer up.

2.3 Setting Up the OS

At this point, your installation of Windows XP is complete. You should be logged into the machine as a local Administrator. Windows takes you through several final customization steps, and we perform a few tweaks to the settings so the computer is ready to use.

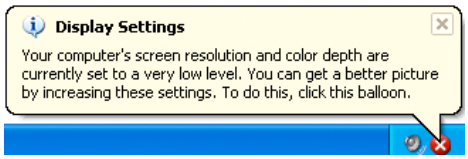
2.3.1 Eject the Installation CD

We're done with the installation CD, so you may eject it from the computer.

2.3.2 Display Settings

Automatic Settings

Windows may pop up an alert telling you that your display settings are set to a very low level:

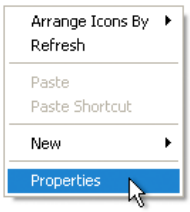


Dismiss the alert; we'll change the settings in the next step.

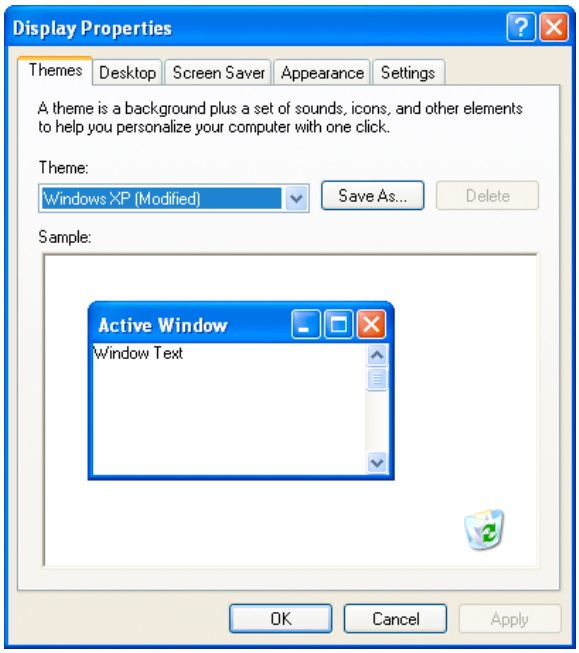
If you do not get the alert, just move on to the next step.

Adjusting the Settings

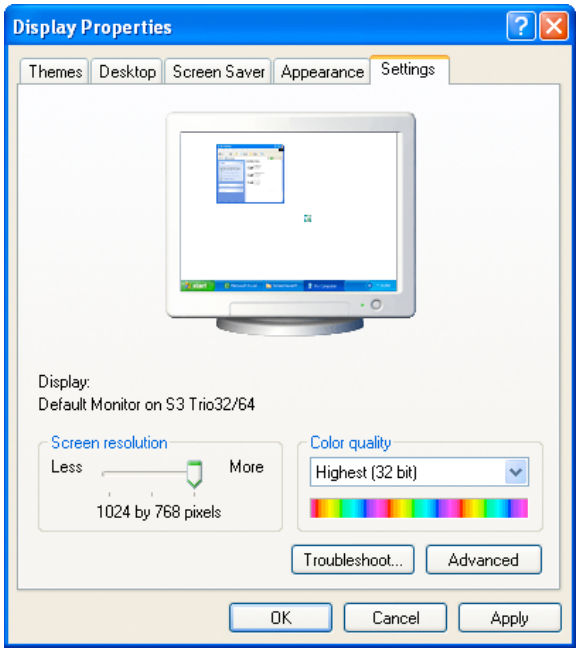
Right-click on the desktop, and select **Properties** from the menu that appears.



The **Display Properties** window should appear.



Select the **Settings** tab.



Use the slider at the bottom of the window to increase the screen resolution. You should select 1024 by 768 pixels. The user may choose a larger resolution

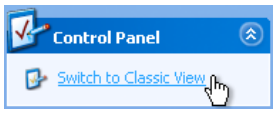
at a later time, if they wish. Under the **Color quality** menu, select **Highest (32 bit)**.

Click the **Apply** button. Confirm that the new settings work on the monitor. Click the **OK** button.

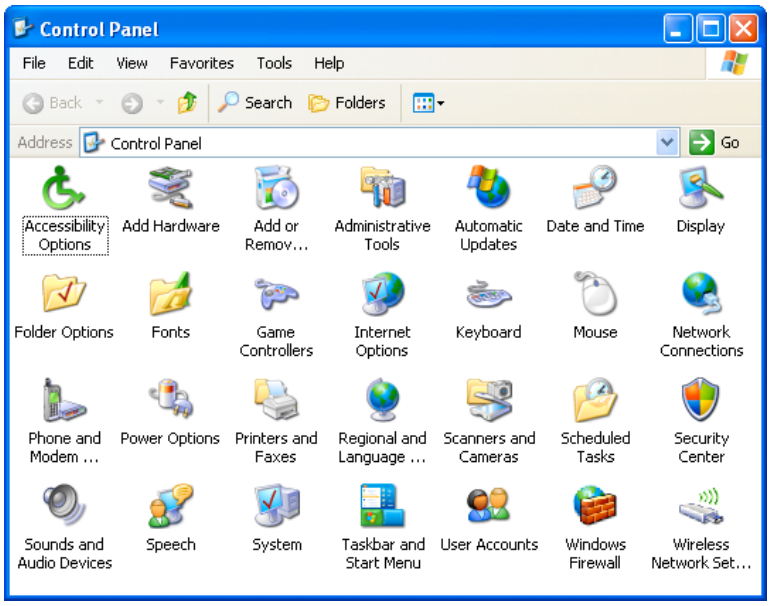
2.3.3 Control Panel Settings

Go to the **Start Menu** and select **Control Panels**.

When the control panel window appears, click the icon on the left-hand side that says **Switch to Classic View**.

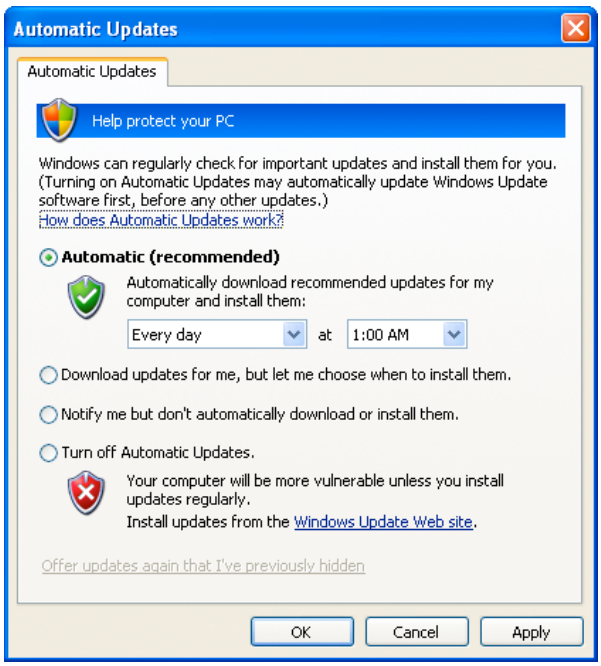


You should now see a full listing of control panels to work with.



For each of the following subsections, open the control panel with the given name and follow the directions provided.

Automatic Updates

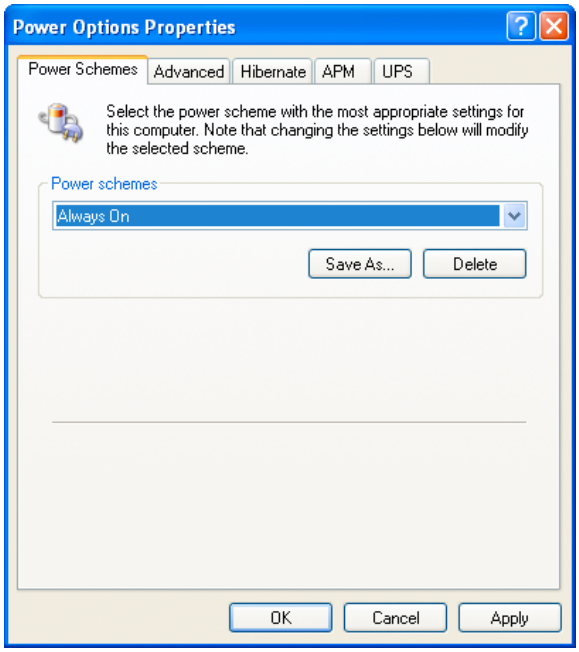


Find the radio button labeled **Automatic (recommended)** and ensure it is *selected*.

Set the computer to download and install updates `Every day` at `1:00 AM`.

Click **OK** to close the control panel.

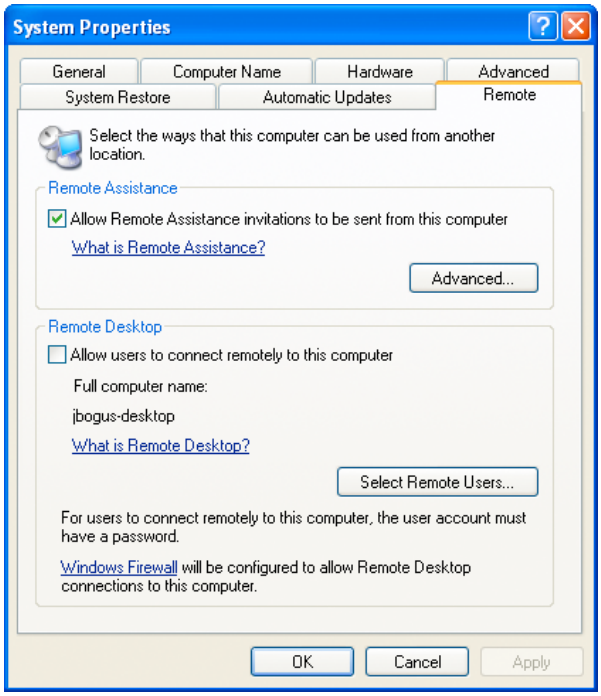
Power Options



From the **Power schemes** menu, select `Always On`.

Click **OK** to close the control panel.

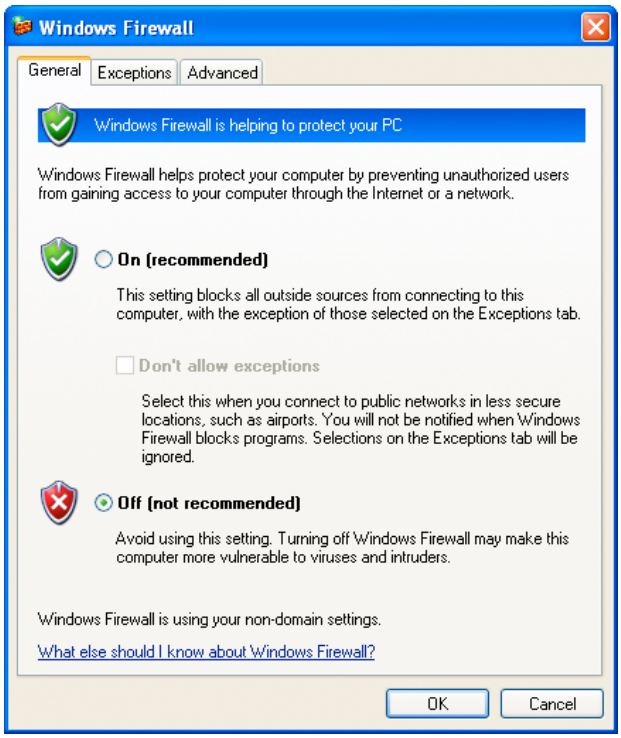
System



Select the **Remote** tab. Ensure that the **Remote Assistance** box is *selected*, and that the **Remote Desktop** box is *unselected*.

Click `OK` to close the control panel.

Windows Firewall



The built-in Windows XP firewall is intended for users on a direct connection to the Internet (like you might find at home). It interferes with software we use on our network, and so it must be disabled.

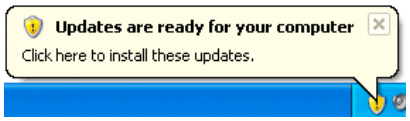
Select the radio button labeled **Off (not recommended)**.

Click **OK** to close the control panel.

When you have completed all of the sections above, close the **Control Panels** window.

2.3.4 Install Pending Updates

At this point, you may see a small yellow shield icon in the system tray:



Click it to display the **Automatic Updates** window. Install any pending sys-

tem updates.

While the updates are installing, you may continue to work with the machine. Move on to the next section and install the standard software on the machine.

When you are prompted to restart the computer, complete the task you are working on, and restart.

2.4 Installing Standard Applications

Once Windows XP is correctly installed and configured, you must install the standard suite of applications in use at Suffield Academy.

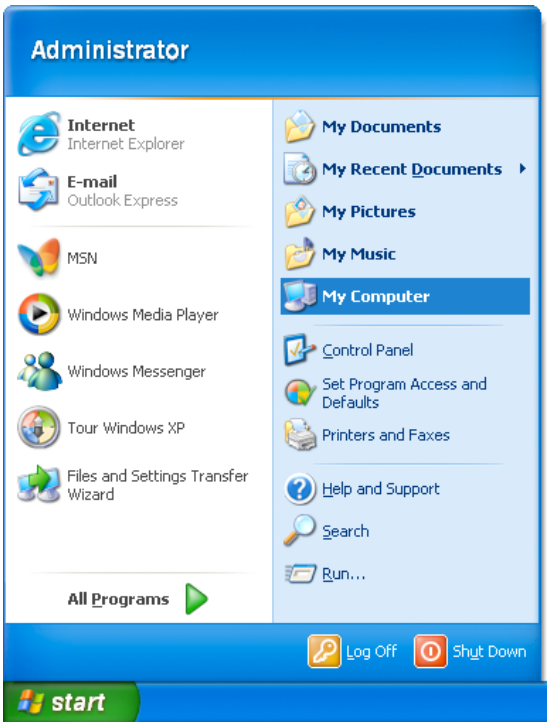
Most installations are simple and straightforward, so we do not provide detailed step-by-step instructions here. Any configuration that requires selecting non-default options will be explained in the sections below.

You should be logged in under the local Administrator account before following these instructions.

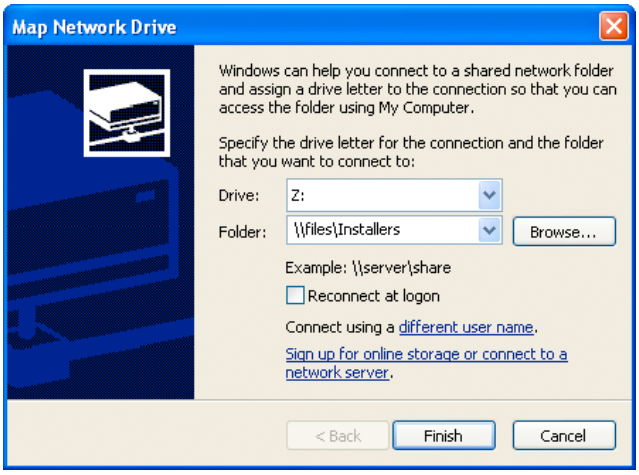
2.4.1 Connecting to the Fileserver

All of the applications are located on our central file server. Because some installers reside in protected directories, you will need to provide a name and password when you access the fileserver.

1. Begin by clicking on the **Start Menu** and selecting **My Computer**.

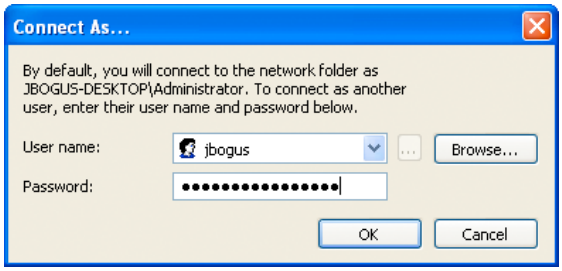


2. In the **Tools** menu, select **Map Network Drive...**.



3. For the **Drive** setting, use any free letter (`Z` is usually a good choice).
4. For the **Folder** setting, enter: `\\files\Installers`.
5. Ensure that the **Reconnect at logon** box is *unselected*.
6. Click on the **Connect using a different user name** link.

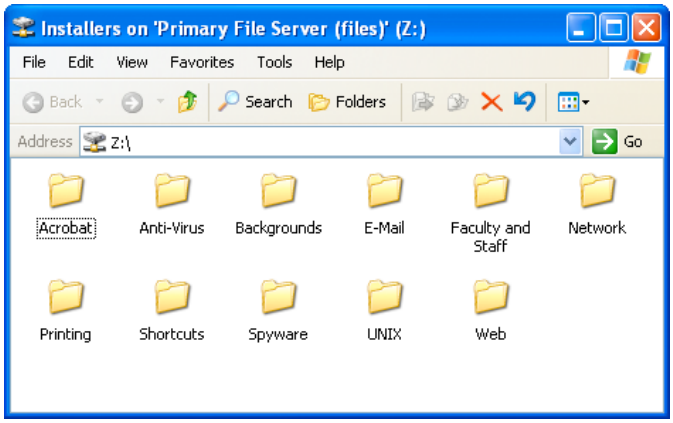
7. In the window that appears, enter your network username and password.



   Click **OK** when you've entered them.

8. Click the **Finish** button to set up the connection to the fileserver.

Your drive should mount, and you should see a new window containing the contents of the fileserver's **Installers** folder:



The sections below work with the contents of this folder.

2.4.2 First Class

Open the **E-Mail** folder on the fileserver.

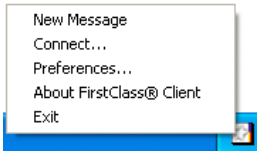
Open `FC8031US` installer program. Follow the default prompts to install the application. Be sure to install the application for **Anyone who uses this computer (all users)** when asked.

You should not need to install the `home.fc` file for network users; it should already be included in their account.

Default Mail Program

We must register FirstClass as the default mail reading application on the computer.

When the installation is complete, FirstClass will launch, and an icon will appear in the system tray.



Right-click on this icon and choose **Preferences**.

In the window that appears, ensure that the box labeled `Register FirstClass as default mail client` is *selected*.

Click **OK** to dismiss the window.

Log in to FirstClass to ensure it works properly. Quit once you have verified that it connects.

2.4.3 Mozilla Firefox

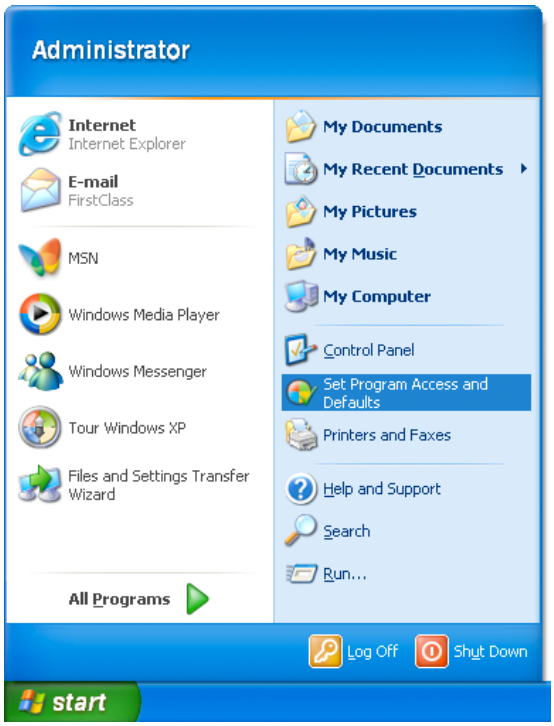
Open the **Web** folder on the fileserver.

Run the `Firefox Setup` installer in this folder. Follow the default prompts to install this application. You do not need to launch Firefox when the installation is complete.

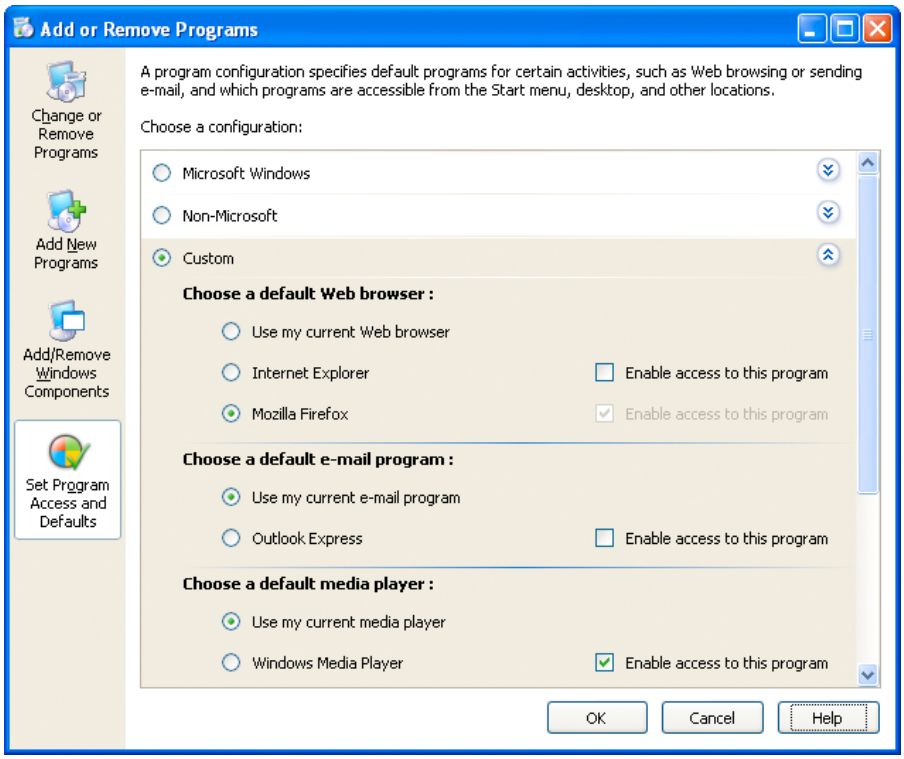
Default Web Browswer

We do not allow the use of Internet Explorer on our workstations. Therefore, Firefox must be set as the default browser so Windows uses it for viewing web pages.

Go to the **Start Menu** and select **Set Program Access and Defaults**. You may need to look in the **All Programs** section of the menu to find it.



Ensure that the **Custom** radio buton is *selected*. Click on the double-arrow to the right of where it says **Custom** to reveal the program settings.



Under **Choose a default Web browser**, ensure that **Mozilla Firefox** is *selected*. Also, *deselect* the **Enable access to this program** checkbox next to **Internet Explorer**.

Under **Choose a default e-mail program**, ensure that **Use my current e-mail program** is *selected*. Also, *deselect* the **Enable access to this program** checkbox next to **Outlook Express**.

Leave the remainder of the sections alone. Click **OK** to save your changes.

2.4.4 Mozilla Sunbird

Open the **Web** folder on the fileserver.

Run the `Sunbird Setup` installer in this folder. Follow the default prompts to install this application. You do not need to launch Sunbird when the installation is complete.

2.4.5 Microsoft Office 2003

Open the **Faculty and Staff** folder on the fileserver, then open the *Office* folder inside of that. Finally, open the **Office Professional 2003** folder (the rest of the folder name is the serial number for the software).

Inside this folder are several subfolders containing both the initial installer and service pack updates to the software.

There should be shortcuts to the various installers in this folder. Simply run the installers in the listed order (*e.g.*, "Step 1", "Step 2", "Step 3").

If ask for an **installation type**, opt for a **Complete Install**. If an **activiation key** is requested, it can be found embedded in the name of the folder that contains the installer program.

2.4.6 FileMaker 6

Open the **Faculty and Staff** folder on the fileserver, then open the **Filemaker** folder inside of that. Finally, open the **FMPro6** folder (the rest of the folder name is the serial number for the software).

Find the program called `Setup` and run it. The activation key for the softare is embedded in the name of the folder that contains the installer program. Follow the default prompts to install the applications. You may safely ignore the registration of the product (choose **Online** and close the browser).

Once Filemaker is installed, the **SMTPit** extension must be installed. Move back to the **Filemaker** folder on the server. Find the file called `SMTPit`, and copy it to the `C:\Program Files\FileMaker\FileMaker Pro 6\System\` directory.

2.4.7 Acrobat Reader 7

Open the **Acrobat** folder on the fileserver.

Run the installer inside the folder. Follow the default prompts to install this application.

2.4.8 Ultr@VNC Server

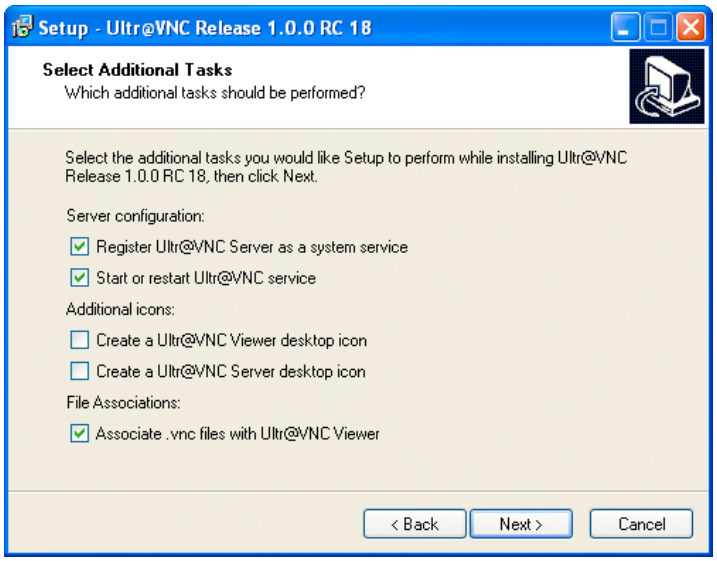
Open the **Network** folder on the fileserver.

Find and run the Ultr@VNC installer. Click through the **License Agreement**, **Information**, and **Select Destination Location** pages.

On the **Select Components** page, accept the default component list.

Accept the defaults for the **Select Start Menu Folder** page.

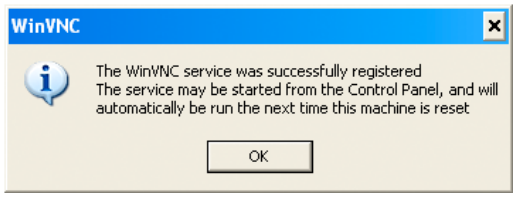
On the **Select Additional Tasks** page, do the following:



select the **Register Ultr@VNC Server as a system service** and **Start or restart Ultr@VNC service** checkboxes. *Deselect* the **Create a Ultr@VNC Viewer desktop icon** and **Create a Ultr@VNC Server desktop icon** checkboxes. Leave the other checkboxes alone.

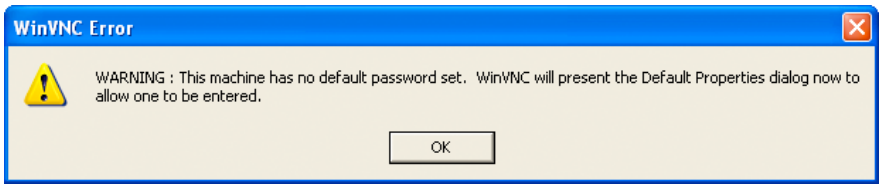
Complete the installation procedure. The installer opens a web browser to the Ultr@VNC web page, which you may close immediately. Additionally, if you receive an error message about a shell command that can't run, you may simply dismiss it.

Ultr@VNC will alert you that it has been registered as a service and will run automatically at startup:

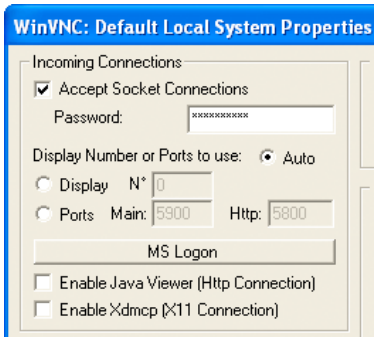


Dismiss this alert.

Ultr@VNC will warn you that there is no default password set:



Once dismissed, it will present you with a settings screen:



On this screen, enter a master password in the **Password** box (consult the Network Administrator if you do not know what password to use). Additionally, ensure that the **Enable Java Viewer** and **Enable Xdmcp** boxes are *deselected*. Click **OK** to save your settings.

2.4.9 Sophos Anti-Virus

Open the **Anti-Virus** folder on the fileserver.

FIXME: To be completed with new Sophos Anti-Virus Instructions!

2.4.10 Printing

Open the **Printing** folder on the fileserver.

FIXME: To be completed with new Printing (direct) instructions!

2.4.11 Macromedia Studio 8 (optional)

Macromedia Studio is an optional component; you only need to install it if the user of the machine requires it.

Open the **Web** folder on the fileserver.

Open the **Macromedia Studio 8** folder.

Run the `Install Studio 8` installer, and perform a default install of the application. The serial number for the product is contained in the name of the folder you are in, or you can find it in the software database.

Chapter 3

Installing ODIN

Last updated 2008/03/18

3.1 Introduction

Odin is a program used by Suffield Academy for Point-of-Sale and student debit accounts. We currently use it in the Snack Bar, Bookstore, and the Business Office (to maintain student accounts).

This document describes how to install the Odin client for Windows (Odin does not run on any other platforms). Please note, our support contract with Odin entitles us to live technical support during normal business hours. Odin has a "live chat" system that allows an Odin support tech to share your screen and help with setup. If you have difficulty getting Odin to work, visit www.odin-inc.com on the client machine and follow the support links. Someone from Odin will look at the machine remotely and help you resolve the issue.

3.2 Prerequisites

This document assumes you have followed our Windows XP Installation instructions, and have a client machine that is on the Suffield network. Briefly, this means you must be running Windows XP Service Pack 2 with Domain Authentication enabled.

You must have access to the local Administrator account of the machine. Once the software is installed, regular accounts will suffice for running the software.

You must also have a networked account that belongs to the **ODIN** group, as well as a valid Odin username and password. Contact the Network Administrator if you do not have access to the Odin group. Contact the Business Manager if you need an Odin username and password.

3.3 Installation

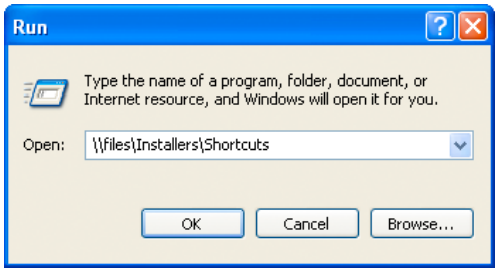
Log in to the machine with the local Administrator account, then follow the instructions below to install and configure Odin.

3.3.1 Logging In to the Server

Get Login Scripts

We must install Odin as the local Administrator, but the data files for Odin are only accessible by certain networked accounts. Therefore, we must download a special login script that connects us to the server with a networked username, while letting us use the local Administrator account on the client machine.

Go to the **Start** menu and click **Run...**. Type `\\files\Installers\Shortcuts` in the box and click **OK**.

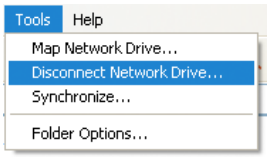


A window called **Shortcuts** will open. Find the file called `Mount Odin Network Drive.bat` and copy it onto the desktop.

Close the **Shortcuts** window.

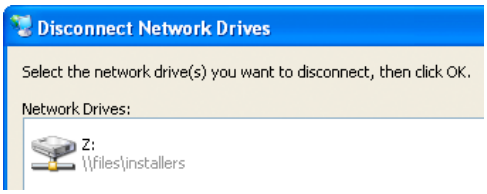
Disconnect Existing Sessions

Before we can connect to the fileserver with a username, we must first disconnect any existing sessions. Go to the **Start** menu and click on **My Computer**. In the window that appears, go to the **Tools** menu and select **Disconnect Network Drive...**.



If you get a message that says **You have no network drives to disconnect**, skip to the next session.

Otherwise, in the list of network drives that appears, select any that start with \\`files`\ and click **OK** to disconnect from them.

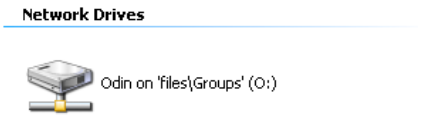


Leave the **My Computer** window open when you're done.

Mounting the Odin Drive

Double-click the `Mount Odin Network Drive.bat` file on the desktop and follow the prompts to log in. You will need the username and password of a networked user who belongs to the Odin group. Contact the Network Administrator if you need access to this group.

When you're done, the **O** drive should appear in the **My Computer** window, with the label `Odin on 'files\Groups'`:



You now have access to the Odin files, and are ready to begin installation.

3.3.2 Installing Odin

Running the Installer

Open the **O** drive. Inside the drive are several folders. Find and open the **Install** folder. Locate the installer program called **SETUP.EXE** and run it.

Install the software using the default prompts provided by the installer.

When you're done, the **Odin** folder from your start menu will appear and will contain shortcuts to the locally installed versions of **Store Manager** and **Store Register**. Delete these shortcuts, as we use the networked versions instead.

Additionally, open the folder `C:\Program Files\Odin` and delete the **Store** and **StoreMgr** programs.

Configuring Odin Database

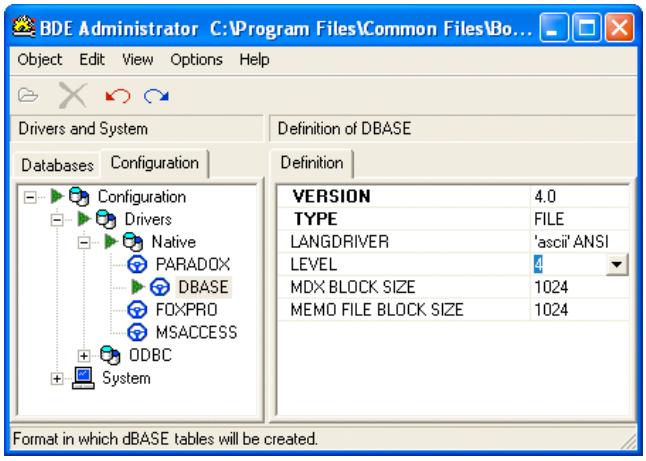
Next, we must configure Odin's database software.

Open the **Control Panel** folder:

Open the **BDE Administrator** control panel.

Click on the **Configuration** tab.

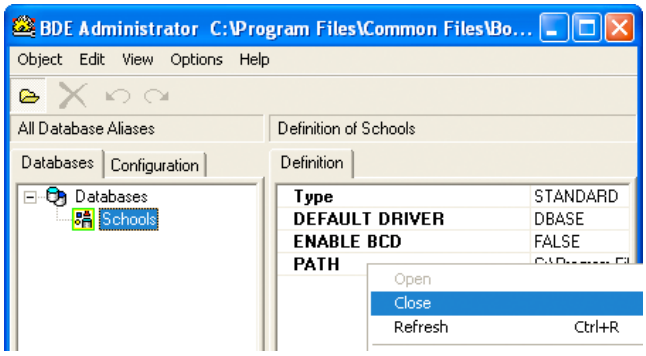
Drill down through the configuration tree to get to **Configuration::Drivers::Native::DBASE**. Change the **LEVEL** parameter to `4`.



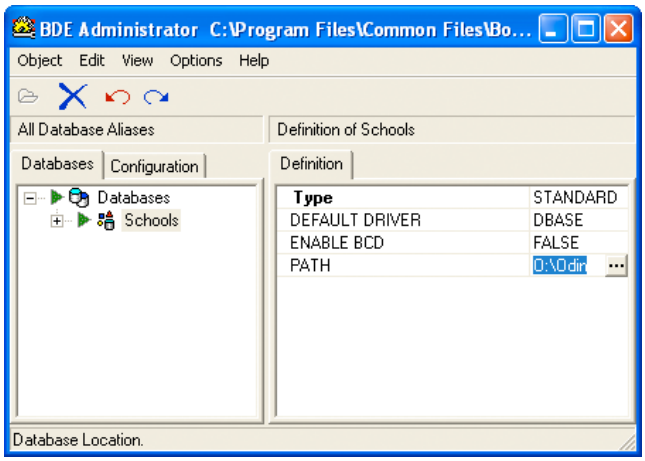
Next, click on the **Databases** tab.

Drill down through the configuration tree to get to **Databases::Schools**.

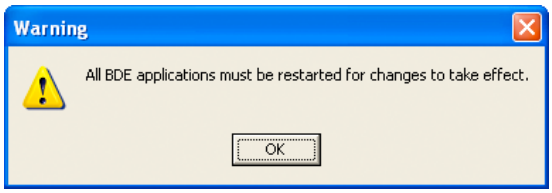
Right-click on the **PATH** key and choose **Close**.



Now change the value for **PATH** to be `O:\Data`.



Close the **BDE Administrator** Control Panel, and save your changes when prompted. You should then receive a warning that all applications must be restarted:

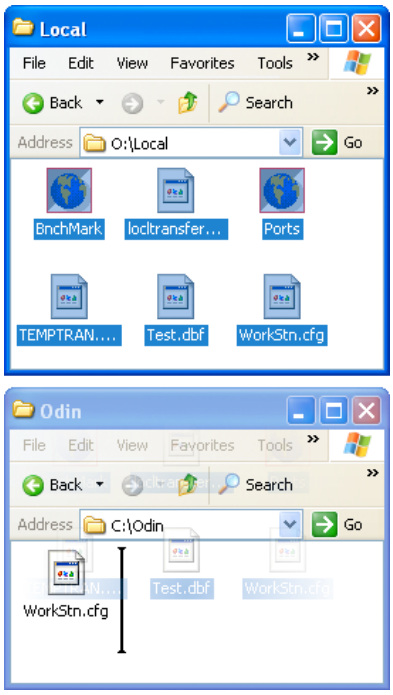


Dismiss the warning, and close the **Control Panels** window.

If you do **not** get this warning, **stop**. Go back and double-check your **BDE Administrator** settings to ensure that they took effect.

Copying the Configuration Files

The server contains a skeleton directory with a mostly-working configuration. Open the folder `O:\Local` and copy the contents of the folder into `C:\Odin`. Overwrite any existing files.



Open the file `WorkStn.cfg` using **Notepad** (or another text editor).

Each workstation that uses Odin must have a unique **Register Code** assigned to it. A list of codes that have already been used can be found in **O:\Odin\Shared\RegisterAssignments.txt**.

Under the `[Register]` section of the `WorkStn.cfg` file, change the line `Code` to have a unique value (single letter or digit). Add the value to the `RegisterAssignments.txt` file when you're done. If you do not have access to this file, notify the Network Administrator of the new register code.

Optionally, you may change the Odin area that the user sees by default. Under the `[Misc]` section of the config file, you can change `DefaultSalesArea` to have one of the values found in the **O:\Shared\odin.cfg** file.

Close the `WorkStn.cfg` file and save your changes.

Link Executables

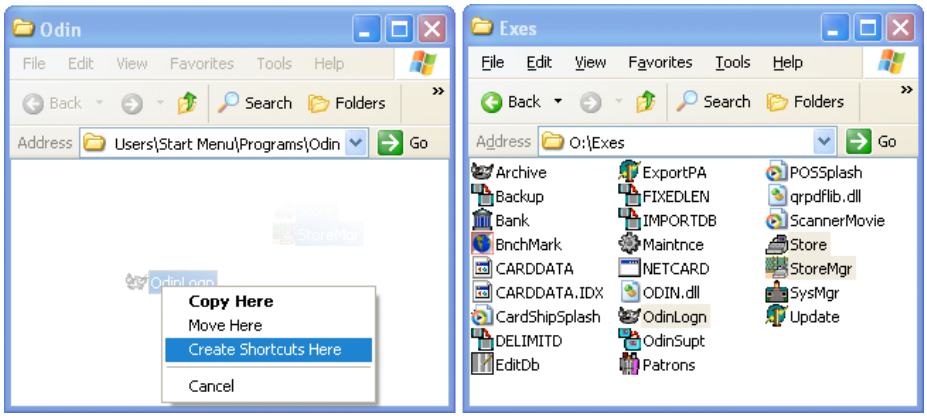
We must now make the programs available to the users on this computer. We must link executables from the **O:\Exes** folder to the local machine.

First, locate the executables in **O:\Exes** that you would like. Typically, most machines use **OdinLogn**, **StoreMgr**, and **Store**.

For each of these files, make shortcuts in the following two folders:

```
C:\Documents and Settings\All Users\Start Menu\Programs\Odin
```

```
C:\Documents and Settings\All Users\Desktop
```



Quick Test

At this point, Odin is correctly installed on the machine, and should be useable by the local Administrator. Try running **OdinLogn** and logging in with a valid Odin username and password. If that succeeds, run **StoreMgr** and confirm that you have access to the inventory files.

If that works, then you're ready to move on to the final step. If you have problems, go back and confirm that the installation was done correctly, or contact Odin Support for assistance.

Set Directory Permissions

Before Odin will work for other users on this machine, the permissions of a few files must be altered.

Follow the the steps in the section below for each of the following files and folders:

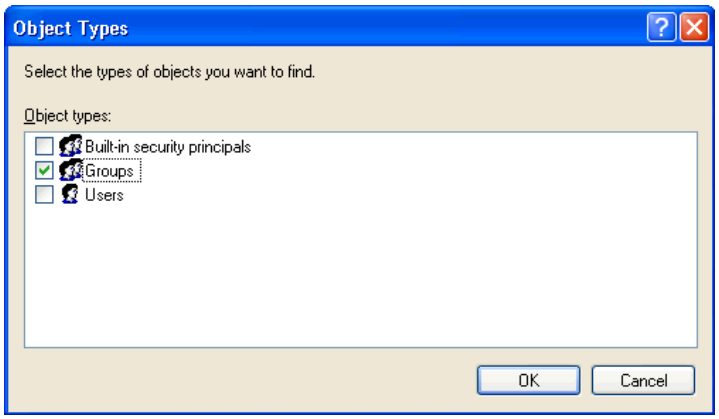
1. `C:\Program Files\Odin`
2. `C:\Program Files\Common Files\Borland Shared`
3. `C:\Odin`

4. `C:\PDOXUSRS`

Right-click the file and select **Properties**. In the window that appears, click the **Security** tab. Click the **Add...** button.

In the window that appears, click the **Advanced...** button.

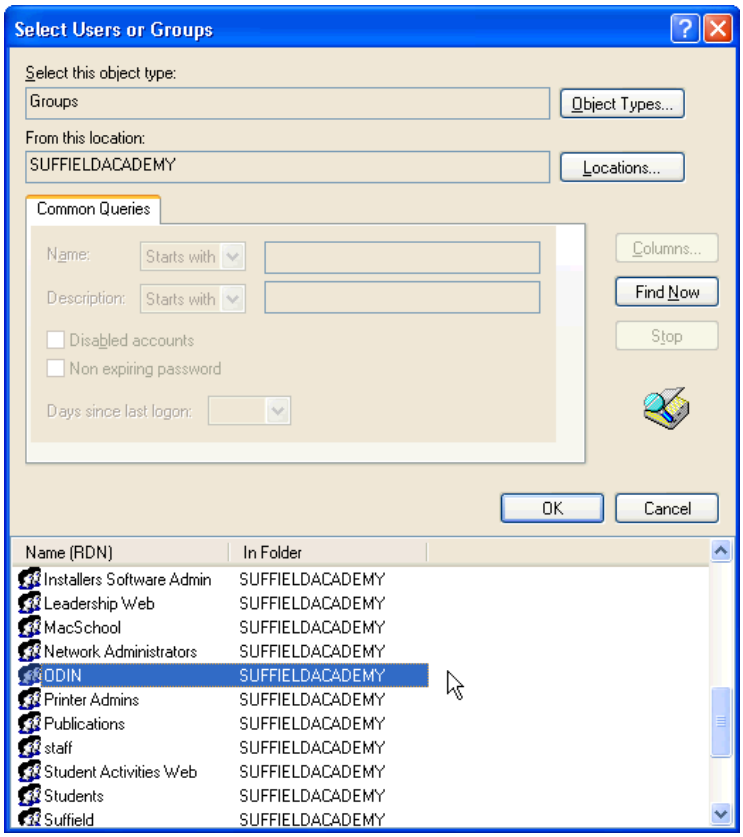
In the window that appears, click the **Object Types** button. In the next window, *deselect* all choices except for **Groups**. Click **OK**.



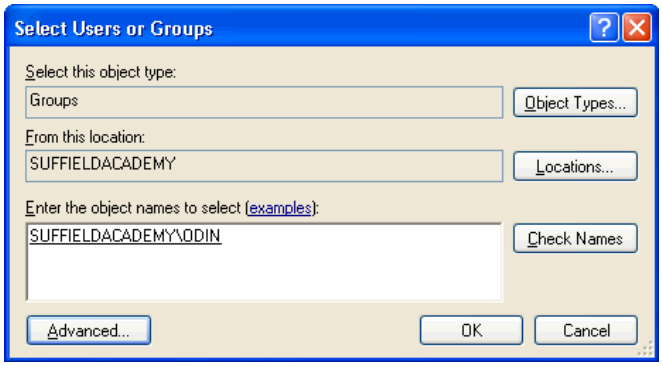
Confirm that the **Locations...** reads `SUFFIELDACADEMY` (change it to `SUFFIELDACADEMY` if it does not).

Click the **Find Now** button. The system may prompt you for a Domain Admin username and password. Enter it if requested:

The system will produce a listing of all the groups in the `SUFFIELDACADEMY` domain. Select the `ODIN` group and click **OK**.

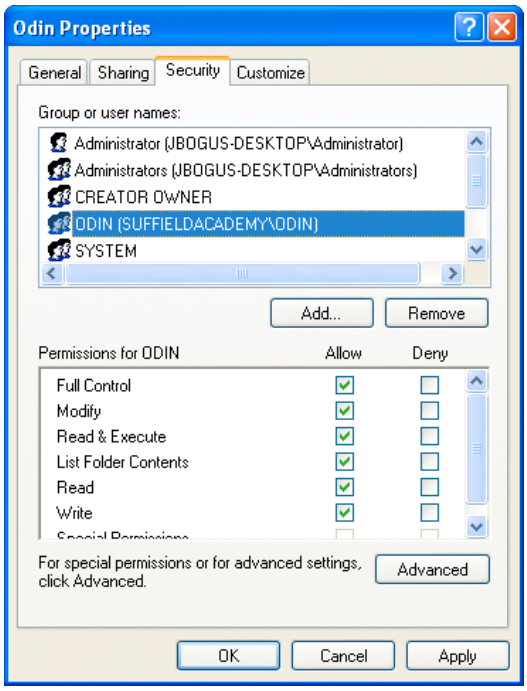


You should now return to the **Select Users or Groups** window, with the group listed in the selection box:



Click **OK**. You should be returned to the **Properties** window, and a new selection should be added to the **Group or user names** box called `ODIN (SUFFIELDACADEMY\ODIN)`. Click on this group.

In the box at the bottom of the window, *select* the **Allow** checkbox for **Full Control**. Your window should now look like this:



Click **OK** to save your changes and close the window.

Repeat these steps for all of the files and folders named above.

Wrapping Up

If you logged in to Odin earlier to test the setup, right-click the Odin icon in the system tray and choose **Exit**.

Finally, if a user of the computer needs to be added to the **Odin** group on the file server, please notify the Network Administrator. Ask them to add the user, and to change their login profile to automount the Odin drive on startup.

Chapter 4

Installing ODIN

Last updated 2008/03/18

4.1 Introduction

Virtual Private Networks (or **VPN**) are used to set up secure communications between hosts on a public network (such as the Internet). There are many ways of accomplishing this, and several technologies that facilitate it.

This document deals with using VPN software at Suffield Academy. We provide VPN servers so that faculty and students may connect from off-campus through our firewall and access services that normally would be inaccessible to them. For example, users can connect to our file server (to access their documents), or to the database server (to view records).

At the moment, we provide both L2TP and PPTP VPN services, which are compatible with the majority of VPN clients.

Before you begin, please make sure you know your **Network Username** (such as "jbogus" or "01abc"), and the corresponding password. You'll need this information in order to connect through our VPN server.

4.2 Mac OS X

(*This section not yet completed*)

This section will detail downloading the Internet Connect configuration file,

installing it, and connecting to the VPN.

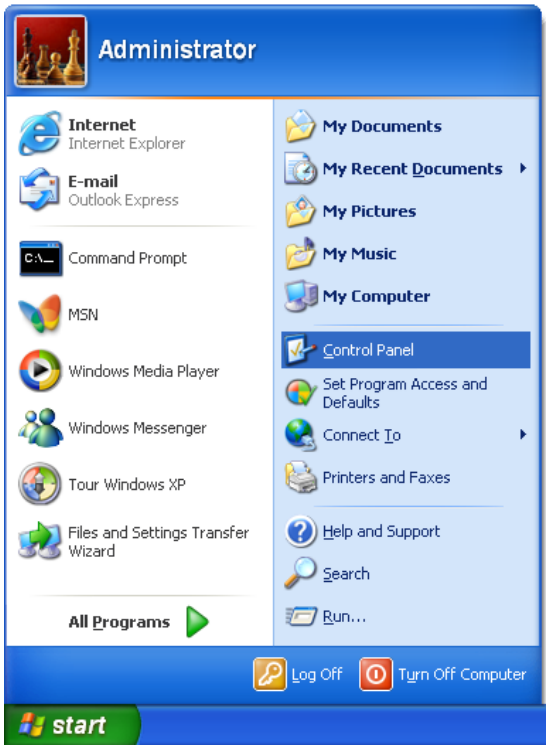
4.3 Windows XP

Windows XP comes with a built-in VPN client that can be used to connect to our network. To set up this client, follow the steps below.

Note that you only need to set up the client once. After the initial setup, you can easily connect using your saved settings.

4.3.1 Configuring the Client

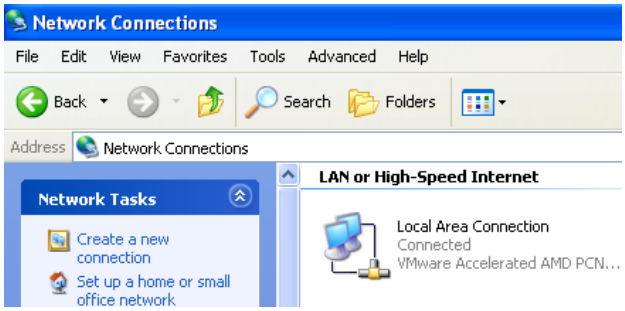
Begin by going to the **Control Panel** in Windows XP.



If you see a choice that says **Network and Internet Connections** and looks like the image below, click on it.

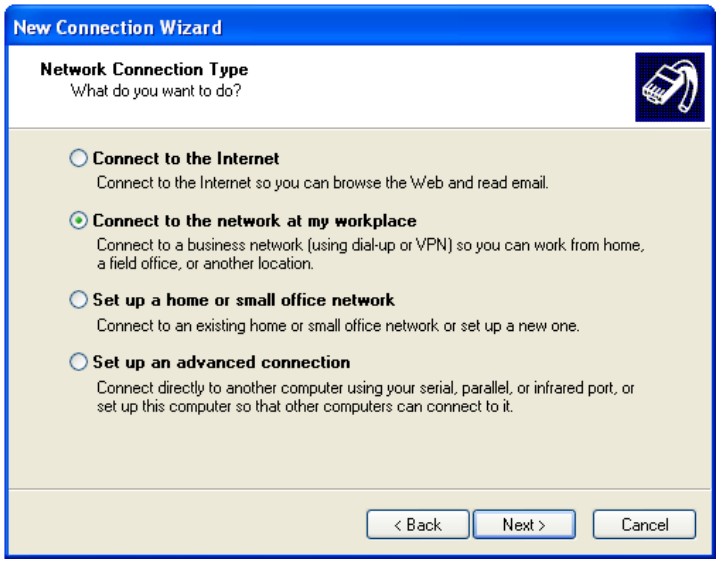
Now, open the **Network Connections** option:

You should now be at a screen that looks like this:



Click on the **Create a new connection** choice on the left. You'll be shown the following screen:

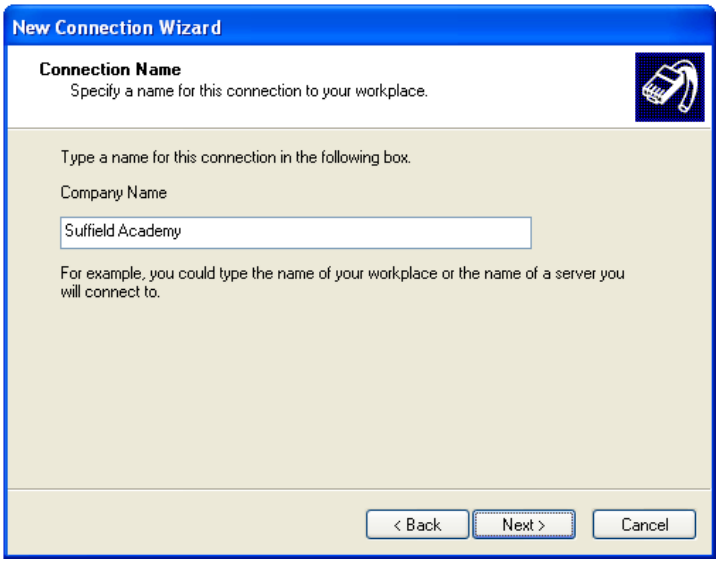
Choose **Next**. You'll now be given a list of choices.



Select **Connect to the network at my workplace** and click **Next**. You'll now be asked to choose between dial-up and VPN:

Choose **Virtual Private Network connection** and click **Next**.

You'll now be asked to enter a company name. Enter anything you like (for example, "Suffield Academy"):

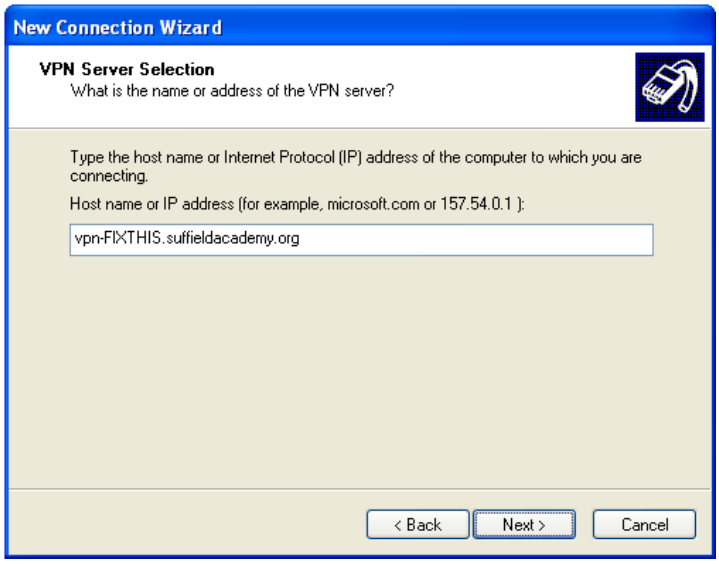


Click **Next**. You will now be asked to enter the VPN server. You must enter a server depending on whether you are a student or a faculty member.

Students must enter `vpn-student.suffieldacademy.org` as the address to connect to.

Faculty must enter `vpn-faculty.suffieldacademy.org` as the address to connect to.

Note: you must enter the proper address for the type of user you are, or it will not work!

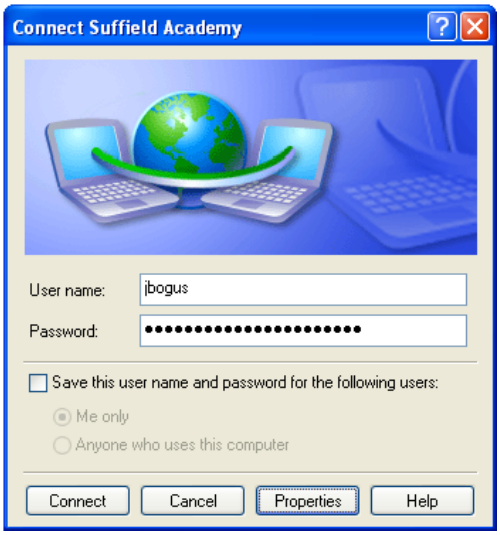


Click **Next**. The system will confirm the addition of the connection, and you may now click **Finish**.

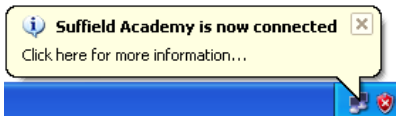
4.3.2 Using the VPN

In the **Network Connections** control panel, double-click your new VPN settings icon:

The connection screen will ask you for a username and password. Enter your network username and password, and then click the **Connect** button.

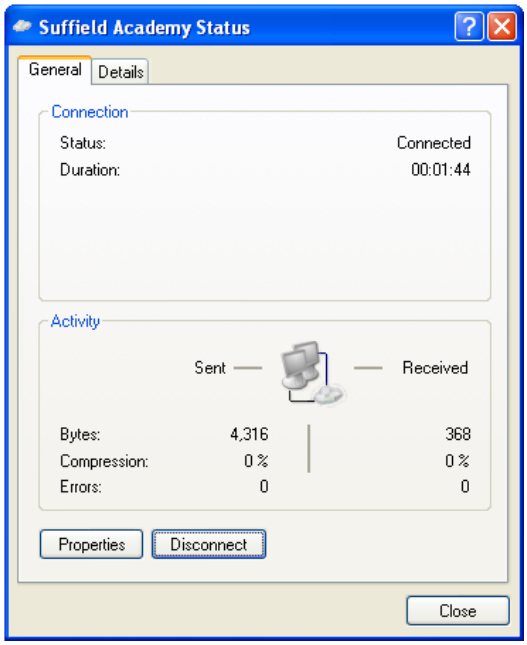


The system will connect to the VPN server. Upon completion of the connection, a small popup window will appear to alert you that the network is ready for use:



You may now work on the Suffield Network as if you were on campus.

When you're done working, click on the small computer icon in the system tray to get the status window for the connection:



Click the **Disconnect** button and the VPN will terminate. You are now disconnected and may use your machine normally.

Part II

Software

This section contains documentation for any programs or scripts developed in-house for system administration purposes. In most cases source code is also included.

This section also includes documentation of any third-party software used for network administration or infrastructure. Examples include mail services, web services, LDAP, DHCP, and DNS. In most cases, the documentation includes a brief overview of the service along with sample configuration files.

Chapter 5

Netboot Services

Last updated 2008/03/18

5.1 Introduction

All recent models of Macintosh computer have the ability to be **NetBooted**, where the computer boots its operating system off of a special server on the network. This approach has several advantages in a managed setting, including the ability to manage software and OS settings in lab environments and not needing to clean public machines manually (as changes are lost after reboots).

Here at Suffield, we do not have many public labs (each student has their own laptop). We primarily use NetBoot as a "rescue disk" for computers. By booting off of the network, the internal hard drive can be examined, repaired, erased, or reimaged with fresh software. Our NetBoot image contains various file repair utilities, as well as installers and system imaging software.

This document describes how to set up a NetBoot server that will service clients. Additionally, we describe how to build a "rescue disk" to serve to clients for repairs and system imaging.

Note: we mainly use our NetBoot image for booting damaged machines and repairing or reimaging them. Reimaging is done using a program called **NetRestore**. While we discuss the installation of NetRestore in this document, it only covers the use of the client. For more information about setting up NetRestore on the server, and for information on building images to install on client machines, see our HOWTO on creating NetRestore images.

5.2 Configuring Netboot Services

To provide NetBoot services to your network, you'll need a machine running Mac OS X Server. These instructions assume version 10.4, though they should work equally well for 10.3. (10.2 is significantly different, however, so these instructions will **not** work for that version.) You should use an **unlimited** client license version of OS X, or else you will only be able to boot a limited number of machines simultaneously.

Please refer to Apple's documentation regarding the specifications of the server hardware itself. Depending on the number of clients, image size, network speed, and other factors, you may need to scale up your hardware. As our server is lightly used (fewer than 15 clients booted simultaneously), we get by with a G4 500Mhz machine with mirrored IDE drives for storage and SCSI for client data.

5.2.1 Initial Setup

Begin by installing and configuring a machine running Mac OS X Server. Disable any services you will not be using. At a minimum, however, you must be running **AFP**, **DHCP**, and **NetBoot** to run a NetBoot server.

(**Note:** on Mac OS X Server 10.3, the **DHCP** service must also be enabled, even if it serves no actual addresses. This bizzare requirement has been dropped as of 10.4.)

5.2.2 AFP Settings

Apple automatically configures a few sharepoints via AFP when you enable NetBoot. You should not alter or remove these sharepoints at any time.

You do not need to make any special changes to AFP to use NetBoot by itself. However, if you plan to use NetRestore for system imaging, you will need to create a sharepoint for the saved disk image files. Create a folder (or volume) on the server, and share it via AFP. You may wish to create a new user account that has access to this share, and disable guests (this prevents people from getting access to the raw image files). You can use the **Workgroup Manager** tool for these tasks.

5.2.3 NFS Settings

In **Server Admin**, click on the **NFS** item. Confirm that the NFS service is running. If you plan to have a heavily loaded server, you may wish to increase the number of server daemon processes.

5.2.4 NetBoot Settings

In **Server Admin**, click on the **NetBoot** item. You should see an overview status of the other services that NetBoot depends on. Ensure that these services are shown as running (except DHCP, if your network already has a DHCP server).

Click on the **Settings** tab at the bottom of the window. Under the **General** tab, you'll have a few choices on how to store client data and images. Note that "images" here refers to images that the clients will boot off of, not system restore images.

Check the box(es) for the drives you wish to use for the different types of data. If possible, split up the data between drives (this helps with speed).

At this point, you don't have any images to serve via NetBoot, so there are no other settings to change. See the next section for information on creating a NetBoot image that clients can start up from.

5.2.5 Network Settings

If you're booting your clients on the same subnet as the server, you should be set to go. However, if you're going to be booting accross subnets, you'll need to do a little more work.

Because NetBoot discovery requests are sent from the client using DHCP, packets from the client must be forwarded on to the server. If you have Cisco equipment, you must use the `ip helper-address` statement in your router configuration to forward the packets.

For example, if your server is on VLAN 10 with IP address 172.16.10.100 and your client is on VLAN 20, your configuration should look something like this:

```
interface Vlan10
  description Server VLAN
  ip address 172.16.10.1

interface Vlan20
  description Client VLAN
  ip address 172.16.20.1
  ip helper-address 172.16.10.100
```

This tells the router to forward broadcast packets accross VLANs to the address you specify. By default, DHCP packets are forwarded, along with other common broadcast traffic. See your network equipment manuals for more information on the default forwarded ports.

Additionally, you must ensure that you are not blocking any traffic between the clients and the server. NetBoot images are served via TFTP, AFP, HTTP, or NFS, so these ports (and any "return" ports for protocols such as NFS) must be open. If your setup doesn't seem to be working, try opening all ports to confirm that the problem isn't networking-related.

5.3 Building a Rescue Image

Using NetBoot, we can create a "rescue disk" that can boot client computers that are damaged or that need system software installation. This is a simple way to keep all your system utilities in a single place, and makes repairing and restoring systems very easy.

5.3.1 Selecting a Machine

To build your rescue image, you'll need a machine to install the software on. We'll call this machine the **master** machine.

Your master machine should be the best computer available to you. Macintosh computers will often run systems from computers that are more recent, but the reverse is not always true.

You may wish to use an external drive to build the system image. This prevents you from having to erase a production machine, and makes loading the image onto the NetBoot server very easy. Alternately, you may prepare the image on a machine's internal hard drive, and then boot it into Target Mode to transfer the image.

You will be installing a system from scratch onto this machine. Make sure you've backed everything up, in case something goes wrong.

5.3.2 Base Installation

OS X Installation

Begin by booting the master machine with the latest installation media you have.

You should choose a full **Erase and Install** option from the installer to ensure that you do not have any leftover cruft from the previous system.

Additionally, you should choose to perform a **Custom Install**. On the customization screen, deselect any options that are not necessary. This includes

Additional Fonts, **Language Translations**, and **Additional Applications**. These things all take up space, and are not needed for our repair image.

When the installation completes, you will be brought to a confirmation screen. Do **not** continue yet.

Enabling The Root Account

While still booted from the installation DVD, choose **Reset Password** from the **Utilities** menu.

In the Reset Password Utility, select the hard drive you just installed OS X onto. Then choose the **System Administrator (root)** account from the drop-down menu.

Set the password for the root account to our standard password.

Quit **Reset Password**.

Disabling Registration

While still booted from the installation DVD, choose **Terminal** from the **Utilities** menu.

For the next step, you must know the name of the drive you installed OS X onto. Typically, the drive will be called "Macintosh HD". In the steps below, replace "Macintosh HD" with the actual name of the drive.

Type the following command (all on one line, capitalized exactly as shown):

```
touch /Volumes/Macintosh HD/private/var/db/.AppleSetupDone
```

Type `exit` on the command line, and quit Terminal.

Initial Boot

Now continue the installation DVD steps, which should cause the computer to restart.

Log in as "root", with the password you specified earlier.

If you are prompted to register the machine, simply quit the registration application.

5.3.3 System Configuration

OS Updates

Run **Software Update** and install any pending updates. Reboot as necessary, and continue running until no further updates are pending.

Resources Folder

Create an alias to the "Resources" folder on Veronica on the Desktop.

Finder Preferences

Show **Connected Servers**.

New Finder windows should open **Applications**.

Sidebar should only show **Hard Disks**, **External Disks**, **CDs, DVDs, and iPods**, **Connected Servers**, **Home**, and **Applications**.

Show all file extensions.

Disable Spotlight

Spotlight is a resource hog, and not useful in a repair image. You can disable it by running the following commands:

For Leopard (10.5):

```
launchctl unload -w /System/Library/LaunchAgents/com.apple.Spotlight.plist
launchctl unload -w /System/Library/LaunchDaemons/com.apple.metadata.mds.plist
```

For Snow Leopard (10.6):

```
launchctl unload -w /System/Library/LaunchAgents/com.apple.metadata.mdwrite.plist
launchctl unload -w /System/Library/LaunchDaemons/com.apple.metadata.mds.plist
chmod 600 /System/Library/CoreServices/Search.bundle/Contents/MacOS/Search
```

Disable Bonjour

Additionally, we don't want to allow bonjour requests to or from netbooted machines. Add the following firewall rules to deny Bonjour traffic in /etc/rc.local:

```
/sbin/ipfw add 1000 deny udp from any to any dst-port 5353 out
/sbin/ip6fw add 1001 deny udp from any to any 5353 out
```

Disable Safe Sleep

Safe Sleep keeps a large file (the size of physical RAM) on the boot volume to allow for hibernation. This is a huge performance hit, so we disable it for our repair image.

Add the following to /etc/rc.local

```
pmset -a hibernatemode 0
```

"Reserved" Space

Under 10.6, System Image Utility shrinks the avilable netboot image size to be only slightly (1GB) larger than the amount of space required by the files on the image. This can cause "out of disk space" warnings.

To combat this, we create a large file on the hard drive (10GB):

```
dd if=/dev/zero of=/private/var/vm/suffield-reserved-space bs=1g count=10
```

Then, we add the following to /etc/rc.local:

```
rm -f /private/var/vm/suffield-reserved-space
```

This has the effect of building the image with 10GB of space that will be freed up immediately on boot.

System Font Replacement

For some reason, our Netboot systems think the system fonts are "damaged" after System Image Utility has worked from a master machine. To prevent this issue from happening, open **Font Book** and go to its preference screen. There, **uncheck** the "Alert me when system fonts change" checkbox.

Then, rename the following directory to "ProtectedFonts.backup" (the lines below should be typed all on one line as a single path):

```
/System/Library/System/Library/Frameworks/ \
ApplicationServices.framework/Versions/A/Frameworks/ \
ATS.framework/Versions/A/Resources/ProtectedFonts/
```

Background Image

To easily identify a computer that has been NetBooted, it is helpful to have a special background image. Suffield has such an image, stored in the **Tech Repair** folder on the server.

To install an image on the master machine, copy it onto the computer and name it `DefaultDesktop.jpg`. Move the file into the folder `/System/Library/CoreServices/`, replacing any existing version.

Disabling Network Authentication

By default, our DHCP server advertises an LDAP server to all booted clients. This LDAP server helps with network authentication, servers, printers, and other centralized services.

Because the rescue image is not really multi-user, we don't need many of these authentication-related services. Additionally, they just slow down the operation of the system when not needed. Therefore, we disable these services for the rescue image.

To disable LDAP authentication, open the **Directory Access** program in `/Applications/Utilities`. Click the **LDAPv3** service and click the **configure** button to access the settings. In the window that appears, *deselect* the **Add DHCP-supplied LDAP servers to automatic search policies** checkbox. Click **OK** to save your changes.

Network Speed Tweaks

By default, Mac OS X comes with conservative networking settings that do not maximize performance over fast (100Mb/s and up) links. Since most of our NetBooted machines will be on a link that is at least this fast, we apply some tweaks to increase the performance of the networking stack.

To do this, edit (or create, if it does not exist) the file `/etc/sysctl.conf`. Add the following lines:

```
net.inet.tcp.mssdflt=1460
net.inet.tcp.sendspace=1048576
net.inet.tcp.recvspace=1048576
net.inet.udp.recvspace=65535
net.inet.udp.maxdgram=57344
kern.ipc.maxsockbuf=10485760
net.inet.tcp.newreno=1
net.inet.tcp.always_keepalive=1
net.inet.tcp.keepidle=3600
```

```
net.inet.tcp.keepintvl=150
net.inet.tcp.slowstart_flightsize=4
```

As always, you may wish to comment the lines to make future edits easier.

5.3.4 System Preferences

Open the **System Preferences** and make the following changes:

Appearance

Set appearance and colors to **Graphite**.

Desktop & Screen Saver

Set the desktop to the replaced DefaultDesktop you installed earlier.

Set the screen saver to **Computer Name**, and enable **Show with clock**. Have the screen saver start after 30 minutes.

Dock

Have the doc appear on the left-hand side of the screen, with size small enough to fit all the icons.

Expose & Spaces

Set the hot corners as follows:

- Top-left: none
- Top-right: Start Screen Saver
- Bottom-right: Disable Screen Saver
- Bottom-left: Desktop

Security

Uncheck all items under the **General** tab.

Displays

Check **Show displays in menu bar**.

Energy Saver

Set the **Computer Sleep** time to **Never**. Set the **Display sleep** time to **1 hour**.

Change the battery status menu to show estimated time.

Print & Fax

Add the **Multimedia Lab** printer.

Network

Disable the **AirPort** card (netbooted machines have a hardwired connection, so there's no need for Airport).

Sharing

Name the computer "Repair-and-Restore". Enable **Remote Login** and **Remote Management**.

Accounts

Set the computer to autologin to the root account, using the password you specified earlier.

Date & Time

Set the clock automatically to `ntp.suffieldacademy.org`.

Confirm that the time zone is set correctly.

Turn on showing the time with seconds.

Software Update

Disable checking for updates.

Time Machine

Turn off, and disable showing status in the menu bar.

5.3.5 Software Installation

Below we describe how to install the standard suite of repair software used by Suffield Academy.

DeployStudio

Download the latest stable version of DeployStudio:

http://www.deploystudio.com/

Launch the installer, and choose **Customize**.

Select only **DeployStudio Runtime** (plus any mandatory greyed-out options).

Install the software.

Add **DeployStudio Runtime** from the **Utilities** folder into the dock.

Run **DeployStudio Admin** and set the server and login credentials. Set the password to be saved for future use. The server address is:

```
http://veronica.suffieldacademy.org:60080/
```

Start **DeployStudio Runtime** and confirm that the credentials have been saved correctly. Log in and confirm that everything is working as planned.

DiskWarrior

Copy DiskWarrior from the original media into the **Applications** folder on the master machine. Add it to the dock.

Launch the program once to ensure that it is correctly installed.

DataRescue

Copy DataRescue II from the original media into the **Applications** folder on the master machine. Add it to the dock.

Launch the program. On the first time through, it will prompt you to activate the registration for the software. Enter in the correct information and quit the program when it has been registered.

TechTool Pro

Run the TechTool Pro installer from the original media and install it onto the master machine. Start the program and register it properly. If any updates are available, install them as well.

Reboot the machine, and **disable** the active protection in the Tech Tools system preference.

memtest

Mount the memtest distribution folder and copy it to `/usr/bin`.

rsync 3 (patched)

Compile and install a patched version of rsync 3 with ACL/metadata support and put it in /usr/local/bin.

Firmware Password Utility

This program may be found on any Mac OS X installer disk. It is used to lock or unlock the firmware on the computer.

System Image Utility

This program is included on Mac OS X server machines, and is used to create Netboot sets. We put it on our image to make it easier to create Netboot sets from any machine.

Safari

Show the status bar.

Show the tab bar.

In the preferences, set the default home page to:

```
http://web.suffieldacademy.org/ils/crc/
```

Save downloaded files to the Desktop.

For Bookmarks, only include Bonjour, and disable all collections.

Set RSS never to update.

Disable all forms AutoFill.

Clear the history, empty the cache, and quit.

Terminal

Set the Terminal to "Pro".

Set dimensions to 80x40.

Set the window to close when the shell exits cleanly.

5.3.6 Performance Tweaks

Deleting Unused Files

To save space on the image, you should delete any applications and files you know you will not need. Good candidates include the **iLife** suite, any games, obscure utilities (**ColorSync**, **ODBC**, etc), screen savers, background pictures, sample media, and developer tool samples.

Delete /private/var/vm/sleepimage

Download and run Monolingual (monolingual.sourceforge.net) to remove all but English localizations from the machine.

5.3.7 Building the Image

At this point, you should have a disk with a fully-functional NetBoot image on it. You must now connect this disk to a machine with Apple's **System Image Utility** installed on it (it is included with Mac OS X Server).

The simplest way to do this is to connect the master image directly to the NetBoot server via firewire. If your image is on a firewire drive, simply connect it. If your image is built directly on a master machine, boot the machine into firewire target disk mode and connect it. Then:

1. Perform any last-minute housekeeping (deleting the files in `/var/vm/`, cache files, etc).
2. Start **System Image Utility** on the server.
3. Choose **New Boot** from the toolbar.
4. Give your new image a name, ID, and description.
5. Under the **Contents** tab, select your master image disk.
6. Click the **Create** button. Save the image to your NetBoot images folder on the server (or elsewhere, if you wish to move it later).

5.3.8 Installing the Image

If you used **System Image Utility** to create an image directly into your NetBoot server folder, then the image is installed and ready to be used.

If you saved the image elsewhere, you must copy it into the **SPxxx** folder on your NetBoot server. The folder name varies depending on how many volumes you have enabled to host NetBoot images. In most cases, the folder is called `SP0` and is located in `/Library/NetBoot/` on the main drive.

If you want this image to be the default NetBoot image, use **Server Admin** to set this image to be the default.

5.3.9 Testing the Image

The moment of truth! NetBoot one of your client machines to your new image and test out the software.

If your client machines won't NetBoot correctly, confirm that there are no firewall or ACL problems between the client and server machines. Recall that a proper NetBoot requires DHCP, TFTP, NFS, and AFP to work properly. Here are the symptoms of one of these protocols not working:

- **DHCP** Machine will not boot at all. Other machines on subnet won't get IP addresses assigned to them.
- **TFTP** Machine will not get to "spinning globe" stage of NetBoot (only flashing globe icon). TFTP is needed to send the initial booter file to clients, so if your boot fails early, check the TFTP service on the server.
- **NFS** NFS is needed to mount the NetBoot image and complete the boot process. If you get to the "spinning globe", but freeze up afterwords, check to make sure you're passing NFS traffic over UDP correctly.
- **AFP** Diskless NetBoot requires AFP be enabled on the NetBoot server, so clients can connect and use space for temporary files. Non-diskless boots do not require AFP. If AFP is not working properly, diskless boot will either fail, or will not allow you to unmount the local hard drive.

If your clients boot, test the software and confirm that they are all properly running and registered. Once that's done, you're all set!

Chapter 6

OS 9 Remote Shutdown

Last updated 2008/03/18

6.1 Introduction

We use NUT to manage our UPSes and automated shutdown of servers (see our NUT documentation for more information). However, NUT has not been ported to run on Mac OS 9, so our older servers would not automatically shut down when a power failure took place.

To work around this issue, we built a small set of scripts that send remote AppleScript events to a Mac running OS 9. While basic, these commands provide enough functionality to shut a machine down remotely.

While these scripts focus on power failure shutdowns, they are very basic and easily adapted to other uses. See the code in the following sections for more information.

6.2 Requirements

The full shutdown package relies on three main components:

1. A machine running Mac OS 9 (9.2 is assumed; earlier versions should work). **Note:** Mac OS versions 8 and below do **not** support these scripts; program linking only works over AppleTalk in these systems, but Mac OS X only supports program linking via IP, so they are not compatible. This

machine is the **target**; it will be shut down when it receives a remote command. AppleScript support must be enabled on this machine.

2. A machine running Mac OS X (10.3 or 10.4 is assumed; 10.2 may work). This machine is the **source**; it will send the remote shutdown command to the target.

3. A **trigger** program that will cause the remote command to be sent. In our case, the UPS software triggers the remote shutdown command, but any scriptable event can act as the trigger.

   The **source** and **target** machines talk to each other via TCP/UDP port 3031 (eppc), so any firewall software must be configured to allow traffic *from* the source *to* the target.

6.3 Configuring the Target

To configure the target, we must install a few scripts on the system volume of the machine. We must also configure the system to receive and process remote AppleEvents.

6.3.1 Installation of Scripts

In the `scripts/target` directory of the project, you will find two scripts:

1. `CommTest.txt`
2. `ForcedShutDown.txt`

Copy these scripts onto the **target** machine (the one running Mac OS 9).

On the **target** machine:

1. Create a folder at the root level of the system hard drive called `RemoteShutdown`.
2. Open these scripts using the `Script Editor` application that comes with OS 9. You may need to open the scripts from within the application, or by dragging the scripts onto the application's icon.
3. For each script, choose **Save As Run-Only...** from the **File** menu. For the name, use the script's existing name, omitting the `.txt` extension (*e.g.*, `CommTest` and `ForcedShutDown`).
4. For the format, choose **Application**. Leave the **Stay Open** check box *unselected*. Make the **Never Show Startup Screen** check box *selected*.

5. Save the compiled script into the `RemoteShutdown` folder you created at the root level of the hard drive.

6. Run each compiled script by double-clicking on it. You should receive a dialog box with buttons for each script. Dismiss the dialog box when you get it.

You have now compiled the scripts correctly. Quit `Script Editor`. If you wish, you may delete the original script files (only the compiled versions are needed for regular operation).

6.3.2 Receive AppleEvents

The **target** machine must be configured to receive remote events from other computers:

1. Begin by opening the `File Sharing` control panel on the target.

2. Select the **Start/Stop** tab, ensure that **Program Linking** is *on*, and that the **Enable Program Linking clients to connect over TCP/IP** check box is *selected*.

3. Switch to the **Users & Groups** tab, and click on the **New User** button.

4. Choose a name and password for the new user. You may want to *deselect* the **Allow user to change password** check box. **Note:** passwords in OS9 must be **shorter** than 8 characters. Additionally, "odd" characters may not work, as we supply the password as part of a URL.

5. Change the popup menu from **Identity** to **Sharing**.

6. The **Allow user to connect to this computer** check box should be *deselected*.

7. The **Allow user to link to programs on this computer** check box should be *selected*.

Make note of the IP address (or DNS name) of this machine, the username you created, and the associated password. You'll need these items later to connect to the machine and execute the shutdown scripts.

Note: if you will use a single source to shut down multiple targets, you may wish to use the same username and password for each target. There are security implications to having the same username and password for each machine, but it also makes sending commands to a large group of machines much easier.

6.4 Configuring the Source

The **source** machine (running Mac OS X) requires no system configuration to work with the targets. The source simply needs to fire the remote AppleEvent and send it to the **target** machine over the network.

The simplest way to fire the event is by using a command-line AppleScript. Open a terminal on the source machine and execute the following (the command should be input as a single line):

```
osascript -l AppleScript -e "tell Application \"Finder\" of machine
\"eppc://username:password@hostname\" to open file \"CommTest\"
of folder \"RemoteShutdown\" of startup disk"
```

(A copy of this command lives in the `scripts/source` directory for this project.)

You must replace **username**, **password**, and **hostname** with the specific values for your target machine. The username and password are set in the **File Sharing** control panel on the target, and the IP address is set in the **TCP/IP** control panel.

Upon running the command, the target machine should launch the **CommTest** script and preset a dialog box. If it doesn't, the source machine should return an error message to help diagnose the problem.

Once you have the sample script working, you may try the **ForcedShutDown** script. If you are installing on a production machine, be careful not to shut it down accidentally!

6.5 Configuring NUT as a Trigger

We designed the remote shutdown script as part of our automated shutdown process during a power failure. This section describes how to integrate the scripts with NUT, our UPS management software. Note that the principles discussed here can be adapted to other uses with minimal effort.

6.5.1 Setting Up NUT

To use NUT as a trigger, NUT must be configured to trigger an `upssched` script when power-related events occur.

Configuring NUT is beyond the scope of this document; please refer to our documentation for setting up NUT for more information.

Before proceeding, you should have NUT configured to call `upssched` for `ONBATT`,

ONLINE, and FSD events. In turn, upssched should invoke a script on the system that correctly parses these events. We'll be modifying this script to add support for calling the remote shutdown scripts.

6.5.2 Sample Configuration Files

As an example, the config files from our stock NUT config are shown below. We'll be working from these configurations in the sections below:

upsmon.conf

The upsmon.conf file must include support for calling upssched when power events occur. This means including the following lines:

```
NOTIFYCMD /usr/local/nut_upsmon/sbin/upssched

NOTIFYFLAG ONBATT WALL+SYSLOG+EXEC
NOTIFYFLAG ONLINE WALL+SYSLOG+EXEC
NOTIFYFLAG FSD WALL+SYSLOG+EXEC
```

You may need to modify paths or modify lines to your liking. The lines above are *an example of the minimum required configuration.*

upssched.conf

The upssched.conf file tells NUT what script to call during power-related events, and which events to listen for. You can configure an event to trigger immediately (EXECUTE), or to wait a certain amount of time (START-TIMER).

In the sample below, we start a timer during on-battery events, but execute immediately when we receive a forced shutdown request (*e.g.*, when the battery is almost exhausted).

```
CMDSCRIPT /usr/local/nut_upsmon/bin/early-shutdown

AT ONBATT * START-TIMER early-shutdown 120
AT ONLINE * CANCEL-TIMER early-shutdown
AT FSD * EXECUTE forced-shutdown
```

early-shutdown

The script `early-shutdown` must handles the commands from `upssched` (*e.g.*, `early-shutdown` and `forced-shutdown`) and makes the calls to any additional scripts. This is where we add our AppleScript code to shut down other machines.

Below is the stock `early-shutdown` script from our NUT package. At the moment, it simply shuts down the local machine. We'll be building on this script to add the code to shut down other machines.

```
case "${1}" in

  early-shutdown)
    logger -t early-shutdown "Early Shutdown time has arrived!"
    /usr/local/nut_upsmon/sbin/upsmon -c fsd
  ;;

  forced-shutdown)
    logger -t early-shutdown "UPS Master sent Forced Shutdown Request"
    /usr/local/nut_upsmon/sbin/upsmon -c fsd
  ;;

  *)
    logger -t early-shutdown "Unknown command: ${1}"
  ;;

esac
```

6.5.3 Creating a Remote Shutdown Script

We'll assume that we need to shut down multiple target machines from a single source. To do so, we'll write a script that keeps a list of machines that must be shut down, and then sends each machine the proper commands.

We have a sample script, called `shutdown_os9`, which is included in the `scripts/source` directory of this project. You should customize it for your needs.

Because this script contains usernames and passwords, its permissions should be set to only allow access to the NUT user (`ups_mon`, in this case):

```
chown nut_upsmon shutdown_os9
chmod 700 shutdown_os9
```

6.5.4 Calling the Remote Shutdown Script

To actually invoke the shutdown script, we must add it to the script that `upssched` calls. In our NUT package, that script is called `early-shutdown`. We simply add the call to the script in the right places, as shown below. We have a sample script, called `early-shutdown-with-remote`, which is included in the `scripts/nut` directory of this project.

Note that NUT must be correctly configured to call this script via `upssched`. See our NUT documentation for more information on configuring NUT.

Chapter 7

FirstClass Scripts

Last updated 2008/03/18

7.1 Introduction

Suffield Academy uses FirstClass 8.0 as it's e-mail server. We currently run it on Mac OS X, which has improved stability and reliability over First Class 7 under Mac OS 9.

First Class 8 (FC8) supports the ability to script certain actions via the command line (start, stop, restart, mirror pause, etc). Additionally, the standard control scripts support additional hooks for custom behaviors when the scripts are run.

Suffield uses both of these features to add custom behaviors to our FC8 server. We use the scripting to execute regular unattended backups of the First Class Network Store, and we use the script hooks for automatic maintenance tasks (such as log rotation).

7.2 Preflight Scripts

FC8 has hooks to execute preflight shell scripts whenever the user invokes the standard server control scripts `fcsctl` or `fcisctl`. These control scripts look in a specific location for a files called `prefcsd` and `prefcisd`, respectively. If they are found, the control scripts call them using the same arguments passed to the control script. When the preflight script returns, the control script resumes control and performs the operation requested by the user.

Because these preflight scripts execute entirely before the regular control script, they can be used to perform additional checks or setup for the FC8 daemons.

We have written a set of generic hook scripts, which can be used to call other scripts as needed. This additional layer of abstraction helps to find and diagnose problems, and also consolidates common code in external scripts.

7.2.1 The Preflight Scripts

The **preflight scripts** are called directly by the FC8 control scripts. They must have a specific name, and reside in a specific location, in order for FC8 to find and execute them.

All preflight scripts must live in the **Documents** folder in the home directory of the FC8 admin user (`fcadmin`). The scripts' names must start with "pre", and end with the name of the service that it ties in to.

Therefore, there are two preflight scripts: `prefcsd`, which goes with the FC8 Core Services daemon (the main server), and `prefcisd`, which goes with the FC8 Internet Services daemon (for SMTP, HTTP, IMAP, and other processing).

Our preflight scripts basically contain a large case statement to determine which scripts to run based on the action requested by the user. Because of this, both `prefcsd` and `prefcisd` scripts are nearly identical, except in name.

You may download the current version of both `prefcsd` and `prefcisd` from our `scripts` directory.

You'll notice that the scripts look for one of the three supported actions: **start**, **stop**, and **restart**. Under each action, you may add any commands that you wish to execute when that action is requested. In most cases, the FC8 control script ensures that the requested daemon is running (or not running, depending on the action) before invoking the preflight script. Thus, you do not usually need to check on the status of the daemons before running your commands.

We have chosen to enclose all commands in external scripts (described below). For example, in the **start** stanza of the `prefcisd` script, we call our log rotation script. Every time the Internet Services daemon starts, the log rotation script executes. (For more information on this script, please see the section on our log rotator script.)

Following our example, all you need to do is write a script that performs the functionality you need, and add it to the proper stanza in the preflight scripts.

7.3 Postflight Scripts

FC8 has hooks to execute postflight shell scripts whenever the user invokes the standard server control scripts `fcsctl` or `fcisctl`. First, the control scripts perform the operations requested by the user. After the normal operation has completed, they look in a specific location for a files called `postfcsd` and `postfcisd`, respectively. If they are found, the control scripts call them using the same arguments passed to the control script.

Because these postflight scripts execute entirely after the regular control script, they can be used to perform cleanup or notification on the FC8 daemons.

We have written a set of generic hook scripts, which can be used to call other scripts as needed. This additional layer of abstraction helps to find and diagnose problems, and also consolidates common code in external scripts.

7.3.1 The Postflight Scripts

The **postflight scripts** are called directly by the FC8 control scripts. They must have a specific name, and reside in a specific location, in order for FC8 to find and execute them.

All postflight scripts must live in the **Documents** folder in the home directory of the FC8 admin user (`fcadmin`). The scripts' names must start with "post", and end with the name of the service that it ties in to.

Therefore, there are two postflight scripts: `postfcsd`, which goes with the FC8 Core Services daemon (the main server), and `postfcisd`, which goes with the FC8 Internet Services daemon (for SMTP, HTTP, IMAP, and other processing).

Our postflight scripts basically contain a large case statement to determine which scripts to run based on the action requested by the user. Because of this, both `postfcsd` and `postfcisd` scripts are nearly identical, except in name.

You may download the current version of both `postfcsd` and `postfcisd` from our `scripts` directory.

You'll notice that the scripts look for one of the two supported actions: **start** or **stop**. Under each action, you may add any commands that you wish to execute when that action is requested. In most cases, the FC8 control script ensures that the requested daemon is running (or not running, depending on the action) before invoking the postflight script. Thus, you do not usually need to check on the status of the daemons before running your commands.

At the moment, we do not make use of any postflight scripts (we use preflight scripts instead). However, to add postflight commands, all you need to do is write a script that performs the functionality you need, and add it to the proper

stanza in the postflight scripts.

7.4 FirstClass Network Store Backup

7.4.1 Background

FirstClass has a built-in mirroring capability that allows for a "warm" backup of the FirstClass system to exist at all times. We have chosen to mirror our FirstClass Network Store (FCNS) onto a second hard drive inside the server. The mirroring is quick, and does not suffer from external factors such as network availability.

That said, we do wish to guard against catastrophic failure of the entire server. We have written a script, `fcnsbackup`, which copies the entire FCNS off of the server and onto a backup server of our choosing. We use `rsync` for this task, so copies are quite fast (usually less than 15 minutes for a full synchronization).

In the sections below, we discuss the design of the script, its use, and how to recover using the backups it creates.

7.4.2 Requirements

This script synchronizes a **mirror** of the FCNS to a remote server. Therefore, the FirstClass server must have mirroring turned on (you cannot sync the main FCNS directly or data corruption may occur). Additionally, our script is written with the assumption that the mirror is locally attached to the machine running the FirstClass core server (if it isn't, then you probably don't need our script...)

You will need a remote host that can be reached over the network. This host must have `rsync` and an SSH server running on it (we tunnel the rsync connection over SSH for security). The remote host must have a directory that is writable by the username that we connect over SSH with.

The script is written for the **Bash** shell. At this writing, the script has been written and tested on machines running Mac OS X 10.3 (Panther). With the exception of file locations, the scripts should work for any Unix-like setup, and may even run under Windows with some minor modifications.

7.4.3 Design

If you'd like, you may view the current version of `fcnsbackup` from our `scripts` directory to see how it works.

Overall, what the script does is very simple:

1. Pause mirroring in FirstClass
2. Use rsync to transfer any changed files to the remote host
3. Resume mirroring in FirstClass

The script has some extra sanity-checking to prevent most basic problems. Briefly:

1. It creates a lockfile to prevent multiple copies of the script from running.
2. If interrupted or killed, the script attempts to "clean up" and restart mirroring in FirstClass.
3. If the synchronization process fails, diagnostic output is printed for analysis.

We use `rsync` to copy files to the remote host. rsync is great because it compares the files on both machines and only sends the bits that have changed. Due to the size of the FCNS, this comparison stage can take a while (10-20 minutes), but it is much faster than blindly copying all the files every time.

For security, we tunnel the rsync connection over `ssh`. To prevent the script from having to enter a password, we recommend setting up a password-less auth key (see the next section for more information).

The script is written such that user-customizable variables appear at the top, where they can easily be edited. The variables are commented, and should be familiar to anyone who has used rsync (if you are unfamiliar with rsync, we suggest running `man rsync` on the command line to view the manual for the program).

7.4.4 Usage

To use the script, copy it onto the server. We've chosen to place it in the **Documents** folder in the **fcadmin** account, as that's where FirstClass likes some of its other scripts to be kept.

SSH Keys

You'll need to set up a password-less SSH key to use for connecting to the remote server. You can do this using a command like this:

```
ssh-keygen -t dsa -N "" -C "Auto-Login Key" -f fcadmin_backup_autologin
```

You can change the part in quotes after `-C` to whatever description you want. Additionally, you can call the file whatever you want (in the example above, we call it `fcadmin_backup_autologin`).

When the key has been generated, there will be two files: one with the name you specified (the *secret* key), and one with `.pub` appended to the name (the *public* key). To use the keys, copy only the public key onto the remote host, and add it to the file `.ssh/authorized_keys` (create the file if it doesn't already exist). At this point, you should be able to log in to the remote host by typing:

```
ssh -i private_key_file user@remotehost
```

Where `private_key_file` is the name of your private key. For more information on using keyed auth with SSH, read the manual page for SSH or search the web for SSH key logins. Many extra options are available (for example, to restrict to the key to certain programs or IP addresses), but those topics are beyond the scope of this document.

Customization

Before running the script, you'll need to edit it and change the variables that appear at the beginning of the file. They are commented to describe their function, and are all self-explanatory. The `REMOTE_KEY` variable should be the path to the secret key file you created above.

Testing

At this point, you should be able to run the script from the command line to see if it works. Run the script, and make sure that you are not prompted for a password when you connect to the remote host.

You may wish to add `-nv` to the line that invokes rsync before you test. Doing so will print verbose information about what the script does, and additionally //will not actually change the remote machine//. Thus, you can see what would have happened, without worrying about messing something up (and without waiting for the data to transfer!).

When the script finishes, make sure that mirroring has turned back on in the FirstClass server.

When you're done, remove any debugging information from the script, and proceed to the next step.

Running Automatically

To have the script run automatically, you should add a line to `/etc/crontab` similar to the following:

```
7  0  *  *  *  fcadmin  /Users/fcadmin/Documents/fcnsbackup
```

If you're not familiar with `cron`, type `man crontab` on the command line for the manual page. The line above says to run the `fcnsbackup` script under the username `fcadmin`. The script is run every day at 12:07am (e.g., hour 0 and minute 7).

You should choose a time when your server is not under a heavy load, as the backup script requires a fair amount of processing power. Additionally, because the mirror is paused during backup, you want to try to backup during off-peak hours to reduce the amount of "catch-up" that the mirror must perform.

7.4.5 Recovering from Zombie Processes

The script takes steps to prevent a failed backup from causing problems down the line. However, it is possible that a badly timed disruption could prevent the script from running properly.

If you get a message like:

```
fcnsbackup: unable to get lock file from user ...
```

It means that another version of the script is still running, or that it didn't get a chance to clean up properly. To fix this problem, check to see if `rsync` is running on the machine:

```
ps auxwww | grep rsync
```

If no processes are listed, then the script was probably interrupted the last time it was run (perhaps due to a crash or power outage). You can remove the file `/tmp/lock.fcnsbackup` and the script should run once again.

If an active `rsync` process is running, you should investigate why. Are you running your backups too close together? Does it normally take this long? If you feel that the process has stalled, you can kill it using the normal `kill` semantics. The backup script should notice that it was interrupted, and exit cleanly. Confirm that the `/tmp/lock.fcnsbackup` file has been removed after you kill the process.

7.4.6 Restoration

Should the unthinkable happen, the instructions below should help you get the server up and running again.

First, you'll need to have a working server with the correct version of FirstClass Core Services installed on it. We keep a system image of the server in the **Network Administrators** share on the file server, along with directions for getting the server software reinstalled.

Once the OS and server are installed, you're ready to restore the FCNS. Because the backup script uses `rsync` to back up its files, we can just use rsync "in reverse" to copy the files back down.

To do this, you'll need the following:

1. The name of the remote machine you backed up to
2. The username on that machine that created the backup
3. The SSH key file or password for the account (if you don't have either, log in as an administrator and change the password to something you know).
4. The remote path where the files are backed up (e.g., `/Volumes/BigDisk/fcbackup/`)

With this information, you can restore the backup. On the FirstClass server, run a line similar to the following:

```
rsync -az --stats -e ssh "user@host:/path/to/fcns/" \
"/Library/FirstClass Server/Volumes/Master/fcns/"
```

Note that you should replace `user`, `host`, and `/path/to/fcns` with the appropriate values. When you run the command, you will be prompted for the password to the remote account.

You may add `-v` to the command line (right after the `-az`) if you'd like more verbose output. This will show you each file as it is copied, but will slow down the transfer somewhat.

We have split the command over two lines, and used a backslash (\) to tell the terminal to split the command. You may type it all on one line, if you wish.

At this point, you will have to wait a while (as of this writing, a full restore for 10GB of data takes 1-2 hours). If you did not add `-v` to the command, you will not see any status of the file transfer, so you just need to be patient. Your best bet is to run `top` in another window and keep an eye on the load average and running processes.

When the copy is complete, your FCNS should be completely restored. At this time, you **must** run the following command on the FirstClass server:

```
"/Library/FirstClass Server/fcfixvol all all"
```

This repairs the permissions on the FCNS and ensures that it's ready to run. When this is complete, you're ready to start the server. Take a deep breath, and then go for it!

7.5 Internet Services Log Rotate

7.5.1 Background

Under Mac OS 9, the FirstClass Internet Services module displayed a running event log as it processed mail. Under OS X, this functionality is no longer turned on by default. We had grown used to the log information, so we turned it on when we switched to OS X.

Unfortunately, doing so created a log file on the machine that, if left unchecked, would grow without limit. While we had plenty of space remaining on the disks, we opted to write a small script to rotate this log file on a regular basis.

We called this script `fcisdlogrotate`, and we describe its design and functionality in the sections below.

If you wish, you may download the the current version of `fcisdlogrotate` from our `scripts` directory.

7.5.2 Requirements

As written, the script requires that you run FC8 Internet Services with logging turned on. Additionally, you must have the `date` and `bzip2` utilities installed on your system (these come standard with Mac OS X 10.3, and most other Unix variants).

You may modify the script to use a different compression program (such as `gzip`) simply by modifying a global variable.

7.5.3 Design

The script simply looks for the current log file generated by Internet Services (`fcisd.log`). If found, it copies the log file to another directory, gives it a name

based on the current date and time, and compresses it to save space.

You may alter most of the script's behavior by modifying the global variables at the top of the script. You may specify which compression program to use, the initial location of the log file, and the destination directory to save the compressed files to.

7.5.4 Usage

To run the script automatically, copy it to the **Documents** folder of the `fcadmin` user, and add a line to the **start** stanza of the `prefcisd` script. See our section on preflight scripts for more information on the `prefcisd` script.

The script should run whenever you start or restart the daemon, and the logfile should rotate accordingly.

To rotate your logfiles regularly, just restart the Internet Services daemon on a regular basis. You can do this using the `cron` utility that comes built in to Unix.

Edit the file `/etc/crontab` and add the following line:

```
29   4    *    *    *    fcadmin    /usr/sbin/fcisctl restart
```

That tells the machine to restart `fcisd` at 4:29am every day. You may customize the time by modifying the fields at the beginning of the entry; see `man 5 crontab` for more information.

7.5.5 Viewing Old Logs

Old logs are stored in the destination directory specified in the script (`/var/log/fcisd/` by default). If your system has a copy of the `bzless` utility installed, you may use this to view the logs directly.

Otherwise, you should decompress the logs using the `bunzip2` utility, and then view the uncompressed logs with any text viewer.

Chapter 8

DNS

Last updated 2008/03/18

8.1 Introduction

The **D**omain **N**ame **S**ystem, or **DNS** performs the valuable service of translating human-readable names (such as `www.example.com`) into computer-usable IP addresses (such as `192.168.1.100`). Many modern network services require a properly functioning DNS setup in order to work correctly.

A full treatment of DNS is far beyond the scope of this document. This document assumes that you are familiar with the basics of DNS, including:

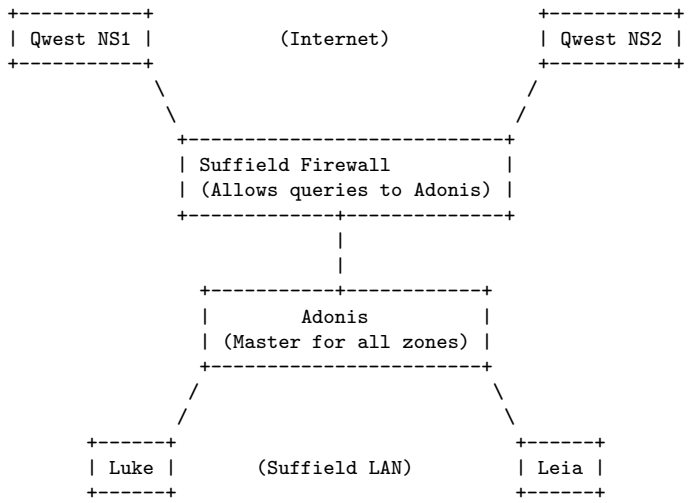
- What a nameserver is, including the difference between a **master** and **slave** nameserver
- How to query a nameserver
- Forward and reverse records
- SOA, A, CNAME, and MX records

If you need to brush up on your basics, please consult the "bible" of DNS: *DNS and BIND* by Paul Albitz and Cricket Liu (published by O'Reilly).

8.1.1 Suffield's Topology

Suffield currently uses 5 nameservers in its DNS topology. We have a single master DNS machine running on an Adonis DNS Appliance. Outside of our firewall, we have 2 slave nameservers that maintain our "public" DNS information (these servers are hosted with our ISP). Inside the firewall, we have 2 more slaves that maintain our "private" (internal) DNS information. These servers also act as the primary published servers for our LAN clients, and perform recursive and cached lookups for those clients.

Graphically, it looks something like this:



8.2 Master (Adonis) Configuration

Suffield currently uses an Adonis DNS Appliance as its master DNS server.

Because the Adonis is self-contained (it has its own GUI for updating zones), this document does not cover the setup of a BIND 9 master server "by hand". Please refer to a full DNS configuration guide for information on setting up a master server.

(In reality, a master server is very similar to a slave server, except for a few settings in how the zones are defined.)

8.2.1 Adonis Notes

The Adonis comes with a detailed manual which explains its features and use. The latest version may be obtained from BlueCat's website: www.bluecatnetworks.com.

Because the manual is so complete, we will not describe the full operation of the appliance here. Instead, we shall describe how we update our appliance at Suffield.

The Adonis appliance runs BIND 9 on Linux. Thus, all the configuration data for DNS are accessible, just as they would be under a hand-configured nameserver. All critical DNS data live in the following directory:

```
/jail/named/
```

Backups of that directory can be taken directly using SSH/SCP.

8.2.2 Updating the Zones

To update zone data on the Adonis, follow these steps:

1. Launch the **Adonis Management Console** on your local computer. The console is written in Java and available for several platforms.
2. After the program launches, you will be shown a welcome screen asking you to open a project. Choose **Check Project out from server**.
3. Enter your **username** (this is used to keep a log of who has checked out the configuration), **server** address, and **password**. Click the **Check-out** button.
4. The **Server Explorer** will open, showing the server configuration.
5. Make any desired changes to the server and zones.
6. **Deploy** the configuration onto the server and verify that your changes have taken effect.
7. In the **File** menu, select **Checkin**.
8. In the window that appears, type a comment about the changes that you made, and click the **Check-in** button to stored the saved configuration file back on the server.

Note that the check-in/check-out procedure does **not** publish any changes on the server. You must still **deploy** the new configuration to the server in order for the changes to take effect. The check-in/check-out procedure simply stores the configuration file on the server so multiple people can access it.

8.3 Slave Server Configuration

Once you have a master server that is up and running, you can create slave servers that will replicate the DNS zones from the master and make them available to clients.

Note: this section describes the theory on setting up a slave server. All of this functionality is included in our standard base config, so if you just want to set up a slave as quickly as possible, please skip to the section on our standard configuration base.

8.3.1 Master Configuration

The master server must be configured to allow the slave servers to query for entire zones of data. Additionally, the master will send notifications to configured slaves whenever the zone data changes, allowing them to update more quickly.

You must add an **NS** record to the domain for every slave you wish to notify. If you do not wish to add **NS** records (for example, if you do not wish to publish the IP of one of your nameservers), you must add the IP of the slave to the **also-notify** parameter on the master.

DNSSEC Keys

For extra security, you may wish to enable DNSSEC keys so that you can digitally sign your zone transfers. This prevents malicious users from stealing all your zone data, and also prevents certain types of poisoning attacks against your server.

On the Adonis, you may set up new DNSSEC keys by clicking on the **DNS Service** configuration item and clicking on the **Security** tab in the configuration window.

Once you've set the key, you may set the **allow-transfer** parameter to use this key (instead of an IP address or ACL).

If you're configuring your master server by hand, you can generate a new key on the command line by executing the following:

```
dnssec-keygen -a HMAC-MD5 -b 128 -n HOST keyname.suffieldacademy.org.
```

That will generate a keyfile which contains the base-64 encoded version of the key, which can be added directly to the config file, like this:

```
key keyname.suffieldacademy.org. {
    algorithm hmac-md5;
    secret "<base-64 key goes here>";
};
```

Alternately, you can include the keyfile directly, so you don't have to keep the secret directly in the configuration:

```
include "/etc/bind/kename-file.key"
```

8.3.2 Slave Configuration

A slave configuration looks very similar to a master configuration. Most of the options, views, and other settings are exactly the same. The main difference for slave servers is that they have a type of **slave** with regards to their zone data, and they must be told which servers to contact to get authoritative zone information.

A slave server must be configured to query the master server(s) for the domains it slaves. This means adding stanzas like this to the config file:

```
zone "suffieldacademy.org" {
    type slave;
    masters { 192.168.0.1; };
    file "slave_suffieldacademy.org";
};
```

This tells the slave to be authoritative for the `suffieldacademy.org` domain, but get all of its information from the server at IP `192.168.0.1`. All the zone data are stored in the file `slave_suffieldacademy.org` in the name server's default directory.

We don't technically have to store the zone data in a file, but it has one major advantage: it serves as a backup of the zone data if the master should permanently lose its data. The slave zone files can be quickly "promoted" to master files, so having the zone data on hand can help in a disaster recovery scenario.

DNSSEC Keys

If the master server requires DNSSEC signatures on zone transfer requests (which re recommend), you must configure your slave server by giving it a key to use and instructing it to use the key for specific servers.

Start by adding a stanza that defines the key you wish to use:

```
key keyname.suffieldacademy.org. {
    algorithm hmac-md5;
    secret "<base-64 key goes here>";
};
```

Alternately, you can include the keyfile directly, so you don't have to keep the secret directly in the configuration:

```
include "/etc/bind/kename-file.key"
```

Next, you must specify which keys go with which server:

```
server 192.168.0.1 {
    keys { keyname.suffieldacademy.org.; };
};
```

This tells the slave to sign all requests to `192.168.0.1` with the given key.

8.4 DNS with Mac OS X Server

As of this writing, Mac OS X Server 10.4 includes the ISC BIND 9 nameserver by default. The Server Admin GUI allows you to configure and run a simple nameserver in either master or slave mode.

While the GUI is adequate for basic nameserver configuration, it does not allow for advanced configuration of BIND's options. Interface options, ACLs, dynamic uptates, TSIG security, and other features are not accessible from Apple's GUI.

For this reason, we choose to configure BIND "by hand" on our servers running Mac OS X Server. The GUI may still be used to start and stop the DNS service, but should not be used to perform any configuration of the server itself.

Additionally, there is an issue with IPv6, BIND, and certain root server queries. As of this writing (Mac OS X 10.4.6, BIND 9.2.2), caching nameservers run on Mac OS X will have extremely slow queries to the root name servers (4 seconds or more). This causes slow lookups for domains on the first attempt, though later attempts are cached and therefore not affected.

BIND 9.3 introduced a flag (`-4`) that temporarily disabled IPv6 which works around this problem. While not an ideal solution (the proper solution would be to fix IPv6 support), it does prevent the bug from happening. Unfortunately, the version of BIND shipped with Mac OS X (9.2.2) lacks this flag. We therefore compile our own version of BIND 9.3 and use it instead.

A version of BIND 9.3 can be found in the DarwinPorts collection, and can be installed without interfering with the native version of BIND shipped with Mac OS X.

8.4.1 Custom Changes

Our DNS configuration for Mac OS X Server builds on the stock configuration created by Apple. Because of this, the configuration file cannot be edited using the **Server Admin** GUI (it will overwrite the changes).

For more information on the changes we make to the config files, please see the section about our standard configuration base.

We currently run our Mac OS X machines as slaves to our master server. Therefore, we use the base config and only include information that pertains to slaves.

8.4.2 Configuration Files

Our standardized config lives in an entirely different directory from the BIND configs that ship with Mac OS X. Therefore, you don't need to worry about them interfering with each other.

8.5 Suffield DNS Config Template

To make setting up nameservers easier, we have broken our standard DNS config into several include files, along with a nameserver template file. Taken together, these files can be quickly pieced together to form a functioning master or slave nameserver.

The files are stored in revision control, and can also be viewed directly from the web:

Suffield DNS Config Template

For servers, we recommend checking the files out of version control. This way, changes to the files can be easily synched up using our version control software.

8.5.1 Differences

Our configuration is broken up into several different files, which makes it easy to plug in different functionality for different servers while maintaining a consistent

set of global options. For example, converting a server from master to slave would require only two lines to be edited.

In the file segments themselves, these are the major pieces of functionality we implement:

- A "main" template file, which is commented to show which sections must be edited to create a functioning configuration. Common configuration options are listed here.
- An ACL file with common Access Control Lists for address blocks.
- Two "views" (internal and external) based on the ACLs which define what answers the server will provide. We have a split-horizon DNS setup: answers are different depending on if a client is inside the firewall or outside.
- DNSSEC options, including cryptographic keys, signed updates, and slave delegation based on keys rather than source IP. Note that for security reasons, DNSSEC keys are **not** stored in source control. The file containing the keys must be crafted by hand on each server.
- Zone declarations, broken up by master/slave status and internal/external view status. Makeing thse things include files makes it very simple to update the zones on several machines (just pull a new version of the include file from source control and reload the server).

8.5.2 Using the Configuration

To use our base configuration, follow these steps:

1. Make sure DNS is not currently running on your server. Use the **Server Admin** GUI to stop the service if it is running.
2. Check out the stock config from our Subversion repository:

   ```
   cd /etc/

   sudo svn checkout \
   svn://svn.suffieldacademy.org/netadmin/trunk/software/dns/named.suffieldconf.d \
   named.suffieldconf.d
   ```

   The command above checks out the stock directory to the name `named.suffieldconf.d`. You may choose a different name for the directory, but you will need to change all of the `include` statements in the config file to reflect the new path.
3. If you're using DNSSEC (which we do), you'll need to create an include file with all the DNSSEC keys in it. First, copy our template config:

```
cp /etc/named.suffieldconf.d/key.inc.template /etc/named.suffieldconf.d/key.inc
```

Then, edit the file and add any keys that are needed for inter-server communication.

4. We need to create a few more directories where BIND will store its zone files:

```
mkdir /var/named/internal
mkdir /var/named/external
```

5. You should create a file in the `host_configs` directory of the `named.suffieldconf.d` directory. We recommend making the file on another machine, checking it into source control, and then pulling it down onto the server using `svn up`.

   Once the file is on the machine, create a symbolic from it to the standard `named.conf` location:

```
ln -s /etc/named.suffieldconf.d/host_configs/host_named.conf /etc/named.conf
```

6. Finally, copy the LaunchDaemon plist file for DNS into the server's `/Library/LaunchDaemons/` directory. You can download the file from our DNS LaunchDaemon repository.

At this point, you are ready to start the nameserver:

```
sudo launchctl load -w /Library/LaunchDaemons/org.isc.named.plist
```

Check the `/Library/Logs/named.log` file to see if the server starts correctly and loads its zone data. You can confirm that the service is running by typing:

```
sudo launchctl list
```

The `org.isc.named` identifier should appear if everything is running properly.

Important Note: the **Server Admin** GUI will show the status of the DNS server when it is running, but you **must not** use it to start and stop the service. Our LaunchDaemon file contains the pointer to our customized config files, whereas **Server Admin** will use the default one shipped by Apple. We use our own file because we observed **Server Admin** crashing when we overwrote Apple's config files with our own.

8.6 Disaster Recovery

Because we have multiple DNS servers running at all times, a short outage on one machine should not pose a serious problem for our DNS service. Even the loss of our master system would not cause a major disruption in DNS; the biggest problem would be that new records could not be added until the master was restored.

In the event that the master machine were offline for a long period of time (more than 4 days), the best way to recover would be to "promote" one of the slaves to become a master. Currently, our slaves are configured to continue serving zone data until they have been out of contact with the master server for more than 1 week. If the outage can be corrected before that time, no promotion should be necessary.

Because our slaves hold all the zone data that the master does, promotion is relatively straightforward:

1. Change all the "zone" entries to be of type **master** instead of type **slave**. Remove any references to master servers.
2. To avoid versioning problems, you should also rename the zone files, both on disk and in the config file. By convention, we name our master zone files with the word `zone_` in front of them (*e.g.*, `zone_suffieldacademy.org`). Slaves name their zone files with the word `slave_` in front (*e.g.*, `slave_suffieldacademy.org`). If you promote a machine to be a master, you should rename the zone files as well.
3. Make sure a promoted slave has all the keys that the defunct master had. We use TSIG to sign all transfer requests. The new master must have all the keys that the slaves will be using in order to complete the transfers.

   If you do not have access to a backup of the master's config file, you can reconstruct the keys by looking at the configurations of the slaves.

Be sure to save a copy of the old slave configuration so that you can "demote" it back when you're done.

Chapter 9

DHCP

Last updated 2008/03/18

9.1 Introduction

The **D**ynamic **H**ost **C**onfiguration **P**rotocol (**DHCP**) is a mechanism for automatically configuring networked computers from a centralized server. Usually, DHCP is used to provide IP addresses to clients without their needing to know anything about the network they are connected to. With more configuration, DHCP can also be set up to provide a rich set of information to clients, from routers to DNS servers all the way up to LDAP providers and diskless bootstrap information.

At Suffield, we use DHCP to automatically assign IP addresses to computers on our network. Because most of our users own laptops, we have the additional challenge of users changing locations (and thus, subnets) frequently. DHCP is used to not only hand out the proper IP address for each client, but also provide subnet- and LAN-specific information to each client so that they can connect to the network without requiring intervention by the user.

9.2 Mac OS X Server Setup

We run DHCP service on machines running Mac OS X Server 10.4. Mac OS X Server comes with its own built-in DHCP server. This server can be configured via the **Server Admin** GUI, and is also integrated into Apple's NetBoot services.

Unfortunately, Apple's DHCP server does not yet implement all of the features we need in a DHCP server, including dynamic DNS updates, failover, and known/unknown client grouping.

For this reason, we choose to install the ISC DHCP server on our Mac OS X Server machines, and configure it as we would on any other standard UNIX machine. (Note that this config would work equally well on a UNIX box, as the version of the ISC server is the same for all platforms.)

9.2.1 Getting dhcpd from DarwinPorts

DarwinPorts is a collection of ready-to-compile software for Mac OS X. If you have not yet installed DarwinPorts, please see our section describing the installation of DarwinPorts. (The section discusses the installation of Subversion, but the DarwinPorts information is not specific to that package.)

Once you have DarwinPorts installed, you simply need to install the `dhcp` port:

```
sudo port install dhcp
```

DarwinPorts will download, unpack, configure, compile, and clean the distribution for you automatically. When done, you should be able to run the following:

```
/opt/local/sbin/dhcpd --help
```

You should get the standard help screen for DHCP. If you do, then the binaries are installed, and you're ready to move on.

Note: as part of the installation process, DarwinPorts creates a new StartupItem called **DarwinPortsStartup**. This item then bootstraps any installed daemons under the DarwinPorts system. In its default state, no services are launched, so the item won't do any harm. We use Mac OS X 10.4 (Tiger), so we will configure **launchd** to start the DHCP server instead.

9.3 Suffield DNS Config Template

To make setting up DHCP servers easier, we have broken our standard DHCP config into several include files, along with a "primary" and "secondary" configuration template file. Taken together, these files can be quickly pieced together to form a functioning primary or secondary DHCP server.

The files are stored in revision control, and can also be viewed directly from the web:

Suffield DHCP Config Template

For servers, we recommend checking the files out of version control. This way, changes to the files can be easily synched up using our version control software.

9.3.1 Differences

Our configuration is broken up into several different files, which makes it easy to plug in different functionality for different servers while maintaining a consistent set of global options. The primary and secondary configs simply include these core options, and only add the small changes required for the symmetry in a failover pair.

In the file segments themselves, these are the major pieces of functionality we implement:

- A *global options* file, which contains all of the server-wide option settings (nameservers, domain names, default lease times, *etc.*).
- A *ddns options* file, which covers all settings related to Dynamic DNS updates (DDNS).
- A *zones* file, which contains all of the forward and reverse DNS zone declarations used by DDNS.
- A *keys* file, to hold DNSSEC keys to digitally sign DNS updates.
- A series of *clients* files, which contain the static host declarations for known clients.
- A *common* file, which ties together all of these options into a single master file.
- *Primary* and *secondary* files, which add the necessary statements for failover operation. These are the files you should use as your main `dhcpd.conf` file on a server.

9.3.2 Using the Configuration

To use our base configuration, follow these steps:

1. Check out the stock config from our Subversion repository:

```
cd /etc/

sudo svn checkout \
svn://svn.suffieldacademy.org/netadmin/trunk/software/dhcp/dhcp.d \
dhcp.d
```

The command above checks out the stock directory to the name `dhcp.d`. You may choose a different name for the directory, but you will need to change all of the `include` statements in the config file to reflect the new path.

2. If you're using DNSSEC (which we do), you'll need to create an include file with all the DNSSEC keys in it. First, copy our template config:

   ```
   cp /etc/dhcp.d/key.inc.template /etc/dhcp.d/key.inc
   ```

   Then, edit the file and add any keys that are needed for inter-server communication.

3. Our configuration is designed for failover operation, so there are two files to choose from: `primary.conf` and `secondary.conf`. Link the appropriate file to `/etc/dhcpd.conf`:

   ```
   sudo ln -s /etc/dhcp.d/primary.conf /etc/dhcpd.conf
   ```

   or

   ```
   sudo ln -s /etc/dhcp.d/secondary.conf /etc/dhcpd.conf
   ```

4. DHCP will not start until you've created an empty lease file for it. This is to prevent errors if you accidentally move the lease file out of the way for manual recovery:

   ```
   sudo touch /var/db/dhcpd.leases
   ```

5. Our DHCP config is set to log via the `syslog` logging facility under UNIX. However, many systems are not configured to "listen" for these log announcements. To fix this, edit `/etc/syslog.conf` and add the following line:

   ```
   local2.info /var/log/dhcpd.log
   ```

   (You may use a different log level facility if you wish; `info` prints all lease transactions and may be too verbose for some users.)

   You will need to kill and restart the `syslogd` daemon for these changes to take effect.

1. Finally, copy the LaunchDaemon plist file for DHCP into the server's `/Library/LaunchDaemons/` directory. You can download the file from our DHCP LaunchDaemon repository.

At this point, you are ready to start the DHCP server:

```
sudo launchctl load -w /Library/LaunchDaemons/org.isc.dhcpd.plist
```

Check the system logs to see if the server reports any errors. You may confirm that the service is running by typing:

```
sudo launchctl list
```

The `org.isc.dhcpd` identifier should appear if everything is running properly. Check the `/var/log/dhcpd.log` file to confirm that the server has started (it should print informational messages on startup).

9.4 Using DHCP

Once DHCP is installed and configured, you are ready to run the daemon in a production environment. Normally, you won't need to interfere with the daemon's operation; it should serve requests normally without need for restart or reconfiguration.

From time to time, you will need to reconfigure the server. Normally, this is done to add hosts to the database that have specific names or IP addresses.

9.4.1 Normal Operation Indicators

When the server is working properly, you will see entries in the log like this:

```
Apr 19 08:17:01 dhcpd: DHCPDISCOVER from de:ad:be:ef:00:0d via eth0
Apr 19 08:17:02 dhcpd: DHCPOFFER on 192.168.0.1 to de:ad:be:ef:00:0d via eth0
Apr 19 08:17:03 dhcpd: DHCPREQUEST for 192.168.0.1 from de:ad:be:ef:00:0d (jersey) via eth0
Apr 19 08:17:04 dhcpd: DHCPACK on 192.168.0.1 to de:ad:be:ef:00:0d (jersey) via eth0
```

In this case, a client has broadcast that it wants an address (DISCOVER), the server has made an offer (OFFER), the client confirms the offer by requesting the specific address (REQUEST), and the server confirms that the address is available for use (ACK).

Later, when clients already have an address, they will renew the address by performing the final two steps again (REQUEST/ACK).

You may occasionally see deny messages (DHCPNACK). These are not cause for concern; normally they are sent to clients that have moved between subnets and no longer have a valid address for the subnet they are on. The server sends a negative ACK, and the client begins the negotiation process anew to get a proper address.

9.4.2 Finding a Client's Address

If you ever need to find the MAC address that goes with an IP address (or vice-versa), just grep the log file for the IP or MAC address. The last matching entry will be the assignment made by the server.

9.4.3 Starting the Daemon

If you're using Mac OS X Server 10.4, the DHCP server should start itself via `launchd` if you've installed our LaunchDaemon item. If you've manually stopped the service, you can start it again using:

```
sudo launchctl load -w /Library/LaunchDaemons/org.isc.dhcpd.plist
```

The `-w` flag makes the load status permanent; that is, the server will continue to start the process from this point forward, even after a reboot of the machine.

If you use a regular UNIX system, launching `dhcpd` is usually accomplished using an `/etc/init.d/` or `/etc/rc.d/` script. Use the method described by your UNIX distribution.

9.4.4 Stoping the Daemon

If you're using Mac OS X Server 10.4, you must use `launchd` to stop the daemon process (if you don't, `launchd` will just start it back up for you). To do this, type the following:

```
sudo launchctl unload -w /Library/LaunchDaemons/org.isc.dhcpd.plist
```

The `-w` flag makes the unload status permanent; that is, the server will never start up again until you re-enable it with `launchctl` (even if you reboot).

If you use a regular UNIX system, stopping `dhcpd` is usually accomplished using an `/etc/init.d/` or `/etc/rc.d/` script. Use the method described by your UNIX distribution. If you're in a pinch, you can also try killing the process directly. Assuming it isn't being respawned by `init`, the DHCP server should exit and stop running.

9.4.5 Loading Configuration Changes

The ISC DHCP server does not have a "reload" option. Therefore, the only way to re-read the configuration files is to stop the server and start it back up again.

When you've made changes to the configuration and are ready for them to take effect, you should always test the configuration first:

```
sudo /opt/local/sbin/dhcpd -t
```

Confirm that there are no errors reported before you restart the server.

If you're using `launchd` to run DHCP (as we describe above), reloading the configuration is as simple as manually killing the process:

```
sudo killall dhcpd
```

`launchd` will notice that the process has died, and will automatically restart it. Upon restart, the new configuration will be read, and your changes should take effect.

On systems that use a traditional start/stop mechanism, you can simply stop the server and start it immediately after, using the start and stop procedures described above.

9.4.6 Adding Fixed Hosts

While the DHCP server usually tries to give the same address to clients every time, there is no guarantee. Some hosts on the network should be given "fixed" addresses that never change (such as network equipment, appliances, and other vital equipment).

Note: "regular" clients can be given a name and parameters that don't change, without needing to assign a specific IP address. If you're adding a record for a user's computer, please see the next section for information on how to add a host record with a dynamic address.

To ensure that a client gets a fixed address every time, follow these steps:

1. Create forward and reverse DNS entries for the client on your DNS server.
2. Edit the section of your configuration files where host declarations are kept. In our config, the host info is kept in the `client_*.inc` files. You should add a stanza that looks like this:

   ```
   host my-new-hostname {
       hardware ethernet fa:ce:ca:fe:ba:be;
       fixed-address new-hostname.example.org;
   }
   ```

This tells the DHCP server to always associate the IP address that `new-hostname.example.org` resolves to with the given Ethernet MAC address.

3. Restart the server to reload the configuration.

9.4.7 Adding Registered Hosts

Under our configuration, the DHCP server knows about three types of clients: "fixed" clients that are given static IP addresses, "registered" clients that have some identifying information known about them, and "rogue" clients that are unknown to the server.

By default, all clients get addresses from the server, so it is not necessary to register a client before connecting it to the network. However, "registered" clients receive better treatment, in the form of (possibly) longer leases, dynamic DNS updates, and a larger pool of addresses.

Registered hosts should be used for any hosts that should receive a consistent name in DNS, but that might travel between subnets (and thus need a different IP at different times). It's somewhat of a comprimise between "rogue" hosts and "fixed" hosts.

To add a registered host, follow these steps:

1. Edit the section of your configuration files where host declarations are kept. In our config, the host info is kept in the `client_*.inc` files. You should add a stanza that looks like this:

   ```
   host my-new-hostname {
       hardware ethernet de:ad:be:ef:00:00;
       ddns-hostname "my-new-hostname";
   }
   ```

   This tells the DHCP server to always associate the given DNS hostname with the given Ethernet MAC address. The Dynamic DNS (DDNS) features of the DHCP server will ensure that the client always gets the same DNS name, even if the IP address it resolves to changes.

2. Restart the server to reload the configuration.

9.4.8 Failover

Our default configuration employs a peered failover system to ensure reliable service. The ISC DHCP server supports the DHCP failover protocol with only minimal extra configuration.

This section gives a brief guide for using failover and recovering from disasters. For more detailed information, please see the manual page for `dhcpd.conf` and `dhcpd`.

For the most part, failover requires no intervention on the part of the administrator. When one server fails, the other notices (you'll see a log message), and will attempt to service clients that were handled by the peer server.

For short outages (server reboots, configuration changes, etc), you don't need to worry about failover. When the other machine comes back, it will automatically syncronize with the running server.

For long outages (days, weeks, months), you may wish to manually place the running server into **partner-down** mode. This tells the running server that its peer is known to be down, so it can take over full service for the network.

Getting In to the Partner-Down State

Note: if you place a server into **partner-down** mode, you must be **absolutely sure** that the peer is legitimately down, and not just out of contact with the server. If the failed server is still running, and the running server begins to take over its addresses, then both servers might hand out the same addresses to different clients.

If a peer in a failover setup fails, and you wish to have the surviving server take over all service for the network, follow these steps:

1. Stop the functioning server, so that it no longer is servicing clients (make sure the server does not automatically restart if you are using `launchd`).

2. Edit the `/var/db/dhcpd.leases` file, and search for the **last** peer declaration stanza. Remove the date from the "`my`" portion of the state, so it is blank. Change the "`my`" portion of the state to have a status of `partner-down`. Leave the "`peer`" portion of the state alone. For example:

   ```
   failover peer "suffield" state {
       my state partner-down;
       peer state communications-interrupted at 2 2005/08/23 13:14:15;
   }
   ```

3. Start the server back up again, and verify that the state is now **partner-down** in the log files.

Over time, the running server will reclaim leases that used to be handled by the failed server.

Getting Out of the Partner-Down State

Note: you **must not** bring a failed peer back up unless the running peer is still functioning and ready to communicate with the failed peer. If you do otherwise, the restored peer will assume that nothing was wrong, and begin serving addresses normally. The restored peer must be able to communicate with the running peer right away so that it can discover that the other peer is operating in the **partner-down** state, and synchronize its database correctly.

After a prolonged outage where the running server has been placed in the **partner-down** state, when you are ready to bring a failed peer back online you simply need to confirm that the two servers can communicate with each other.

Start the failed peer up normally. It should restore communication with the running peer, discover that the running peer took over service, and synchronize its database with the running peer. Once the two machines have synchronized, the restored machine will wait for **MCLT** before serving clients. This gives any clients that received a lease from the failed peer time to expire, preventing conflicts.

At this point, both servers should return to their normal state, and begin serving clients in failover mode again.

Recovering from a Lost Lease Database

If a peer fails catastrophically and loses its lease database, you do not need to take any special action to recover. When the failed peer is repaired and able to communicate with the running peer, you may start it normally.

The failed peer will note that it has no state information, and assume that it needs to synchronize its state. It will download a copy of the lease database from the running peer, wait **MCLT** to prevent address conflicts, and then begin serving requests again normally.

Chapter 10

Printing

Last updated 2008/03/18

10.1 Introduction

Printing is a service that is both popular and complex. Users want to immediately connect to printers to produce their work (especially students with papers that are overdue), but don't want to understand the complexity of drivers, PPD files, installed options, and spooling.

Over the past few years at Suffield, we've gone through several methods of printing. Very briefly:

- When we had an AppleTalk network, all printers were advertised via AppleTalk. Users with Macintosh computers simply opened the Chooser and selected the printer (by name) that they wished to use. Driver selection was automatic, or the default "generic" driver sufficed.
- As printers began to move to IP, centralized management of printers became easier, but users could not easily discover printers. We tried several print servers based on CUPS (the Common UNIX Print System) and IPP (Internet Printing Protocol). The protocol did advertise printers, but clients had to be configured to "listen" to the announcements.
- Centralized queue management was put in place to prevent printing abuses. However, the queues frequently stopped due to "errors" at the printer (such as paper jams and low toner), which required manual restarts of the queue. Users became frustrated because they could fix the underlying issue (the paper jam), but not restart the queue themselves.

- We reverted to a pure "every man for himself" system, where clients printed directly to printers. Discovery was not automatic, though, so we ended up creating a small installer program for users that would properly configure their machine with all of the printers on campus.

This leads to our current situation. Starting with Mac OS X 10.4 (Tiger), it's now possible to advertise services via a managed DNS domain. This uses the Bonjour (formerly Rendezvous) technology, but through a centralized system rather than peer-to-peer broadcasts. Clients can now auto-discover available printers and print directly to them.

This document describes the various pieces of the network puzzle that make up our printing services. It also describes a helper script that we wrote to generate the various configuration files necessary to keep printing running.

10.2 Printing Overview

Below we provide a brief overview of how all the parts of our printing system work together. The various services are discussed in more detail in the following sections.

10.2.1 Printer Hardware

When a printer is connected to the network and powered up, it asks for an IP address via **DHCP**. The DHCP server provides the printer with basic information (it's name, address, DNS servers, *etc.*). The DHCP server also provides the name of a **TFTP** configuration file. The printer fetches this file from our TFTP server, and uses it to set more advanced configuration options (available protocols, ACLs, SNMP settings, *etc.*).

Once configuration is complete, the printer goes online and is ready to accept jobs directly from users.

10.2.2 Client Machines

When a client machine connects to our network, it should notice our Service Discovery Browse Domain (via **Bonjour**/Rendezvous/Zeroconf). This is a special DNS domain that lists services (web, printing, *etc.*) on our network and the hosts that provide them.

When a user wishes to add a printer to their system, they simply use the built-in tools under Mac OS X to browse from the list of advertised printers. The make

and model of printer is published, making driver selection automatic for clients that have the proper PPD files.

Once the printer is installed, the client may print directly to the printer without further configuration. Additionally, the configured printer remains for future use.

10.2.3 Older Clients

Clients that don't support domain-wide Service Discovery (Mac OS X 10.3 and earlier) can fall back to running an installer provided by us. This installer uses CUPS command-line programs to manually add queues for all the known printers on campus. This method is less flexible than auto-discovery, but it is relatively quick (less than 1 minute), and easy (no user interaction required).

Once the queues are installed, the user simply selects the one they wish to use, and they can print directly to the printer. Installation does not need to happen again unless the network addresses of the printers change.

10.3 DNS Service Discovery (Bonjour)

To automatically advertise printers to clients, we use DNS-based service discovery (DNS-SD). Apple refers to this technology under their trade name **Bonjour** (formerly Rendezvous). While DNS-SD can be used for more than just printing, we'll focus on its printing uses in this document.

For more information on DNS-SD, please visit their web site:

http://www.dns-sd.org/

We use what they describe as "Wide-area DNS-SD", which is a fixed form of service discovery published via traditional DNS. Another form does exist, where clients may multicast the services they provide, but we're interested in a centrally-controlled publication of services, since we only wish to advertise services provided by the school.

In a nutshell, two things require setup in order to use DNS-SD:

1. A DNS zone for the publication of services must be created and advertised

2. Services offered by hosts must be added to this zone

10.3.1 Publishing a Service Zone

First, we create a zone to publish our service announcements in. This does not have to be a separate zone; you can use your top-level domain if you wish. However, we publish in a separate zone because our primary DNS server can't properly edit DNS-SD records (plain vanilla BIND9 works fine, so we use that server to host our DNS-SD domain).

We have called our domain `zeroconf.suffieldacademy.org`, and we host it on a BIND9 server. In order to advertise this as our SD domain, we need a few glue records in our primary domain:

```
b._dns-sd._udp.suffieldacademy.org. IN PTR zeroconf.suffieldacademy.org.
lb._dns-sd._udp.suffieldacademy.org. IN PTR zeroconf.suffieldacademy.org.
```

These records advertise the "browse" and "legacy" browse domains in our default domain, and refer clients to the "zeroconf" domain to actually perform discovery. If you wanted, you could refer to any other domain (including the top-level domain you're already using).

Once these changes are published, you have created and advertised the domain, and you're ready to publish service records.

10.3.2 Publishing Service Records

Once you have the SD domain created, you need to create several records for each service you'd like to offer. In general, three records are required to advertise a single service. We'll discuss each one and give an example below.

Let's suppose we want to create a service offering for a printer. The name of the service should be "Faculty Room Printer" (this is a "friendly" name shown to users). When users select this service, they should print to a printer called "hplj4100-facroom.suffieldacademy.org" using IPP as the printing protocol.

The Advertisement Record

The first record links the service (IPP) to a particular offering ("Faculty Room Printer"). Clients can then ask for a service (IPP) and get an enumeration of all the hosts offering that service.

A sample record would look like this (BIND9 configuration file syntax):

```
_ipp._tcp IN PTR Faculty\032Room\032Printer._ipp._tcp
```

Note the use of escapes to convert non-alphanumeric characters into their decimal equivalent (32 is the space char in ASCII). You'll need to do this for all characters that are not valid in a DNS hostname.

This record literally says, "anyone looking for TCP/IPP service should check out the Faculty Room Printer TCP/IPP service offering". DNS automatically allows for multiple pointers, so if there are multiple offerings each one gets its own record.

The Service Record

The second record links the offering ("Faculty Room Printer") with a particular DNS hostname ("hplj4100-facroom.suffieldacademy.org"), using a standard DNS SRV record. This allows you to create "friendly" names for offerings and link them to actual DNS hostnames behind the scenes.

Here is a sample record, broken up onto two lines:

```
Faculty\032Room\032Printer._ipp._tcp IN SRV 0 0 631
hplj4100-facroom.suffieldacademy.org.
```

This time, we're associating a friendly name on the left with a specific DNS hostname and port number on the right.

The Options Record

The last record associates a DNS TXT record with the offering ("Faculty Room Printer"). This record is used to store additional information about the offering (*e.g.*, printer make and model, queue name, and options). In this way, clients can auto-configure themselves to use the offering.

The configuration is stored as a series of key=value pairs, lumped together in a single TXT record. Here is a sample record in BIND9 format:

```
Faculty\032Room\032Printer._ipp._tcp IN TXT (
"txtvers=1"
"Color=F"
"Duplex=T"
"note=Faculty Room, next to copier"
"product=(HP LaserJet 4100 Series )"
"ty=HP LaserJet 4100n"
)
```

The specific key/value pairs are described in more detail on the DNS-SD site. The two most important (for the end user) are the "note" key, which provides

a description to the user, and the "product" key, which helps the client find a suitable PPD to autoconfigure the printer.

10.3.3 Final Notes

Those three records must exist for every service offering. Note that if a printer supports multiple protocols (*e.g.*, IPP and LPD), you'll need a set of three records *for each protocol*. In order words, you'd need three records under the `_ipp._tcp` domain (for IPP), and three more under the `_printer._tcp` domain (for LPD).

10.4 CUPS Configuration

For computers that don't yet support DNS service discovery, we fall back to providing an installer that creates the queues for them. All Mac OS X machines (since 10.2) use CUPS as their underlying print client, and we can configure queues for CUPS directly from the command line.

We create a simple shell script that adds print queues using the `lpadmin` command:

```
lpadmin -p "hplj4100-facroom.suffieldacademy.org" -E \
    -v "ipp://hplj4100-facroom.suffieldacademy.org/ipp/" \
    -L "Faculty Room, next to copier" \
    -D "Faculty Room Printer"
```

Note that it specifies the host, URL (IPP in this case), location, and "friendly" name. We simply run this for every printer available on campus.

We place this shell script inside an empty Mac OS X installer package. The package doesn't install any files, but instead runs the script as a "postinstall" and "postupgrade" step. This allows the user to have a simple familiar interface (package installation) which runs a shell script.

10.5 DHCP Configuration

To make our life easier, we use DHCP (and TFTP; see below) to auto-configure printers. This doesn't have anything to do with auto-discovery of printers by clients; it's just to make printer management easier from our end.

This is not a document on DHCP; for that, we recommend you check out Our documentation on running DHCP. Briefly, you'll need to do a few things:

1. First, if you're going to use TFTP (see next section) with HP printers (what we use), you'll need to enable a few options to make this possible. In the global section of your DHCP config, add options 144 and 150 to your server:

   ```
   option jd-tftp-config code 144 = string;
   option jd-tftp-server code 150 = string;
   ```

   As you can see, 144 defines the TFTP configuration file path, and 150 defines the TFTP server to use.

2. Next, define the TFTP server (if you don't, the printer will default to using the DHCP server):

   ```
   option jd-tftp-server "tftp.suffieldacademy.org";
   ```

3. Finally, create a host stanza for every printer, and assign the printer a name and TFTP config file:

   ```
   host hplj4100-facroom.suffieldacademy.org {
       hardware ethernet de:ad:be:ef:f0:0d;
       ddns-hostname "hplj4100-facroom.suffieldacademy.org";
       option jd-tftp-config "/private/tftpboot/printers/deadbeeff00d";
   }
   ```

   Note that we store the configuration by MAC address. This parameter must be kept relatively short, as some printers won't accept a filename that is too long.

Once you've finished setting up DHCP and TFTP, you can power up the printers and they should be automatically configured.

10.6 TFTP Configuration of Printers

The printers we use (HP LaserJets) have the ability to take configuration options via TFTP at boot time. By specifying the path to a TFTP file in the DHCP options for a printer (see previous section), the printer will get this path when it boots and try to read that configuration file.

This process doesn't have anything to do with making the client experience better; it's just to make managing printers a little easier. You can skip this process and still have auto-discovered printers using only DNS-SD.

The configuration file is in the same format as a telnet-based config; any command you can issue via telnet should also work when placed verbatim inside a

file. We suggest telnetting to your printer and using the "export" command to produce a file that you can then customize.

Once you have the configuration file the way you want it, simply make it available on a TFTP server. TFTP servers are available for all UNIX-based platforms, and Mac OS X comes with one built-in. Just enable it with:

```
sudo launchctl load -w /System/Library/LaunchDaemons/tftpd.plist
```

To make the configuration files available, just put them in `/private/tftpboot` (or wherever your TFTP server path is). You can even place the configuration files in their own directory to keep them all together.

Once that's done, just provide the path to the config in the DHCP host stanza. The next time the printer boots, it should look for the configuration file option, try to fetch the file via TFTP, and load its contents. Check your syslog (or configuration page) for errors in this process.

10.7 Our Helper Script

To assist with all of these different configuration files, we've created a unified helper script. The script takes a single description file as its input, and produces configuration file snippets for all the services mentioned previously (DNS, DHCP, TFTP, and CUPS).

The script is well-documented, including a sample configuration file. Please download the script from our website and use "perldoc" on it to read the built-in documentation:

```
perldoc exporter
```

To use the script, simply list the services you want to generate, followed by the input file and an output directory. For example, to generate all service files, you would do the following:

```
./exporter tftp dhcp dns cups printers.dat output_dir
```

The output directory will now contain a partial DNS zone file, DHCP host stanzas, a CUPS installer script, and a directory of TFTP configuration files.

The script has two dependencies: a directory of template files used for creating the TFTP configurations, and a directory of PPD files that can be used by the CUPS installer.

The TFTP template files are named by printer model, or by exact hostname. If you like, you can browse our template directory online.

The CUPS PPD files are simply the PPDs provided with the printer, or that we have used from machines that work. You should download or use PPDs that are known to work with your printers.

Chapter 11

Syslog (Centralized Logging and Analysis)

Last updated 2008/03/18

11.1 Introduction

In a perfect world, nothing would ever go wrong. Unfortunately (at least in the network world), nothing is ever perfect, and things do go wrong from time to time. One of the best ways to diagnose problems is through the use of **logging**. Most applications and systems generate log messages, either on a regular basis (status logging) or when something goes wrong (error logging). Collecting and analysing these logs is a great way to track down problems.

Unfortunately, keeping track of all the logs on every piece of equipment can be a daunting task. This document explains how to collect all application logs in a central location for easier analysis. We also discuss software to monitor and report on anomolies found in the log files.

11.1.1 Logging Software

At Suffield, we mainly use machines and appliances that adhere to the **syslog** log format. This format has been used in the UNIX world for quite some time, and it is readily supported by many operating systems, appliances, and software packages.

Syslogs are normally collected by a **syslog daemon**. The daemon we have

chosen to use is called **syslog-ng** (for "Syslog Next Generation"). It allows for complex setups with multiple sources and destinations of logging, along with built-in pattern-matching conditions and custom hooks for external analysis.

11.1.2 Analysis Software

While **syslog-ng** does possess some filtering and reporting capabilities, we prefer to run our logs through an external tool. This allows us to search the logs for unusual behavior (failed logins, warning messages, *etc.*) so we can take corrective action as soon as possible.

11.1.3 Intended Audience

This document assumes that the reader is proficient in the installation and configuration of a UNIX-like operating system. We will be using the Debian flavor of Linux for our examples, though most other unices should operate in a similar manner.

11.2 syslog-ng

We use a drop-in replacement for the standard `sylog` daemon called sylog-ng. It receives messages from remote hosts in the standard syslog format, but it provides many nice features that other daemons lack. Of particular interest is its ability to route messages to different files, external databases, or external programs. The system is very flexible, and works well even for large-scale installations.

For more information on `syslog-ng`, including detailed installation instructions, the latest downloads, and other information, please see the main syslog-ng web site:

http://www.balabit.com/products/syslog_ng/

Additionally, you may wish to read O'Reilly's *Building Secure Servers with Linux* book. Chapter 12 is available for free online, and deals specifically with syslog-ng (as well as some of the other topics discussed here). The book's website is:

http://www.oreilly.com/catalog/linuxss2/

11.2.1 General Concepts

Syslog-ng works with three major components in its configuration:

1. **sources** of log information (e.g., network sockets)
2. **destinations** of log information (e.g., files)
3. **filters** that match certain log information

Syslog-ng combines these three things into one or more log pipelines. This gives you the ability to pipe multiple sources into a certain destination, send one source to many destinations, or send only certain logs to particular destinations based on filters.

11.2.2 Installation

Many systems have pre-built packages of `syslog-ng` available, without the need for source. We use Debian linux, and the install is as simple as:

```
# apt-get install syslog-ng
```

This will create a sample configuration file in `/etc/syslog-ng`, which you can customize.

11.2.3 Configuration

All configuration occurs in the file `/etc/syslog-ng.conf`. The sections below discuss changes to the defaults provided under Debian. Note that some installations may use different defaults for other values; check the documentation for more information about special options.

Global Options

All options belong in a configuration stanza named `options{}` (the configuration values go between the { curly braces }).

We use this section to set global timeouts and retries for logs and network connections. In some cases, these values can be overridden in later sections of the files.

Additionally, this section deals with use of DNS to resolve hostnames. The important values here are:

- `chain_hostnames(no)`: tell the server not to report every hostname between the source and receiver; just use the source's address.
- `keep_hostname(no)`: do not trust the hostname used by the source; replace it with the name queried from DNS. Note that if you do not have proper reverse DNS set up, this can cause `syslog-ng` to block while it waits for DNS queries.
- `use_dns(yes)`: allow the server to make DNS queries for host names.
- `use_fqdn(yes)`: store the full name of the source, rather than just the short hostname.

Other values exist for tweaking the caching and timeouts of DNS lookups; see the configuration file for more information.

Sources

Syslog-ng must be told which sources to listen on to receive messages. The most frequently used ones are:

- `internal()`: This source is generated from within syslog-ng itself. Any status messages that syslog-ng creates will be sent from this source, so you should include it in at least one log statement to capture its output.
- `file()`: This reads from a special file on the local system. Under Linux, you'll want to use this to read `/proc/kmsg`. Note that this does **not** "tail" a regular file; it only works with special files designed to work as pipes.
- `unix_dgram()`: This reads from a local UNIX socket in a connectionless fashion (use `unix_stream()` for connection-oriented sockets). On Linux, this is used to read the special file `/dev/log`.
- `udp()`: This causes syslog-ng to listen to a UDP port on the network interface for messages from other systems. Since we're building a centralized logging host, you'll **definitely** want to use this option. A TCP option also exists, though it is less frequently used (most clients default to UDP).

You'll need to declare one or more sources, each using one or more of these sources. We recommend lumping the sources into broader source categories. For example, `/proc/kmsg`, `internal()`, and `/dev/log` are all local to the machine that is running syslog-ng. Therefore, you may wish to merge them all into a single source that tracks local machine messages.

To declare a source, simply use the `source` parameter, give it a name, and enclose all the sources in { curly braces }:

```
source s_local  { unix_dgram("/dev/log"); internal() };
source s_remote { udp(); };
```

Filters

Filters determine which messages will be routed to which destinations. Filters can test every aspect of an incoming message (often using regular expressions), making them extremely powerful. You can match source IP, subnet, syslog priority, log facility, or even strings embedded in the messages.

You can combine conditions with "and" and "or", as well as negate them with "not". This allows you to build up a complex filter out of many simple matching rules.

To declare a filter, use a `filter` parameter, give it a name, and enclose the matching rules inside { curly braces }:

```
filter f_simple  { match("foobar"); };
filter f_complex { facility(mail) and (level(info .. err) or host("bob")); };
```

Destinations

Once messages have been filtered, they must be sent to a destination. Syslog-ng can use many different types of destinations, including plain files, pipes, other syslog hosts, and external programs.

Files are the most frequently used destination in a typical setup, as they are permanently stored, easy to search, and work in the same way that a traditional syslog daemon does. We will primarily examine files in this section.

However, it is worth noting two other destinations, though we will not discuss them further:

- The `program()` call forks to a new process and sends messages to it via STDIN. This is an easy way to quickly get messages to an external program. However, `syslog-ng` only forks once per startup of the daemon, which means that the program is tied to the execution of `syslog-ng` itself.

- The `pipe()` call sends messages to a named pipe on the system. This can be useful when sending messages to a completely separate process (such as `xconsole` or a custom script that's waiting for input). It is more robust than `program()`, as the log files can be "dropped off" at the pipe and later read by a different program with different privileges or parameters.

Destinations may have permissions, owners, groups, sync times, and other options that override the ones defined in the global `options{}` section of the config file. You may wish to tweak some of these settings (for example, setting a lower sync rate on busy log files).

Destinations may also set a **template**, which defines how the logs are formatted as they are output. Each template contains a formatting string which can contain **macros** that are expanded to their values at the time of output. For example, to format each line with the full date, host, and level of severity:

```
template("$FULLDATE $HOST [$LEVEL]: $MESSAGE");
```

Additionally, for the `file()` destination, you can use template macros in the filename. This allows for simple writing to different files based upon content in the message itself (such as date, time, or host).

Here is a sample output to file using a templated name, and specially formatted output:

```
destination d_file { file("/var/log/$YEAR.$MONTH.$DAY/$HOST"
                          template("$FULLDATE $HOST [$LEVEL]: $MESSAGE")
                         );
                   };
```

Log Rules

Once you've defined sources, destinations, and filters, you're ready to combine them into **log rules**. A log rule simly ties together a source and destination, and optionally allows for a filter to only allow certain messages to match the rule. In its simplest form, a log rule looks like this:

```
log { source(s_local); filter(f_important); destination(d_varlog); };
```

(Assuming that `s_local`, `f_important`, and `d_varlog` have already been set up earlier in the file.)

You are free to provide multiple sources, filters, and destinations in a single log statement, allowing you to easily create complex log setups.

Finally, newer versions of syslog-ng support a **flags** option in log rules. This special option has three settings:

1. **final** means that once a message matches this rule, it is no longer considered for other log rules.

2. **fallback** means that this rule matches only if the message has not matched any non-fallback rules so far (useful as a "default" rule to catch unmatched messages).

3. **catchall** means that only the filter and destination are used for this rule; all possible message sources are used.

Thus, to make a rule that matches only unmatched messages, we might say:

```
log { source(s_local); destination(d_default); flags(fallback); };
```

You should create log statements that link up all the sources and destinations that you will be using. Note that unless you use flags like "fallback", it is possible for a log message to match more than one log rule, and thus be logged to more than one destination.

11.3 Syslog Client Configuration

To send information to the central server from a machine running `syslog`, we simply instruct the daemon to forward information to a remote host.

The simplest way to do this is to add a catch-all for every message in `/etc/syslog.conf`:

```
*.* @172.30.0.10
```

Then, restart your syslog daemon (using an init script, `kill -HUP`, or whatever method your platform uses).

11.4 Logcheck

Once you have a centralized logging machine in place, its time to do something with those logs. While it's helpful to keep logs files for later review, it's even more helpful to automatcially scan log files for "strange" events so they can be reported. What constitutes a "strange" event is up to you; you'll need to decide what sorts of events require further inspection.

At Suffield, we make use of the `logcheck` program to help us with automated analysis. The program scans log files at regular intervals, and e-mails any lines that look suspicious. In this way, we are automatically alerted when certain log messages appear.

11.4.1 Obtaining the Software

We use the Debian flavor of Linux, which includes a prepackaged version of `logcheck`. Simply install the tool from the command line with:

```
apt-get install logcheck
```

If you don't use Debian, your distribution may still include a package for `logcheck`. Consult your distribution's package database.

If you can't find it, you can download the upstream version directly:

http://sourceforge.net/projects/sentrytools

11.4.2 Usage

Normally, logcheck is run from `cron` (or some other scheduling process) on a regular basis (*e.g.*, once an hour). The script checks a list of log files for specific patters, and reports any that are found.

Logcheck includes three basic levels of reporting functionality: *workstation*, *server*, and *paranoid*. Each level includes everything in the levels below (*i.e.*, *server* includes all of *workstation*). You should choose a level that reports enough information, without flooding you with too many false reports. Our approach has been to select the highest level (*paranoid*), and then selectively disable warnings using custom patterns.

Note that logcheck defaults to reporting **all** unrecognized messages. In other words, if you do not have a pattern which explicitly matches a message type, logcheck will include it as suspicious. This is good, as it ensures that you receive messages that you haven't necessarily dreamed up a configuration for.

11.4.3 Syslog-NG and Logcheck

Logcheck requires a list of log files to check. Unfortunately, our `syslog-ng` setup files logs away in many different files (by date, category, program, *etc.*). To make running logcheck easier, we have added an additional destination file to syslog-ng. We funnel many different types of logs to this file, so that logcheck only needs to inspect a single file.

Because this log file is a copy of data logged elsewhere, we don't need to keep it around when we're done. Thus, we use syslog-ng's ability to name files by day to set up a weekly rotation where old files get overwritten by new ones. This allows for a short amount of history (one week) without wasting too much space.

A sample syslog-ng stanza might look like this:

```
destination df_logcheck {
  file("/var/log/logcheck/$WEEKDAY.log"
       template("$FULLDATE: $HOST ($FACILITY/$LEVEL) [$PROGRAM] $MSGONLY\n")
       template_escape(no)
       remove_if_older(518400) # overwrite if older than 6 days
  );
};
```

You can configure syslog-ng to send any (or even all) data to this file in addition to the regular files you would otherwise use.

11.4.4 Basic Configuration

At the minimum, you must provide a list of log files to check, the patterns you wish to check for, and an e-mail address to send reports to.

Begin by editing the `/etc/logcheck/logcheck.conf` file. Choose a `REPORTLEVEL` value (we use "paranoid"), and customize `SENDMAILTO` to point to a live e-mail address in your organization. You may also wish to enable `SUPPORT_CRACKING_IGNORE` if you're getting cracking-level events that are false positives (if you aren't, leave this at its default value for now).

Next, edit `/etc/logcheck.logfiles` to include the path to the log files you wish to check. If you're using our syslog-ng config above, you'd list each weekday rotation file:

```
/var/log/logcheck/Sun.log
/var/log/logcheck/Mon.log
# ... and so on ...
/var/log/logcheck/Sat.log
```

Finally, feel free to edit `/etc/logcheck/header.txt` to include a custom message at the start of each e-mail.

Depending on your setup, you may need to enable the logcheck script in cron. Under Debian, you must edit the file `/etc/cron.d/logcheck` and uncomment the line that starts with:

```
#2 * * * *    logcheck    ...
```

Once you've made these edits, logcheck should begin checking your logfiles and mailing you reports. The crontab line above would run logcheck every hour at 2 minutes past the hour. Check your distribution's documentation for more details.

11.4.5 Custom Rules

You should let logcheck run during a period of normal traffic on your network. You'll receive several alerts, and probably some false positives. Once you've verified that a false positive is really false, you can write rules to exclude the message from future reports.

Before we get to writing our own rules, here's a brief introduction to rule processing in logcheck. Each message in a log file is scanned, and then checked against the following conditions (in order):

1. Does the message match a rule in `cracking.d`? If so, the message is marked as an **attack alert**, and processing continues.

2. Does the message match a rule in `cracking.ignore.d`, and is the global `SUPPORT_CRACKING_IGNORE` enabled? If so, the message is **ignored** and **processing stops**. If the message does not match, but has been marked as an **attack alert**, then it is included in the outgoing e-mail, and processing stops. Otherwise, processing continues.

3. Does the message match a rule in `violations.d`? If so, the message is marked as a **security event**, and processing continues.

4. Does the message match a rule in `violations.ignore.d`? If so, the message is **ignored** and **processing stops**. If the message does not match, but has been marked as a **security event**, then it is included in the outgoing e-mail, and processing stops. Otherwise, processing continues.

5. Does the message match a rule in `ignore.d`? If so, the message is **ignored** and **processing stops**. Otherwise, the message is included in the outgoing e-mail.

As you can see, the rules follow an "include/exclude" pattern, with a default "include" at the end for any unmatched messages. When you create your own rules, you must be careful to match rules at the correct level (so as to not mask rules at a lower level), and to only match exactly what you need.

In general, you should write "include" rules wherever you think most appropriate (*e.g.*, **cracking** or **violation**), and write "exclude" rules at the same level as the rule that creates the false positive.

All matching rules in logcheck are written as *extended regular expressions* (as understood by the `egrep` program). A full discussion of regular expressions is far beyond the scope of this document; please consult other internet sources or books for more information. Also, feel free to "steal" from existing rules that are known to work!

To create a rule, simply code up a regular expression and drop it in a file. We suggest that you name the files with a local prefix (*e.g.*, "suffield"). For example, if you are getting log messages similar to the following:

```
[dhcpd] Wrote 0 new dynamic host decls to leases file.
```

You could ignore them by placing ignore rules in a file called `suffield-dhcpd`, and saving the file into the appropriate `ignore.d` directory. The ignore rule itself might look like:

```
\[dhcpd\] Wrote [0-9]+ new dynamic host decls to leases file
```

(The "[0-9]+" is a regular expression pattern; if you don't know what it does, you should consult external resources for more information.)

You may place more than one regular expression in a file, one per line. Remember the order that rules are evaluated in; you may need to create overly-broad violation rules, and then selectively ignore certain cases.

Chapter 12

OpenDirectory (LDAP/Kerberos)

Last updated 2008/03/18

Chapter 13

Name of Project Here

Last updated 2008/03/18

13.1 Introduction

Suffield provides networked file server space for all of its users. This gives people a place to back up their important files, share files with others, and access their files from machines that aren't their own.

Additionally, users may create web pages in their file server accounts, which are automatically hosted on our internal webserver.

Finally, school-owned desktop and lab computers directly attach to file server storage when a user logs in. This way, a user's files "follow" them around from machine to machine, making access and backup simple.

This document describes how to set up a Mac OS X Server machine as a central file server. We cover configuration for Macintosh and Windows clients, as well as networked home directories for Macintosh and Windows. Finally, we'll discuss exporting files via NFS.

13.2 Initial Setup

Begin with a fresh install of Mac OS X Server 10.4. Note that you'll most likely want an **unlimited client license**, or you'll be severly limited in the number of connections your server will support.

Perform a standard installation. When running the **Server Setup Assistant**, you should be careful when choosing the **Computer Name**. This is the name that will be shown to users when they connect to the fileserver, so it should be something meaningful that they will understand.

When asked which services to start automatically, you should choose **Apple File Sharing**. Additionally, if you wish to let Windows users connect to the server, you should enable **Windows File Sharing**.

Once the base install is complete, run any pending software updates.

At this time, you should connect any external drives you plan to use for shared data. External disks, RAID arrays, or other media should all be connected, named, partitioned, and ready to use before continuing.

13.2.1 Open Directory Integration

If your network has one or more Open Directory servers, and you plan to authenticate file sharing logons using these servers, you'll need to set up directory authentication:

1. Open the **Directory Access** application in `/Applications/Utilities`.
2. If necessary, click the lock icon and authenticate.
3. Click on the item named **LDAPv3** and click the **Configure** button.
4. In the window that appears, *deselect* the **Add DHCP-supplied LDAP servers to automatic search policies** box.
5. Click the **New** buton.
6. Enter the name or IP address of your Open Directory server.
7. Ensure that the **Use for authentication** box is *selected*.
8. Click **Continue**.
9. Optionally, bind the machine to the OD server by entering the machine name and directory admin information. Click **Continue**.
10. In a similar fashion, add any backup Open Directory servers on your network.

Quit the **Directory Access** application when you are done.

13.2.2 Joining Kerberos

If you wish to gain the benefits of single-signon using Kerberos, you must first join your server to the OD server's Kerberos domain.

1. On the OD master server, open the **Server Admin** application.
2. Click on the **Open Directory** service listing.
3. Click on the **Settings** tab at the bottom of the window.
4. On the main settings screen, click the **Add Kerberos Record** button.
5. For the **Administrator Name** and **Password**, enter the master password for your OD domain.
6. For the **Configuration Record Name**, enter the name of the server you wish to join to the domain. The server must be bound to the domain, as described in the previous section. If you're unsure of the record name, look in **Directory Access** for the server you're trying to join. It should look something like "`cn=myserver,cn=computers,dc=suffieldacademy,dc=org`". Use the name part of this record (*e.g.*, "myserver").
7. For the **Delegated Administrators**, enter the name(s) of administrative users who will add the computer to the domain. These users must be in the OD domain (they cannot be local to the server).
8. On the computer you're adding to the domain, open **Server Admin**.
9. Click on the **Open Directory** service listing.
10. Click on the **Settings** tab at the bottom of the window.
11. Click the **Join Kerberos** button.
12. Enter one of the valid administrative usernames and passwords you specified on the OD server.
13. If no errors are reported, you're done!

13.3 AFP Sharepoints (Macintosh Clients)

AFP sharepoints are the native method for clients running Mac OS X. If you wish to share files to Macintosh clients, you should enable AFP sharepoints.

13.3.1 Server Admin Settings

In **Server Admin**, click on the **AFP** service entry.

Under the **Access** tab, *select* the **Enable Guest access** checkbox.

Under the **Logging** tab, enable any logs you wish to keep on the server.

Save your changes by clicking the **Save** button at the bottom of the screen.

13.3.2 Workgroup Manager Settings

Open **Workgroup Manager** and click on the **Sharing** icon at the top of the window.

From this point, you may add, modify, and delete share points on the server.

To add (or modify) a sharepoint:

1. Click on the **All** tab and navigate to the folder you wish to share.
2. Under the **General** tab, *select* the **Share this item and its contents** checkbox.
3. Under the **Access** tab, ensure the folder permissions are how you want them. Pay particular attention to the **guest** settings; if guests should not be allowed to access a share point, make sure they do not have rights to the folder.
4. Click **Save**. Move to the **Protocols** tab.
5. Move through each protocol section, and only enable those that you need for this share. See other sections of this document for information on the settings for Windows and NFS clients.
6. Again, check to confirm that you've enabled guest access only if you need it.
7. Save your changes and test your share point. You should be able to connect to the server with a valid name and password (or, with guest access if you've enabled it).

13.4 CIFS Sharepoints (Windows Clients)

CIFS sharepoints are most frequently used by computers running Windows, though UNIX, Mac OS X, and other clients are also cable of speaking the protocol.

CIFS shares may be enabled simultaneously with an AFP share, or it may exist on its own (*e.g.*, for Windows profile directories, which are not needed on the Mac).

13.4.1 Server Admin Settings

In **Server Admin**, click on the **Windows** service entry.

Under the **General** tab, set the **Role** of the server to **Domain Member**.

Enter the name and description of the machine, and set the domain to the proper value for your network (*e.g.*, `SUFFIELDACADEMY`).

Under the **Access** tab, decide if you wish to allow guests or not.

Under the **Advanced** tab, *deselect* the **Workgroup Master Browser** and **Domain Master Browser** checkboxes (we assume you have set up a valid Primary Domain Controller on another machine; if this is your PDC, leave the boxes checked).

If you're using WINS on your network, enter the IP address of your WINS server to register your computer's name.

You will probably want to enable **Virtual Share Points**, which allows users to connect directly to their home folder without needing the intervening path information.

Click **Save**. You will be prompted to enter a Open Directory Administrator password, which adds your computer to the domain.

13.4.2 Workgroup Manager Settings

Open **Workgroup Manager** and click on the **Sharing** icon at the top of the window.

Select a share point from the pane on the left, and click the **Protocols** tab on the right.

Choose **Windows** from the drop-down menu to edit the settings related to CIFS shares.

If you wish to share this folder to CIFS clients, *select* the **Share this item using SMB** checkbox. You may also *select* the **Allow SMB guest access** checkbox if you wish to allow guests.

Click **Save** to save your changes, and your share point should now be available to Windows clients.

13.5 Macintosh Network Home Directories

13.5.1 Enabling Local Home Directories

If you plan to let users log in to the file server directly (*e.g.*, via SSH), you'll find that the network home directories do not work, because they map back to the server.

To override this, we must tell the server to ignore the home directory attribute from the OD server, and replace it with a custom local value instead:

1. Open the **Directory Access** application, in `/Applications/Utilities`.
2. If necessary, click on the lock icon and authenticate.
3. Click on the item named **LDAPv3**, and click the **Configure** button.
4. For each of the servers in your search list, do the following:
   (a) Click on the **LDAP Mappings** popup menu, and choose **Custom**.
   (b) Click on the **Search & Mappings** tab at the top of the window.
   (c) Expand the **Users** section of the mapping list.
   (d) Find the attribute named **NFSHomeDirectory** and click on it.
   (e) In the right-hand pane, delete the current value, `homeDirectory`.
   (f) Click the **Add** button, and create a new value that stars with a pound sign (`#`) and contains the local path to your users' home directories. You may use the macro "`$uid$`" to add the user's login name. For example:

   ```
   #/Volumes/BigRaid/Users/$uid$
   ```

   If a user named `jbogus` logged in, the home directory would be set to `/Volumes/BigRaid/Users/jbogus`.
   (g) Click **OK**. Repeat the above steps for all bound OD servers.
5. Click **OK** on the main screen when you've completed your changes.

You can check your settings by logging in to the terminal and typing the following:

```
lookupd -d
```

This begins an interactive version of `lookupd`, which merges directory information on Mac OS X. Type the following command:

```
userWithName: jbogus
```

(Substitute a real username for `jbogus`.)

You should see a list of attributes for this user. Under the `home` attribute, you should see your new mapped value, rather than the network-mounted default.

13.6 Windows Roaming Profiles

1. Make sure the domain is set up properly (check domain WINS resolution).
2. Set perms on /etc/netlogon to 755 root:staff (copy in scripts and default profile)
3. Set perms on /Profiles to 770 root:staff
4. Set ACL on /Profiles to be "Suffield" (or whatever group all your users belong to), full read, create folder, this folder only.
5. Edit /etc/smb.conf, in Profiles section, set create mask to 0640 and dir mask to 0750.
6. Make fileserver version of file uchg (not necessary on PDC and BDC)

13.7 NFS Exports

NFS is the traditional file sharing mechanism for UNIX clients. It allows a machine to mount an entire directory and make it available to all of its clients, with proper permissions and ownership settings.

NFS does have its warts, however, including a bad security track record. Therefore, we only recommend enabling NFS for read-only, non-root exports. That minimizes exposure to security flaws that could corrupt your data.

13.7.1 Server Admin Settings

NFS is automatically enabled when share points are available for other machines to use. Thus, you do not need to explicitly start any services in **Server Admin**.

You may change a few parameters for the NFS daemon itself, but you should not need to do this unless you know what you are doing.

13.7.2 Workgroup Manager Settings

Open **Workgroup Manager** and click on the **Sharing** icon at the top of the window.

Select a share point from the pane on the left, and click the **Protocols** tab on the right.

Choose **NFS** from the drop-down menu to edit the settings related to NFS shares.

Because NFS is not a very secure protocol, we recommend restricting the export as much as possible.

If possible, use the **Client** or **Subnet** designation for the share point, to restrict which machines can connect to the share. Try to restrict it as much as possible, including only the hosts you trust to access the share.

If possible, export the share **read-only** to prevent clients from making changes.

If you need to preserve ownership of files for clients, *deselect* the **Map all users to nobody** checkbox. If you simply need to export all the files (and ignore ownership), leave this box *selected*.

You should *always* leave the **Map root user to nobody** box *selected*, unless you know what you are doing.

Chapter 14

Squid (WWW Proxy Server)

Last updated 2008/03/18

14.1 Introduction

A large portion of our internet link is used for web (HTTP) traffic. In addition to popular websites, many key pieces of software (system updates, linux packages, and multimedia programs) make use of HTTP to deliver their content.

To increase the efficiency of our internet connection, we use a **caching web proxy**. A web proxy makes the HTTP request on behalf of the client, and caches the result for later. That way, if multiple users request the same document, the cache can serve its local copy, rather than wasting bandwidth. This has two benefits: users get their content faster (when its cached), and the internet connection isn't bogged down with duplicate requests.

14.2 Suffield's Requirements

At Suffield Academy, we use Squid as our proxy server. Briefly, these are the features that Squid supports that make it attractive for us to use:

- Squid can act as an **interception** (or **transparent**) proxy. This means that the clients on our network don't need to be reconfigured; all web

traffic is automatically cached without them needing to do anything. (We put our proxy inline with other network devices, though it is possible to set up routing rules to forward web traffic to the proxy.)

- Squid can use external **redirectors** for every request. This means that we can log or block unwanted sites using 3rd-party blacklists (note that we don't do this currently, as Suffield does not block traffic).

  Additionally, redirectors can be used to circumvent requests to unwanted URLs (such as those for advertisements). We use such a program, which makes pages load faster (no need to fetch the ads), and also prevents users from having to see unwanted advertisements.

- Squid has a feature called **delay pools**, which allow us to specify per-user and global transfer rates. This means that we can throttle users who consume more than their fair share of bandwidth. Additionally, we can apply rules to specific types of resources to prevent them from being abused (for example, we can throttle connections that go to http://www.example.com).

All of these features make Squid an extremely attractive piece of software. It's relatively straightforward to compile and set up, and it's Free software.

14.3 Hardware

Squid has one basic job: fetch data over the network, and store a copy on the local machine. Information is kept in RAM when possible, and on disk the rest of the time.

Thus, the two most important factors in hardware are the amount of RAM given to Squid, and the speed of the disk subsystem used to store the cache data.

The Squid listserver archives contain numerous messages about the "recommended" machine configuration for Squid. Unfortunately, there is no good answer to this question, as every site's needs are different.

Our first Squid machine was a Pentium III 850MHz with 1.5GB of RAM. It had two 36GB SCSI disks, which we ran in a RAID-1 configuration. The machine performed well, but as our internet connection speed grew, it began to show its age. We especially wanted a faster processor so we could have the flexibility to run filtering software inline with the proxy.

Our current machine is a P4 3.4GHz with 3GB of RAM. The machine has 6 15K SCSI disks. One is used for the system, and the others are used for the Squid cache. As our needs increase, we can add more drives to the system, and split them onto different controller cards. We no longer run the machine in a RAID configuration, as that just brings a speed penalty (the Squid daemon automatically balances load between multiple disks for us).

14.4 Custom Compilation

We use our Squid box as a bridging interception proxy. This means that the machine acts as a bridge, and is inserted "inline" between other network devices (in our case, just before our firewall). Additionally, Squid is configured as an interception proxy, which means that it will grab requests destined for port 80 (the standard HTTP port) and run them through the caching engine. Connections not bound for port 80 are passed straight through the bridge.

This setup is advantageous because it makes the proxy completely invisible to the rest of the network (and the users). If we need to, we can simply take the machine out of the network and everything will continue to work just as before (except that it's not being cached). No settings on the client machines need to change.

However, this setup requires compiling our own Linux kernel, and also our own version of Squid. We'll walk you through the steps below.

Note: it is possible to set up squid to work in a non-bridged mode (all of our custom compilation in this section deals with a bridged version of Squid). This requires some other means of getting the packets to your squid box, such as via WCCP or router next-hop mangling. We used to do this (we had a `route-map` command on our Cisco core switch), but the routing was done in software, and it drove up the load on our core. We've now switched to an inline approach, which doesn't impact performance on our other gear.

Note: these steps assume Debian Linux ("Etch" is the release we're using). If you use a different system you may need to compile according to your distribution's conventions.

14.4.1 Preparing The System

First, set some environment variables so your custom builds will have sane values in the description files:

```
export DEBFULLNAME="Your Name"
export DEBEMAIL="you@example.com"
```

Now we need to download all the components necessary to build packages on our system. Make sure your package repository listing is up-to-date:

```
apt-get update
```

Move into the source directory on the machine:

```
cd /usr/src
```

Now we need to download a special patch for bridging proxy support in the kernel and iptables, and unpack it.

```
wget http://www.balabit.hu/downloads/files/tproxy/obsolete/linux-2.6/cttproxy-2.6.18-2.0.6.tar.gz
tar -zxf cttproxy-2.6.18-2.0.6.tar.gz
```

We are now ready to fetch and build the three other components we need: the kernel, iptables, and squid.

14.4.2 Kernel Compilation

First, fetch the kernel package and all of the packages necessary to build a kernel (you can put the following command all on one line):

```
apt-get install linux-source-2.6.18 kernel-package libncurses5-dev \
fakeroot bzip2 build-essential dpatch devscripts
```

This will put a tarball of the Linux source in /usr/src. Go there and unpack it:

```
cd /usr/src
tar -jxf linux-source-2.6.18.tar.bz2
```

Now move into the Linux source directory:

```
cd linux-source-2.6.18
```

You need to apply the patches included with the tproxy download you got earlier:

```
for patch in ../cttproxy-2.6.18-2.0.6/patch_tree/*.patch
do patch -p1 < $patch
done
```

Now it's time for a standard Debian kernel build. I suggest making based on the existing config:

```
make oldconfig
```

(You may also use `make menuconfig` for a graphical interface.)

That will ask you about the new features added by the patch. You want to enable the following:

```
Transparent proxying (IP_NF_TPROXY)
tproxy match support (IP_NF_MATCH_TPROXY)
TPROXY target support (IP_NF_TARGET_TPROXY)

NAT reservations support (IP_NF_NAT_NRES)
```

You're now ready to build the kernel:

```
make-kpkg clean
make-kpkg --rootcmd fakeroot --initrd --append-to-version=-tproxy.1.0
kernel_image
```

Wait a while for the kernel to compile. When it's done you'll have a linux-image package in `/usr/src`. Go ahead and install it:

```
cd /usr/src
dpkg -i linux-image-2.6.18*.deb
```

If the installation was successful, then we're almost done (with the kernel, anyway).

We want to load the tproxy modules by default, so add the following lines to the end of your `/etc/modules` file:

```
ip_tables
iptable_filter
ipt_TPROXY
ipt_tproxy
```

Reboot your machine into the new kernel. You can confirm the version by running `uname -a`. You can confirm that tproxy was installed by running:

```
dmesg | grep TPROXY
```

14.4.3 IPTables Compilation

The new version of the kernel has a patched version of iptables support, which requires that we recompile the iptables binaries to match.

First, move into your source directory:

```
cd /usr/src
```

Next, get the source for iptables:

```
apt-get source iptables
apt-get build-dep iptables
```

Move into the source directory:

```
cd iptables-1.3.6.0debian1
```

Patch the sources (note change of directory to inner iptables dir):

```
cd iptables
patch -p1 < ../../cttproxy-2.6.18-2.0.6/iptables/iptables-1.3-cttproxy.diff
chmod +x extensions/.tproxy-test
```

Additionally, we need to copy our patched kernel sources into the iptables tree so it can build against them. Change back to the root iptables source tree:

```
cd /usr/src/iptables-1.3.6.0debian1
```

Now move the default linux dir out of the way:

```
mv linux linux-orig
mkdir linux
```

And copy your linux files in:

```
for x in COPYING Makefile include net
do cp -a ../linux-source-2.6.18/$x linux/$x
done
```

Now you're ready to build. Susbstitute your e-mail address in the build command:

```
dpkg-buildpackage -rfakeroot -us -uc -b -myou@example.com
```

You'll end up with an iptables `.deb` file in your `/usr/src` directory. You can move into that directory and install the package:

```
cd /usr/src
dpkg -i iptables_1.3.6.0debian1-5_i386.deb
```

To confirm that everything is working correctly, try the following command:

```
iptables -t tproxy -L
```

If that doesn't give an error, then all of the modules are installed correctly, and you're ready to go.

14.4.4 Squid Compilation

For maximum performance, the Squid maintainers recommend that you compile your own version of Squid from scratch. Additionally, our "tproxy" setup is not included in Debian, so we must compile our own anyway.

Change into the source directory:

```
cd /usr/src
```

Fetch the source for squid, and the build dependencies:

```
apt-get source squid
apt-get build-dep squid
```

Move into the `squid-2.6.5` directory.

The Debian package-building system includes a `rules` file that dictates what configure options to use when building the package. We'll edit this file to include only the options that we want (you may wish to choose other options depending on your setup).

The Debian build has reasonable defaults for Linux (async-io, aufs, etc). We simply edit the rules file to remove code we don't need (mostly dealing with authentication methods). We also need to add tproxy support to the build.

In the `debian/rules` file, make the following changes:

- Add `--enable-linux-tproxy` to the list of build options
- Add `--enable-multicast-miss` to the list of build options
- Add `--disable-ident-lookups` to the list of build options (ident just slows us down)
- Remove `--enable-useragent-log` (don't need to log User Agent data)
- Remove `--enable-referer-log` (don't need to log referer)
- Remove `--enable-underscores` (don't allow illegal hostnames)
- Change the `--enable-auth` line to say `--enable-auth="basic,digest"` (Don't need NTLM auth)

Save the file when you're done.

We need to make the tproxy headers available to Squid for compilation. You can do this by running the following:

```
cp /usr/src/linux-source-2.6.18/include/linux/netfilter_ipv4/ip_tproxy.h \
/usr/include/linux/netfilter_ipv4/
```

You must also install `libcap-dev` to get the capabilities file for compiling Squid (if you don't do this, you'll see "CAP_NET" errors):

```
apt-get install libcap-dev
```

Also, **be sure** to add `capability` to your `/etc/modules` file (Debian doesn't load capabilities by default).

You may wish to set certain compiler flags (*e.g.*, `CFLAGS='-O2 -march=pentium4'`) before the build to include any processor-specific options. Note that these packages may only work for the architecture they're compiled for!

Now, build the package! Susbstitute your e-mail address in the build command:

```
dpkg-buildpackage -rfakeroot -us -uc -b -myou@example.com
```

Once you've built the binary installation packages, you're ready to install them. They'll appear in the parent directory to your source folder, and there should be 4 Debian package files:

```
squid_2.6.5-6etch1_i386.deb
squid-cgi_2.6.5-6etch1_i386.deb
squidclient_2.6.5-6etch1_i386.deb
squid-common_2.6.5-6etch1_all.deb
```

You'll need to install any dependencies manually. At the moment, this includes `lsb-base` (for squid) and a web server (for squid-cgi).

To install `lsb-base`:

```
sudo apt-get install lsb-base
```

For a web server, you can go with the tried-and-true apache. However, we want as lightweight a server as possible, so we opt for `thttpd`:

```
sudo apt-get install thttpd
```

You can install the Debian packages using `dpkg -i`:

```
sudo dpkg -i /usr/src/squid*.deb
```

Once this is done, you should have Squid installed, and ready to configure!

14.5 Configuration

At this point, you should have Squid installed on your system. You can confirm this by typing `squid -v` on the command line. You should get back Squid's version, along with any compile-time options.

If you don't have Squid yet, visit the previous section for information on compiling it from scratch, or install a packaged version from your OS vendor. For example, you could say this under Debian:

```
sudo apt-get install squid squidclient squid-cgi
```

Now that you've got Squid, it's time to configure it. Squid has a great deal of run-time options, which affect how it caches documents. We've tuned our config to work with our particular needs; the descriptions below explain the choices we've made. You should make changes as you see fit.

14.5.1 Bridge Setup

We configure our system as a network bridge, which means that it sits between two physical devices on our network and relays the packets between them. However, there's a twist: we intercept certain packets (those destined for port 80) and shunt them to Squid for processing.

You'll need two ethernet cards in your machine to bridge between (one "in" and one "out", as it were). You can use another card for a management IP address, or you can actually assign an address to the bridge itself and reach the machine just as you would a "real" interface.

In order to set up the bridge, we need to make a few tweaks to the system. First, we need to install some software that's necessary for setting up a bridge:

```
apt-get install bridge-utils
```

Next, edit `/etc/network/interfaces`. You should already have a stanza for a statically configured interface (*e.g.*, `eth0`). Keep the settings for the stanza, but replace the interface name with `br0`. Also, add the line `bridge_ports ethXXX ethYYY` to add them to the bridge. For example:

```
auto br0
iface br0 inet static
    bridge_ports eth0 eth1
    address 192.168.0.100
    netmask 255.255.255.0
    gateway 192.168.0.1
```

Additionally, if your setup is like ours you'll need to add some routing to the box so it knows where to send packets. Our Squid box sits just between our firewall/router and LAN. Thus, it needs to be told how to route packets to the LAN and packets to the outside world. We do this by specifying the firewall as the "gateway" in the `interfaces` file, and adding a static route for our LAN. Thus, you would add the following lines to `/etc/network/interfaces` in the `br0` stanza:

```
up route add -net 192.168.1.0/24 gw 192.168.1.1
down route del -net 192.168.1.1/24 gw 192.168.1.1
```

We'll need to tell the kernel that we're going to forward packets, so make sure the following are set in `/etc/sysctl.conf`:

```
net.ipv4.conf.default.rp_filter=1
net.ipv4.conf.default.forwarding=1
net.ipv4.conf.all.forwarding=1
```

Once you're all set, the easiest thing to do is reboot for the bridge config to take effect. The other settings should now be working also. `cat /proc/sys/net/ipv4/ip_forward` to confirm that the machine is in forwarding mode.

14.5.2 Disk Setup

Squid is going to need a place to store its data. On a single-disk system, this can be a directory or partition. On a multi-disk system (such as ours), we give entire disks over for Squid to use.

The listserver archives contain some debate about the best filesystem to use on the partition(s) for the Squid cache. From our reading, there appear to be 3 choices:

- ext2 (a non-journaled Linux filesystem). It's very fast and lightweight, but the lack of journalling can mean long recovery times in the event of a crash.
- reiserfs (a journaled Linux filesystem). Also reported to be fast, and very efficient with directories containing large numbers of files (as with a cache).
- xfs (a journaled filesystem from SGI). Designed for high-demand multimedia storage (large files, large numbers of files).

We ruled out ext2, as we wanted a journaled system that could recover from crashes easily. That left the two journalled filesystems, reiserfs and xfs. Reiser

seems to have a loyal following, though users who have tried xfs report that it works very well (especially with the epoll, which is on by default in Linux and Squid 2.6). Because we use xfs as our standard Linux filesystem on our servers, we decided to use it for our cache as well.

Note that regardless of the filesystem selected, Squid users recommend using the `noatime` option for any cache partitions. This option prevents the system from updating the *last access time* for cache files, as it is unecessary and just slows down the system.

Make sure that the directories are writable by your proxy user. On a Debian system, this means executing:

```
sudo chmod -R proxy:proxy /var/spool/squid
```

(Substitute your cache directory for `/var/spool/squid`.)

Also, if you add, remove, or change the locations of the cache directories, be sure to run `squid -z` to rebuild the cache directory structure.

14.5.3 Squid Configuration

The main configuration file is `/etc/squid/squid.conf`. It controls nearly every aspect of Squid's behavior, from memory consumption to ACLs to exceptions for caching rules.

We start with a stock Debian Squid configuration file and change it as necessary. Rather than describe the changes here, we comment the configuration file carefully. Simply search for the word **suffield** in the configuration file to find all of our changes.

Remember: we're building a bridged, intercepting (transparent) caching proxy with ad-blocking/monitoring. The options, numbers, and paths are specific to our setup and our machine. If you are building your own configuration, be careful to substitute values that work for your site.

Download the Suffield Squid configuration file (squid.conf)

14.5.4 TPROXY Rules

We've set our server up as a bridge, meaning that packets received on one network interface are sent out on the other interface. In this sense, the server "looks like" a straight wire to the rest of the network.

In order for squid to perform its work, we must divert packets passing over the bridge that we'd like to cache. Under Linux we can do this, which makes the

bridge not quite a bridge anymore (some people call it a "brouter", since it's more like a bridge combined with a router).

Our patched version of iptables is capable of doing everything we need. You should have already enabled ip forwarding via `sysctl` above. Once you have, you're ready to enable the redirection:

```
/sbin/iptables -t tproxy -A PREROUTING -p tcp -m tcp \
--dport 80 -j TPROXY --on-port 81
```

(The command can be written on a single line by removing the backslash.)

A quick breakdown:

- `-t tproxy` means to affect the `tproxy` part of the routing tables. This is a special table that only exists if we've patched our kernel as described earlier in this document.
- `-A PREROUTING` means to "Add" a rule to the PREROUTING chain. This means that our rule will apply before the kernel applies its standard routing rules. This is what we want, since we're basically circumventing the standard rules.
- `-p tcp -m tcp` means to match only TCP traffic (not UDP or other protocols)
- `--dport 80` means to intercept traffic destined for port 80 (the port that HTTP traffic uses by default)
- `-j TPROXY` means to "Jump" to the TPROXY section in the kernel. tproxy will deal with maintaining the proxied connection's state, and do all the other "magic" for us.
- `--on-port 81` this is the local port on this machine to forward the request to. This should be the port that your squid instance is listening on, as defined in your `squid.conf` file. The line must read `http_port <port num> transparent tproxy`, with an actual number substituted for `<port num>`. We've selected a non-standard port of 81 so we can use the default squid port (3128) for manually-configured proxy hosts (you can't mix standard and transparent hosts on the same port). You can pick any port you want (8080, 8000, 666, etc), so long as it isn't in use. Just make sure that the port number matches in the squid config and your iptables rule.

That rule enables the redirection of packets to squid. You may want to add other options to is (for example, you probably only want to redirect packets

that are sourced from your trusted network range). Read the manual page for iptables for more information on these options.

To remove the rule and return to pure bridging, simply execute the same command, but substitute `-D` for `-A`. This says to "Delete" the rule, rather than add it:

```
/sbin/iptables -t tproxy -D PREROUTING -p tcp -m tcp \
--dport 80 -j TPROXY --on-port 81
```

Read on in the next section for a way to automatically keep these rules in sync with the state of squid.

14.5.5 Watching Squid

We now have two independent processes (Squid and iptables) that must cooperate to make a functioning box. If we have squid running, but haven't enabled the iptables rules, nothing will be proxied. Meanwhile, if the iptables rules are enabled but squid isn't running, all web traffic will be blocked (making for unhappy users).

We've written a script to keep these items syncronized together. It runs all the time, periodically polling squid to make sure its running. If squid suddenly stops running, the script disables the iptables rules. Similarly, if squid returns, the rules are re-enabled. Finally, the script can also receive explicit commands to enable or disable the iptables rules (useful if you're planning to turn squid off; you can shut down the rules first to make for a smooth transition).

You can download a copy of our squid-watcher script. It can be installed anywhere on the machine (we use `/usr/local/bin`).

You'll need to edit the script and check the variables at the top. They define how frequently to poll squid, and how to alert you if something stops working (syslog or e-mail).

Also, the script defines two "hooks" where you can specify the commands to run when squid starts and stops. This is where you should put your iptables rules (our rules are there; you can modify them to suit your needs).

To set it up to run, you'll need to add a line to `/etc/inittab` that looks like this:

```
sq:2345:respawn:/path/to/squid-watcher poll < /dev/null > /dev/null 2>&1
```

Make sure to set the correct path to the script.

Once everything is ready to go, reload INIT (we assume init is PID 1 on your machine; use `ps -e | grep init` to confirm):

```
kill -HUP 1
```

You should see a message in the system logs, or via e-mail (depending on how you configured logging) confirming that the script has started.

If squid is running, you can force the script to enable your rules by running:

```
squid-watcher start
```

To disable your rules, you can run:

```
squid-watcher stop
```

Finally, you can ask the script about its current state by running:

```
squid-watcher info
```

You'll see a message in the logs telling you what the current state is. It will be one of the following:

```
RUNNING - Squid is running, and the script's rules are enabled
STOPPED - Script's rules are disabled (squid's status is not checked)
BROKEN  - Script tried to enable rules, but squid is not running
```

The script will automatically move between the `RUNNING` and `BROKEN` states, depending on the state of squid. The `STOPPED` state prevents the script from changing anything (useful when you know that you're turning squid off, and don't want the script to recover for you).

You should add commands to your squid init scripts (see below) to make sure that you enable and disable the script when appropriate.

14.5.6 Initialization Scripts

These commands can be added to the Squid init scripts, which live in `/etc/init.d`. You can see the changes we've made by downloading our Squid init script.

Basically, whenever we tell squid to start, we also tell our `squid-watcher` script to start as well. Similarly, just before we stop squid we tell `squid-watcher` to stop monitoring (so it doesn't freak out when squid terminates). Other than that, it's pretty much the stock config.

Other minor changes

We've also tweaked our init script to perform other minor tuning. We set the "swappiness" of the VM in the kernel to prevent the machine from swapping too aggressively (we'd rather use the memory for caching!).

You can do this by adding:

```
vm.swappiness=10
```

to `/etc/sysctl.conf`

14.6 Cache Manager

Squid includes a `cachemgr` CGI binary that allows you to query a running Squid instance for statistics. These stats can help you determine if Squid is running efficiently.

You do not need to install `cachemgr` on the same machine as Squid (for example, you may wish to install it on a standard web server). We choose to run `cachemgr` on the local machine in a lightweight HTTP server such as `thttpd`.

14.6.1 Configuration

The `cachemgr` program has its own configuration file in `/etc/squid/cachemgr.conf`. The only configurable option is a list of server addresses, ports, and descriptions. Enter the addresses of the cache(s) you wish to monitor.

14.6.2 Security

By default, Squid does not allow `cachemgr` to perform destructive operations (shutdowns, reloads, *etc.*). Even if you don't enable these features, however, you may wish to restrict access to `cachemgr`, either through firewall rules or web server access controls.

14.7 Squid Statistics

To keep track of how well your Squid installation is doing, it's helpful to analyze the logs that it produces. Numerous log analyzers exist for Squid; some are text-only, others HTML, and others are graphical.

At Suffield, we've settled on an analyzer called Calamaris:

http://cord.de/tools/squid/calamaris/

The software has few requirements, is written in Perl, and can produce text and HTML reports. The latest beta (V2.99.4.0 as of this writing) also supports creating graphs in the HTML version.

14.7.1 Downloading and Installing Calamaris

Debian includes a version of Calamaris in its software repository. Thus, you can install it directly using:

```
apt-get install calamaris libgd-graph-perl
```

For other platforms, check the software repository to see if a packaged version is available. Otherwise, you may download the script from the website listed above.

14.7.2 Configuring Calamaris

The Calamaris configuration file is really just a large Perl file that gets included at runtime. You may set all the options relating to how the script produces output in this file. Our version of the config file is commented with all the changes we've made, and you can download it from our web server. All of our changes are commented with the word "suffield".

Additionally, Debian provides another file called `cron.conf` which defines the reports that get generated, where they are stored (or e-mailed), and how often to generate the reports. You may download our version of this config file to see how we've configured it.

Finally, Debian uses a script launched from `cron` to execute Calamaris with the proper options. We've edited this script to make it work more the way we want. There are only two major changes:

1. Modify the report format to produce frameset-HTML with embedded graphs.
2. Modify the output file and directory names to be specified by date (rather than just overwriting the logs each time).

The changes are well-commented and pretty straightforward. Download our version of the cron script to see the changes we've made.

14.7.3 Running Calamaris

Calamaris is automatically executed by `cron` every morning. The output is mailed or dumped to a file (or both) when processing completes.

If you choose to output to an HTML file, you should configure the script to drop the reports directly into your web server directory. The files will be made available immediately via your web server.

Chapter 15

Network UPS Tools

Last updated 2008/03/18

15.1 Introduction

To prevent data loss and corruption, Suffield has several **U**niterruptable **P**ower **S**upplies (**UPS**es) that provide battery-backed power in the event of an electrical outage.

Most of our UPSes have a serial port, which allows a computer running monitoring software to receive status updates from the unit. This way, the host can shut down when the batteries of the unit run low.

Because each of our UPSes have more than one server attached to them, we needed a centralized way to manage the UPSes and shut down all affected servers during an outage. We settled on a software package called NUT, which stands for **N**etwork **U**PS **T**ools.

The software is easy to compile for UNIX-based operating systems, and there is also a Windows client available. NUT provides monitoring, reporting, alerts, and a client-server model that allows for very complex setups involving multiple servers and UPSes.

15.2 Suffield's Setup

At Suffield Academy, we have a single centralized host that acts as the master server for UPS information. It has a multi-port serial card, and all UPS units

are attached directly to it. All other hosts connect to this server via the network and poll it for UPS status information.

This centralized setup allows us to easily add new servers to our power configuration. We simply add client software to the machine and configure it to poll a particular UPS instance on the server. The server automatically broadcasts power outage events to the clients, which shut themselves down before the batteries are completely drained.

15.3 Server Setup

Suffield uses a setup where all the UPS units are connected to a single centralized server. This section deals with setting up such a centralized system, though it is relatively straightforward to adapt it to a multiple-master configuration as well.

15.3.1 Install the Software

Get and install the `nut` package. This means compiling it from source, or downloading a version through your distribution's packaging system. Note that some package systems break NUT up into several pieces (*e.g.*, drivers, client, and server), so you may need to install more than one package.

Under Debian Linux (our platform of choice), this ought to do it:

```
apt-get install nut
```

15.3.2 Port Identification

The first step is to get a list of the port devices on your server, and figure out which ones go with which UPS. For example, if you've plugged a UPS into serial port 1, it might be connected on `/dev/ttyS0` under Linux. If you have multiple serial ports (perhaps through an add-on board or other interface), you'll need to come up with a mapping of all the device names to the ports they use.

We highly recommend labelling the ports externally so that you may easily verify their device assignments.

You must change the permissions on all ports so they are accessible by the daemon running NUT. Under Debian, the files should be owned to user `nut` and group `nut`. Additionally, make sure no other users have write access to the device:

```
sudo chown nut:nut /dev/<device node>
```

```
sudo chmod 660 /dev/<device node>
```

15.3.3 Configuring the UPSes

Begin by editing the `ups.conf` file. This file defines all the UPS hardware that is directly connected to this machine.

For each UPS, you must specify the following:

- A name for the UPS
- The device it is connected to
- A short description
- When to shut down the load (`sdorder`)
- How to shut down the load (`sdtype` – certain models only)

The `sdorder` flag allows certain UPSes to be powered down before others. Use this to enforce an ordering on the powerdown sequence (lower numbers are powered down before higher numbers; -1 excludes from the sequence entirely).

The `sdtype` is special to the `apcsmart` driver (which we use with our APC UPSes). It determines the shutdown command to send to the UPS (*e.g.*, power off but don't restart automatically, power off now, power off with delay).

A sample stanza might look like:

```
[nicole]
    driver = apcsmart
    port = /dev/ttyS0
    # Set load to turn back on automatically when power returns
    sdtype = 0
    desc = "Smart-UPS 1000"
    sdorder = 10
```

Note: we name our UPS hardware after bratty spoiled people (the "SSW" naming scheme), which is why you'll see several women's names in our sample configs.

15.3.4 Configuration Access Controls

Once you've defined the hardware attached to the machine, you must now specify who has access to the hardware (via the `upsd` daemon).

Configuring Network Access

Begin by editing the `upsd.conf` file. This file allows you to specify ACL statements defining network hosts (or blocks).

You must add two lines for each host or netblock you wish grant access to. The first defines the host/netblock with a name, and the second grants (or revokes) access. Any configuration should include entries for the local machine, and a default deny:

```
# Define blocks
ACL localhost 127.0.0.1/32
ACL all 0.0.0.0/0
ACL myservers 172.16.0.0/12

# Grant or revoke access (deny by default)
ACCEPT localhost
ACCEPT myservers
REJECT all
```

Finer-grained control of who can log in from where is possible by combining ACLs with user-based permissions (see next section).

Configuring User Access

You may further restrict who has access to which UPS (and what they can do with it) by specifying users. In our case, the "division of labor" is quite clear-cut (a server which controls all the hardware, and numerous clients that may only poll for status). Thus, we only need a few users: one with full control, one that can monitor and take action on all UPSes, and one that can only monitor.

If your setup is more complex, you may need additional users. Additionally, you may wish to create individual users for particular machines for finer-grained control.

Here's a sample minimum configuration:

```
# Local user with full manual control over all UPSes
[superuser]
    password = supersecret
    allowfrom = localhost
    actions = SET FSD
    instcmds = all

# upsmon agent user with full control over all UPSes
# (used for local monitoring agent on master server)
[masteruser]
    password = secret
    allowfrom = localhost
```

```
    upsmon master

# upsmon agent with read-only privs
# (used for remote monitoring agents on other machines)
[slaveuser]
    password = kindasecret
    allowfrom = localhost myservers
    upsmon slave
```

15.3.5 Configuring Monitoring

The server is now configured to "listen" to several pieces of UPS hardware. We must now specify which UPSes this machine will monitor and interact with, so that we can automatically shut down machines, send pages and e-mails, and take other actions.

We do this by editing the `upsmon.conf` file. This file is quite large, so we'll take several steps to make all the edits necessary.

Monitoring

You must specify the UPSes you wish to monitor. Since we have a "one central server" approach, we add monitoring lines for **all** UPSes attached to the machine.

The central server should be allowed to interact with the UPS hardware (*i.e.*, turn it off and on), so we use our "master monitor" name and password for each UPS.

```
MONITOR nicole@localhost 1 masteruser secret master
MONITOR paris@localhost 1 masteruser secret master
```

Note that each `MONITOR` line must have a "power value" associated with it, specifying how many outlets on the current machine are powered by this UPS. Unfortunately, setting the power value to 0 means that the system will not automatically report status on the UPS. Thus, for a central master system, you'll need to assign a positive integer for every UPS that you're managing.

This complicates things a little, as now you have a machine that appears to be powered by many UPSes. The remedy is as follows:

- Set the power value of the UPS(es) that provide power to the server high enough so that it is greater than the power value of all the other UPSes combined.
- Set the `MINSUPPLIES` value to this high value.

The net effect is that the UPSes connected to the server will be "weighted" to count more than the other UPSes. This way, the system won't actually initiate a shutdown until it's lost the UPSes that are cruicial to its operation, but will still process events from the other UPSes.

External Processes

NUT has the ability to run an external program whenever an event is received from a UPS. To make this as flexible as possible, NUT includes a binary called `upssched` which processes events from the UPS and fires off a script of your choosing when they occur.

It's a little complicated because of this extra binary, but it works well and is quite flexible.

Begin by editing `upsmon.conf` and adding an entry for `NOTIFYCMD`. We use the built-in `upssched` binary, as it provides several useful features (like timers):

```
NOTIFYCMD /sbin/upssched
```

(Note that the location of the binary may be different on your system.)

Once you've done that, move down in the file and add entries for `NOTIFYFLAG`. We must tell `upsmon` what to do with each event it receives from a UPS; it can be any combination of the following:

- Write a message to the system log
- Write a message to all logged in users
- Execute the `NOTIFYCMD`
- Ignore the event

We've opted to log all messages to syslog and execute `upssched`:

```
NOTIFYFLAG ONLINE SYSLOG+EXEC
NOTIFYFLAG ONBATT SYSLOG+EXEC
NOTIFYFLAG LOWBATT SYSLOG+EXEC
NOTIFYFLAG FSD SYSLOG+WALL+EXEC
NOTIFYFLAG COMMOK SYSLOG+EXEC
NOTIFYFLAG COMMBAD SYSLOG+EXEC
NOTIFYFLAG SHUTDOWN SYSLOG+EXEC
NOTIFYFLAG REPLBATT SYSLOG+EXEC
NOTIFYFLAG NOCOMM SYSLOG+EXEC
```

Now, save your changes and edit the file `upssched.conf`. This file determins what `upssched` does when it receives an event. At first, the file seems a little limiting; you may only specify one script to execute for **all** events. However, it provides one very important feature which makes it worth it: **timers**.

`upssched` has the ability to trigger a **timer** for any event, and to cancel a timer that has not executed. For example, you could set a 30-second timer to send an e-mail when an UPS goes on battery, and then cancel the timer when the UPS goes back on line power. This gives you an easy way to "buffer" events before they get acted on, which is very useful for power "blips" (*e.g.*, outages lasting fewer than 30 seconds).

Begin by declaring the command to execute for all events. We have written our own script, called `upssched-dispatch` (download it from our website), which is written in Perl:

```
CMDSCRIPT /usr/local/bin/upssched-dispatch
```

You may need to specify values for `PIPEFN` and `LOCKFN`, depending on your operating system layout and configuration defaults. For example, on a Debian system you need the following:

```
PIPEFN /var/run/nut/upssched.pipe
LOCKFN /var/run/nut/upssched.lock
```

Finally, you must specify the events you wish to act on. We do this by specifying `AT` commands in the config file. The general form is:

```
AT <event> <upsname> <action> <arguments>
```

The event may be any UPS event (*e.g.*, `ONBATT`, `COMMBAD`). The upsname is the name@host identifier for the UPS. The action is one of `EXECUTE`, `START-TIMER`, or `CANCEL-TIMER`. The arguments get passed to your command script.

You need to pick an arrangement of actions such that you're notified of all the events you want to hear about. See our configuration file for an example of how we've done it.

One important note: you may use `*` in place of a particular UPS name. However, if you do so with a timer, you may find that some events get masked by others. If you're going to use timers, you should set a specific timer for each individual UPS.

15.3.6 Enable NUT

Now that NUT is configured, you're ready to turn it on. Under some systems, you may need to edit a file so NUT knows to officially launch and take over power management for your system.

Under Debian, you must edit the `/etc/default/nut` file and enable the UPSD and UPSMON subsystems.

Once you've done this, you can start the daemon, and check your syslog file to see if it's started properly. You should see status messages from all your UPSes.

15.3.7 NUT CGI

NUT includes one final piece of functionality: the ability to report UPS status via a web interface. Most packaged versions of NUT include this functionality as a separate install, so it may not be included with the standard NUT daemon.

The CGI is simple to configure; enable it as you would any other CGI (see your web server documentation for more info), and then add the files `hosts.conf`, `upsstats.html`, `upsstats-single.html`, and `upsset.html` to your configuration directory (you may omit `upsset.html` if you do not wish to allow editing UPS configurations via the web).

The `hosts.conf` file simply lists the UPSes you wish to make available via the web interface.

The HTML files contain a template file that is used by the CGI to generate pages about all (or one) UPS(es). The default files show a reasonable amount of information, though you may wish to edit them to your taste (to display additional information, or to alter the formatting).

15.4 Compiling NUT

Precompiled versions of NUT exist for many free operating systems (such as Linux or the BSDs). If you use one of these operating systems, consult your package manager to see if a ready-made version is available.

We have compiled our own versions of the NUT software for the operating systems that we use at Suffield (see previous sections for downloads). Here is a brief guide for compiling from scratch, geared towards Mac OS X.

15.4.1 Getting the Sources

There are three major versions of NUT available as of this writing:

- The "old" 1.4 series.
- The "stable" 2.0 series.
- The "unstable" 2.1 series.

While the 1.4 and 2.0 series are protocol-compatible (*i.e.*, a 1.4 client can talk to a 2.0 server), the configuration files and structure are **not** compatible. Thus, you should try to stick with the same major release across platforms. We currently use the stable 2.0 series.

Download the latest stable client from the NUT downloads page:

http://www.networkupstools.org/source.html

Unpack the distribution on your machine, and move into the top-level directory.

15.4.2 Add a User

For security purposes, NUT prefers to run under its own username. You should create a local username on your machine using the standard account creation tools. The user should not have login access to the account, so disable logins via a bogus password or other account locks.

We use `nut_upsmon` as the username for our custom installations.

15.4.3 Build the Software

In the top-level configuration directory, run the `configure` script. You may wish to provide options to `configure` to customize its behavior; here are the options we use:

- `--prefix=/usr/local/nut_upsmon`, which locates all the installed software to a single directory on the filesystem (makes for easier packaging).
- `--with-statepath=/usr/local/nut_upsmon/state`, which tells NUT to store driver information in the local software tree. Defaults to `/var/state`, but we move it to make it easier to track.
- `--with-user=nut_upsmon`, which is the user we created in the previous step.

- `--with-drivers=apcsmart`, which reduces the drivers we use. In reality, the clients probably won't need **any** drivers, as they'll be querying our master server. We throw the drivers in "for free" in case we need them (we only use APC model UPSes, so this is an easy choice). You should include only the UPSes that your site uses.

Once the configure step is done, run `make` to compile the software.

When `make` is finished, you may install the software directly by running `make install`, or you may divert the install to another directory for packaging. Since we build packages, we opt for the latter:

```
make DESTDIR=/tmp/nut_upsmon install
```

15.4.4 Configure the Software

You'll need to add configuration files to `/usr/local/nut_upsmon/etc/` for the clients. Because we run a centralized server, the clients should not need any configuration files dealing with the "server" parts of NUT. At a minimum, you'll need an `upsmon.conf` file (specifying a remote UPS to monitor), and you may optionally want an `upssched.conf` file if you wish to perform additional tasks (such as early shutdowns).

Our setup involves three custom files, which you may download below:

- `upsmon.conf` (download), which specifies a remote server to query for UPS state information.
- `upssched.conf` (download), which specifies a special script (`early-shutdown`; see below) to execute when a UPS has been on battery for a certain amount of time. This allows certain machines to be shutdown before the battery goes critical.
- `early-shutdown` (download), which is a script that receives events from `upssched` and shuts the machine down early.

15.4.5 Launch the Software

The software should be configured to run on boot. For UNIX-based systems, this probably involves editing the init scripts on the platform.

For Mac OS X, we prefer to use `launchd`, which is Apple's new way of automatically running software starting with 10.4. Unfortunately, `nut` doesn't play nice with `launchd` by default, as it detaches and forks from the controlling terminal.

We've written a simple wrapper script that sits between `launchd` and `upsmon` (the client part of `nut`). It's called `upsmon_launchd_wrapper`, and you can download it from our site. You'll also need the launchd plist file, which should be placed in the `/Library/LaunchDaemons` folder and activated.

15.5 Replacing Batteries

15.5.1 When to Replace?

Ordinarily, the UPS will let you know when it thinks it's time to replace its battery. Most UPSes self-test on a regular basis, and if one of these tests shows that the battery isn't holding a proper charge, a REPLBATT message will be issued by NUT. **Do not ignore these messages!** Batteries that don't hold a charge may provide little (or no) runtime.

15.5.2 Replacing the Battery

Order a replacement battery for the UPS. In the case of APC Smart-UPSes, the batteries can be swapped while the UPS is powered on. Simply unscrew the bezel, pull the large power connectors, and remove the old battery. Replace with the new battery and reverse the steps to close the UPS up.

15.5.3 Updating the Battery Date Variables

APC UPSes have internal variables that allow you to keep track of when the battery was installed. Unfortunately, these variables are not set automatically, so you'll need to freshen them up when you add the new battery.

To do this, run the `upsrw` command, and provide the date (you'll need the master user password):

```
upsrw -s "battery.date=2007Apr1" -u upsmaster "myups@localhost"
```

We use a textual month to prevent ambiguity in date format.

15.5.4 Refreshing UPS Battery Data

Once the battery is installed, you'll need to let the UPS know that there's a new battery. **APC recommends waiting at least 8 hours for the battery**

to get a full charge before you re-run the self test. Once 8 hours have elapsed, issue a self-test and the "replace battery" light should go out.

Unfortunately, this process isn't perfect, and sometimes the UPS will still think it needs a new battery. This can manifest in one of two ways:

1. The UPS continues to fire a REPLBATT event (the red light on the front panel remains lit)
2. The UPS reports a "low battery" condition, even though it is on line power and not over load. You will note that the estimated runtime is lower than it should be (possibly only 1-4 minutes).

If this happens, try the following until the situation is resolved:

The Easy Way

Issue a self-test on the UPS, by holding in the test button (or starting the test from software). The "replace battery" light should go out after the test.

The Hard Way

If the self-test doesn't seem to "take", you can perform a runtime calibration. You can start this by issuing an ups command:

```
upscmd -u upsmaster -p <password> myups@localhost calibrate.start
```

The calibration process must be started with the ups at a 100% charge! The calibration also must have a load attached (if you're nervous about production machines being attached, find a "dummy" load to run.

The process puts the UPS on battery, and drains the battery to 25% before automatically switching to line power. The UPS uses the load data and time to compute the new runtime estimate.

The Really Hard Way

If a software calibration doesn't help, then a forced calibration may be necessary. The following are taken from the NUT FAQ (http://www.networkupstools.org/faq/) and also from phone instructions given by an APC technition.

1. Disconnect any serial monitoring cables (we don't want the server interfering with the UPS)

2. Hook up a dummy load (we're going to drain the UPS completely, so don't use any production machines). Ideally, the load should be about 30% of the UPS capacity (2 out of 5 LEDs lit on the front).

3. With the load attached and stable, pull the plug. The NUT FAQ says that the UPS must remain grounded, but APC didn't mention this. If you want to stay grounded, keep the UPS plugged in, but attach it to a device that can cut the circuit without disconnecting from ground (a surge protector, perhaps).

4. Kick back and wait for the UPS to drain completely. It will beep a lot; just go do something else for a while and let it drain.

5. Once the UPS has drained, plug it back in and turn it back on. The UPS should now have re-calibrated itself with the runtime data.

Chapter 16

Incremental Backup Script

Last updated 2008/03/18

16.1 Introduction

Suffield Academy provides networked disk space for all of its users, and encourages its use as part of a regular backup strategy. We back up the entire disk array nightly to another machine as part of our disaster recovery plan.

Unfortunately, we do not have enough space to archive each of these full nightly backups. So, while we are protected against the server crashing, we do not have the ability to recover files from a particular point in time.

To remedy this problem, we designed a custom backup script to satisfy the following criteria:

- Perform "snapshot" backups that capture the entire file structure at that point in time.
- Only store changed data for each backup.
- Allow for exclusion of particular files (temporary, cache, trash, *etc.*).
- Allow for exclusion of directories that are over a certain allowed size (*e.g.*, 1 GB).
- Report excluded directories to the users that own the files.

We found that the best way to accomplish this was to use a collection of scripts to wrap existing utilities and provide the functionality we needed.

We use `rsync` as the tool to perform the backups, and it is the workhorse of our system. Rsync contains configurable settings to allow the exclusion or inclusion of particular files, and we can script the automatic exclusion of users who are over a particular quota.

Meanwhile, Rsync also includes the ability to hard-link against existing files (a method later used by Apple's Time Machine), so that you can store full file system copies, but only have to use disk space for the files that have changed. While not quite as good as copy-on-write filesystems that operate at the block level, this approach is very portable across operating systems.

Finally, we wrap Rsync in custom scripts that archive completed backups, timestamp them, and weed out older backups to conserve space.

The final collection of scripts are written in Perl and Bash, with very few module dependencies. The scripts include full documentation and are configurable.

16.2 Design Issues

(This section deals with the design considerations for the script. If you just want to start using the script, skip down to the usage section.)

While searching for methods to create *snapshot backups*, we found an excellent strategy for backing up only incremental changes. It involves using *hard links* on the filesystem to store redundant (*e.g.*, unchanged) files, and rsync to transfer only files that change between backup sessions. Please read the paper for more information; a discussion of the technique is beyond the scope of this document.

We investigated existing solutions that use this strategy (including a program called rsnapshot, which looked very promising. Based on the features and existing code base, we decided to adopt this strategy for our backups.

We would have liked to use an existing solution for backing up, but encountered a few significant issues that could not be resolved without writing custom code:

- Preservation of HFS metadata, such as resource forks and creator codes. Currently, only Rsync version 3 and up support all of the necessary metadata preservation that we need.
- Tight integration with our directory service to exclude users who are over quota from backup, and notifying these users automatically via e-mail.

To attain these goals, we use Rsync 3 as the core of the backup system, and wrap it with shell scripts that provide the additional policies that we need to back up the correct information and retain it.

In the sections below, we discuss each of the issues in more detail.

16.2.1 HFS Metadata

Most Macintosh computers running OS X use the HFS+ filesystem for storing data. HFS+ has two features which frustrate file transfers, especially to non-Mac OS X systems: **file metadata** (especially **type and creator codes**), which have no counterpart on other file systems, and **forked files**, which split files into several unique parts. Because other filesystems do not use this approach, storing HFS+ data correctly becomes much more difficult.

There is a great utility called Backup Bouncer that automatically tests backup tools and their ability to preserve these sorts of data. Based on our testing, we discovered that a patched build of Rsync 3 provided the best retention of metadata while also supporting a very efficient transfer mode (differential copies with hard-links at the target).

16.2.2 Directory Services Integration

To save space in our backups, we needed the ability to exclude files from the backup based upon a user's quota status. We do not enforce a hard quota limit on our fileserver (to allow users to store large files temporarily), but we didn't want to waste space with large files that didn't need to be backed up.

When backing up user home directories, the script communicates with our Open Directory server to find users that are over quota. If a user is over their quota, files are excluded from the backup (until the non-excluded files fit within their quota). When a user's files are excluded, their e-mail address is queried and the user is notified that certain files were not backed up.

16.2.3 Remote Backups

Rsync can perform backups via the network, and we have designed our scripts to allow this behavior as well.

Because our script performs various housekeeping duties (rotating directories, locating old directories to link against, *etc.*), remote backups must adhere to a specific way of doing things in order to work properly.

We do this by using Rsync's daemon mode, and using a pre/post execution script. The script is automatically run by rsync before and after the transfer, and it takes care of creating and renaming directories, ensuring that transfer options are set correctly, and other housekeeping. Because it's run on the server

side, the sender (client) simply uses regular rsync without needing to worry about the policies on the server.

16.3 Script Usage

The scripts depend on the following packages to work correctly:

rsync 3.0+ The rsync binary must be installed on the system somewhere. The name does not have to be "rsync"; you may simply change the name used in the configuration file. Note that if you are going to support Mac OS X extended attributes (metadata), you'll need to use a patched version of rsync.

To download and build your own version of rsync, follow these directions (many thanks to Mike Bombich and his tips posted at http://www.bombich.com/mactips/rsync.htm

First, download and unpack the sources (substitute the current version of rsync instead of "3.0.6"):

```
wget 'http://rsync.samba.org/ftp/rsync/rsync-3.0.6.tar.gz'
wget 'http://rsync.samba.org/ftp/rsync/rsync-patches-3.0.6.tar.gz'
tar -zxf rsync-3.0.6.tar.gz
tar -zxf rsync-patches-3.0.6.tar.gz
```

The final step above merges the patches into the main source tree.

Next, move into the source tree and apply the Mac OS X metadata patches:

```
cd rsync-3.0.6
patch -p1 <patches/fileflags.diff
patch -p1 <patches/crtimes.diff
```

Now configure the sources:

```
./prepare-source
./configure
```

And finally, build and install:

```
make
sudo make install
```

You should be all set with a patched version of rsync (installed to `/usr/local/bin` by default). Run `rsync --version` to confirm the version and patches.

Unix::Syslog.pm This is a reasonably standard perl module which allows the scripts to record debugging information directly to syslog on your system.

Rsync is run in "daemon mode" on the server you wish to use as the destination for backups. The daemon is normally started as root (so that it can bind to a privileged port), and then an alternate user is specified to own the tranfered files.

You can use the standard processes for starting a daemon on your system (init, daemontools, rc, etc). If you use Mac OS X as your server, we have Launchd plist files available for starting and running rsync. There are two files:

1. rsync-daemon.plist
2. rsync-watcher.plist

The first launches the rsync daemon, but only if the path where the backups go is present (we use an external disk for our backups, so we don't want the daemon to run if the disk isn't mounted).

The second watches the backup path and kills rsync if it is no longer available (the disk is unmounted). You'll need to customize the paths if you plan to use this functionality. If you don't need to worry about the path unmounting, you can just use the first script to keep the daemon alive.

Finally, Mac OS X users should ensure that the destination directory for their backups has **Ignore Ownership and Permissions** turned **OFF**. You can check this by choosing "Get Info..." on the destination volume. If privileges are not preserved, then rsync will assume that all the files have changed ownership (since, as far as it knows, they have), and every file will be retransmitted, making the backup non-incremental.

16.3.1 Organization

There are two main scripts to use as part of this system: one for the client (sender) and one for the server (receiver). Others are available for special tasks. Each script includes documentation in comment form, but we also have a brief introduction below:

`rsyncd_prepost` **(Download)** This file is executed by the rsync daemon on the server before and after every transfer. It also can be run from the command line to create the basic directory structure for a backup set.

`rsync_snapshot` **(Download)** This is a sample script to use on a client machine to send a backup to the server. It basically gathers up the username,

password, and paths, and then calls rsync. You may need to customize it for your client machines (for example, removing unused rsync flags, or adding exclude rules).

`users_over_quota` **(Download)** We use this script to total up the home directory sizes on our file server and write rsync excludes that eliminate large files from the backup. This script is custom to our environment, and you don't need it as part of the backups. We include it here so you can see how you might customize a backup using a pre-processing script that writes out a list of files to exclude from the backup.

16.3.2 Configuration Files

On the client side, there are no configuration files. All configuration is done directly when you invoke `rsync`, so you should modify the script that uses rsync to have whatever options/includes/excludes that you want.

On the server side, the `rsyncd_prepost` script shares the same configuration file as the `rsync` daemon (typically, `/etc/rsyncd.conf`). Our script parses the `rsyncd.conf` file to get the same options that rsync uses. Additionally, you can specify settings for the `rsyncd_prepost` script by placing special comment lines in the configuration file that our script understands (see below for an example).

Examples

We've included a sample `rsyncd.conf` file below. This file defines a single transfer target called "test". We've included detailed comments for each option so you know why the values are specified in this particular format.

Note: the rsync daemon re-reads its configuration file for every connection, so it is not necessary to signal (*e.g.*, HUP) the daemon if you change the config.

```
# /etc/rsyncd.conf


# This option forces rsync to write its PID out to a file.  Our
# launchd watcher script uses this PID to send signals to rsync, so
# this path should match that used by launchd.
pid file = /var/run/rsyncd_snapshot.pid

# Ordinarily, the user that starts the daemon process also owns all
# the files from the transfer.  We choose a non-root user here to own
# all the files.  The prepost script also uses these values when
# creating and setting ownership on directories.
#
# See also "fake super" and "incoming chmod" below
```

```
uid = rsync_user
gid = rsync_user

# Additional safety measure: change to the destination root of the
# transfer before executing any operations.
use chroot = yes

# use syslog instead of plain file logging
syslog facility = ftp

# enable plain file logging for more detail and debugging
#log file = /tmp/rsyncd.log
#transfer logging = yes
#log format = "%m:%u %h %o %f (%b/%l)"

# The following file contains the usernames and passwords for
# connections to the daemon.  File format is username:password, one per
# line.  We use this to restrict transfer modules to specific users.
secrets file = /etc/rsyncd.secrets

# Normally, to preserve ownership and permissions, rsync must run as
# root.  However, by using "fake super", rsync stuffs all file
# metadata into xattrs on the files and lets them be owned by a
# non-root user.  Additionally, this allows you to store metadata
# not supported by the native file system.
fake super = yes

# When using fake super, you are not running the transfer daemon as
# root.  This means that certain operations on the destination files
# can fail (such as setting attributes) if the permissions on the
# files are not permissive enough.  For example: if a file does not
# have read permission for "user", when it is transferred to the
# destination the rsync daemon user (who is not root) cannot read its
# xattrs to determine if the file changed, resulting in an error.
#
# To work around this, we set a user-permissive umask (or chmod) on
# the receiving side to guarantee that the rsync daemon can at the
# very least read and write to files it owns.  Be aware that when you
# restore files back to a host, the permissions may be (slightly) more
# permissive, though it's rare that you actually have a file that the
# owner cannot read or write to...
incoming chmod = u+rwX

# We specify our pre/post script for all transfers, using this file
# (rsyncd.conf) as the first argument so it can find all configuration
# options.
pre-xfer exec = /usr/local/bin/rsyncd_prepost /etc/rsyncd.conf
post-xfer exec = /usr/local/bin/rsyncd_prepost /etc/rsyncd.conf

# to prevent multiple connections overwriting each other, only allow
# one connection at a time (note that you must specify a lock file for
# each module, as the connection limit is enforced on a per-lock-file basis).
max connections = 1

# allow shares to be writeable (otherwise it's hard to back up to them!)
read only = no
```

```
# by default, don't advertise module names
list = no

# Comments that begin with "rsyncd_prepost" are read by the prepost
# script and used to set values in that script.  Because they are
# comments, they are ignored by the rsync daemon.

# date format for snapshot directory names (per strftime())
# rsyncd_prepost: dateformat=%Y-%m-%d

# The pre/post script renames backups by the date and time they
# complete.  Since backups take a variable amount of time to finish,
# it can be helpful to round off the time to the nearest hour/day/etc.
# Specify a time in seconds that you would like to round do (we
# use 1 day, or 86400 seconds).
# rsyncd_prepost: dateround=86400

# This is a module named "test".  In general, you want one module per
# set of files you're going to back up.  Most of our modules are for a
# single server, though some larger servers use multiple modules to
# spread out the organization of the files.
[test]
        comment = This is the comment

        # The path MUST always end in "rsync", and the level above
        # that should match the name of the module.  The pre/post
        # script will create the module dir and any necessary subdirs
        # if you run the prepost script with the module name as the
        # last argument.
        path = /Volumes/encrypted/test/rsync

        # To limit the number of connections to a particular module,
        # you must specify a lock file that is unique to that module.
        # Otherwise, the shared connection limit is global (for all
        # modules) and you'll likely get conflicts.
        lock file = /var/run/rsync_snapshot_test.lock

        # List any users from /etc/rsyncd.secrets that should have
        # access to this module
        auth users = test

        # List any machines that should have access to this module
        # (typically, only the machines that are sending the backups)
        hosts allow = 192.0.2.1

        # If desired, you can specify per-module options for the
        # pre/post script here as well.  The lines below define how long
        # snapshots are kept, and how far apart in time they are spaced.
        # See the pre/post script for more details.
        # rsyncd_prepost: snapshotpreserve=60 30
        # rsyncd_prepost: snapshotpreserve=120 60
        # rsyncd_prepost: snapshotpreserve=300 -1
```

16.4 Recovery

To recover files from the backup server, there are two basic options.

The first is to create a new module specification that includes the path to the snapshot directory you wish to recover from. You may specify an additional username or host restriction if you're recovering from a different host. Be sure to OMIT the "pre-xfer" and "post-xfer" lines from the config so the server doesn't think it's a live backup request.

However, that approach is cumbersome, and requires editing and saving files based on parameters at the time of recovery. An easier approach is to tunnel the rsync request via SSH and specify the path to recover from on the fly.

To do this, you need to have SSH access to the server, and that SSH user must have (at least) read permissions to the module folder you wish to restore from.

You can then use rsync tunneled through SSH to connect to the machine and restore the files. Consider the following example (which should be typed all on one line, or use continuation backslashes as we have here):

```
sudo /usr/local/bin/rsync -aNHAXv -e ssh \
--rsync-path="/usr/local/bin/rsync --fake-super" \
ssh_user@backups.example.com:/path/to/module/snapshot/2009-10-11/ \
/tmp/restore_to_here/
```

The rsync options (`-aNHAX`) should match those used when you created the backup from the sender. The `-e ssh` tells rsync to use the SSH tunnel instead of a direct connection. If your server normally has the **fake super** option set in its `rsyncd.conf` file, you need to tell the tunneled rsync daemon to turn it on as well using the `--rsync-path` option as we have. Finally, you specify the full source and destination path. In the case of the most recent backup, it will live in `.../module/rsync/complete/`. Otherwise, you can specify the snapshot to restore from with `.../module/snapshot/(date)/` as we have above.

Chapter 17

Subversion

Last updated 2008/03/18

17.1 Introduction

Subversion is a source-control and revisioning repository. In short, it's a system that allows multiple people to edit files and keep up-to-date with each other's changes. Additionally, it stores all versions of a file as its being edited, so it's simple to restore previous versions of a file in the event of a problem.

At Suffield, we use Subversion to store configuration data, web sites, and documentation (including the document you're reading right now).

17.2 Mac OS X Client Setup

As Mac OS X is a UNIX-like operating system, the command-line parts of Subversion compile cleanly and are easily ported to the system. Several precompiled versions of Subversion are available for download on the web in a packaged form. Additionally, you may build Subversion yourself from source, either by hand or by using a port utility.

17.2.1 Precompiled Versions

Martin Ott (one of the developers of SubEthaEdit) has statically-compiled binaries of Subversion for Mac OS X:

http://homepage.mac.com/martinott/

Download, install, and you're ready to go! This solution is good for machine that only need Subversion, and not a whole package-management system.

17.2.2 Fink

Subversion is included in the Fink project. If you already have Fink installed on your machine, you can easily install it using the package management tools.

17.2.3 DarwinPorts

For machines that will have several open-source packages installed on them (in addition to Subversion), it's best to use a port-management system. Fink works well, but we are now tending towards the DarwinPorts system. This system is similar to Fink, except that you compile the software yourself. We choose it because the software tends to be a little more up-to-date than the precompiled versions in the Fink repositories.

To install the Subversion client, you must first install DarwinPorts. Visit the main site (http://www.darwinports.org/) and install the Mac OS X installer. (For more information on what DarwinPorts is and how it works, please refer to the documentation on the website.)

Note that DarwinPorts requires a working compiler and other build tools. You can get these by installing the **XCode** tools on your machine (the **XCode** tools can be found on your installation DVD).

If you've just installed DarwinPorts for the first time, you'll need to add a line to your `.bash_profile` that looks like this:

```
# Add DarwinPorts path
export PATH=${PATH}:/opt/local/bin
```

Once you've installed the software, you're ready to update the list of available ports. Type the following in the Terminal:

```
sudo port selfupdate
```

You are now ready to build Subversion. Run the following command:

```
sudo port install subversion
```

DarwinPorts will download, unpack, configure, compile, and clean the distribution for you automatically. When done, you should be able to run the following:

```
svn help
```

You should get a list of Subversion commands on your screen. If you do, then subversion is installed, and you're ready to go!

Chapter 18

PostgreSQL

Last updated 2008/03/18

18.1 Introduction

PostgreSQL (pronounced "post-gress-cue-ell", or "postgres" for short) is an open-source relational database engine. The database runs on Unix variants (Linux, BSD, Mac OS X), as well as Windows. Clients can connect to the database using native clients or JDBC. Interaction with the database is performed using SQL. For more information about the features of PostgreSQL, please visit the project's web page:

http://www.postgresql.org/

PostgreSQL is an alternative to other SQL-based database products, such as MySQL, Microsoft SQLServer, Oracle, and FileMaker (note, however, that FileMaker has a built-in GUI tool which most other databases do not have). In terms of free database engines, PostgreSQL "competes" most directly with MySQL. Suffield uses PostgreSQL because it is free, robust, and has some features that MySQL lacks. Additionally, while most MySQL software can be made to run under PostgreSQL, the reverse is not always true.

A full explanation of relational database systems is beyond the scope of this document; it is assumed that the reader is familiar with the basic concepts of relational databases, tables, columns, data types, and SQL queries.

This document describes the basic installation, setup, use, and maintenance of a PostgreSQL database server.

18.2 Installation

Suffield uses Mac OS X and Debian Linux as its primary server platforms. Please follow the instructions for the appropriate platform.

18.2.1 Debian Linux

Debian Linux supports PostgreSQL as a package in their standard distribution. Install the `postgresql` and `postgresql-client` packages for the latest stable version, or include a version number to get a specific revision:

```
apt-get install postgresql-8.2 postgresql-contrib-8.2
```

The packages will be downloaded and installed automatically.

Note that different versions of Debian support different versions of PostgreSQL in their package repositories. Also, depending on the relase dates of Debian and PostgreSQL, you may not be able to get the latest version in your distribution.

To get the latest version in Debian, you can use "apt-pinning" to draw in selected files from unstable distributions of Debian. To do this, add the "backports", or "unstable" (or possibly "experimental") releases of Debian to your `/etc/apt/sources.list`:

```
# only need one of these!
deb http://www.backports.org/debian etch-backports main contrib non-free
deb http://debian.lcs.mit.edu/debian/ unstable main
deb http://debian.lcs.mit.edu/debian/ experimental main
```

Then, set the pin-priority for these new releases in `/etc/apt/preferences` (create the file if it doesn't exist):

```
Package: *
Pin: release a=stable
Pin-Priority: 700

Package: *
Pin: release a=etch-backports
Pin-Priority: 650

Package: *
Pin: release a=unstable
Pin-Priority: 600

Package: *
Pin: release a=experimental
Pin-Priority: 500
```

Note that the "stable" distribution has the highest priority. This means that only packages that cannot be found in the stable distribution will be fetched from the unstable distribution.

Now you can run `apt-get update` and install the version of PostgreSQL you'd like.

18.2.2 Mac OS X

Note that PostgreSQL can be installed on a regular "client" build of Mac OS X; you do **not** need the "server" version of the OS.

Mac OS X supports several methods for installing PostgreSQL: via Fink, DarwinPorts, or PostgreSQL Tools for Mac OS X. The first two are packaged installers using a build or package framework (DarwinPorts is like "ports" for BSD, and Fink is like "apt-get" for Debian). The latter is a native Mac OS X package that installs independently of other software.

However, most of these packaging systems only install the base version of the server. We have since switched to compiling our own binary direct from source, as this gives us access to some necessary extensions for the software. Don't worry; this isn't difficult at all (the software builds cleanly on Mac OS X with no additional dependencies).

Downloading

To obtain PostgreSQL, just visit their website:

http://www.postgresql.org/ftp/source/

That's a direct link to their source section; you can also go to their main site and follow links to their download section.

Download the full source distribution (it's the one that's called `postgresql-X.Y.Z.tar.bz2`). The `base`, `docs`, `opt`, and `test` tarballs are all included in this file, so you don't need to download them separately.

Compiling and Installing PostgreSQL

You'll need to have a compiler available on the machine, which means installing **Apple's Developer Tools CD**. You can get it from:

http://connect.apple.com/

Once the developer tools are installed, you're ready to build PostgreSQL.

Unpack the tarball into a directory that's stable (we want to keep the compiled sources around for extensions, so don't put the sources anywhere that's temporary):

```
tar -jxf postgresql-X.Y.Z.tar.bz2

cd postgresql-X.Y.Z
```

Now it's time to run the `configure` script. Nearly all the options that PostgreSQL uses are supported out-of-the-box by Mac OS X, so we enable most of the options. We tell the configure script to place the resulting software in `/opt/postgresql-X.Y.Z`, though you may choose to put it almost anywhere (`/Applications`, `/Library/PostgreSQL`, *etc.*):

```
./configure --prefix=/opt/postgresql-X.Y.Z --with-perl --with-tcl \
            --with-python --with-krb5 --with-pam --with-ldap \
            --with-bonjour --with-openssl
```

Once the configure script has completed, you're ready to build the software. Run make:

```
make
```

And wait for the compilation to finish. Assuming there are no errors, you may now install the software:

```
sudo make install
```

You now have a base installation of PostgreSQL installed.

Compiling and Installing Extensions

In addition to the base install, we install a few extensions that are useful, or required by other software that uses PostgreSQL. These programs live in the `contrib` directory in the source folder, so we must move there first:

```
cd contrib
```

First, we build the `tsearch2` full-text search engine:

```
cd tsearch2
make
sudo make install
cd ..
```

We also use the xml2 extensions:

```
cd xml2
make
sudo make install
cd ..
```

The full suite of software is now installed.

Account Creation

You must create a `postgres` user account on your machine, if one does not exist already. You can do this using the usual account creation tools (**System Preferences** or **Workgroup Manager**), or you can do it from the command line using this script:

```
#!/bin/bash

NAME="postgres"
HOME="/opt/postgresql-X.Y.Z"

# Check to see if the user exists or not
nicl . -read /users/${NAME} >/dev/null 2>&1
if [ $? != 0 ]; then

    echo "User ${NAME} does not exist; creating..."

    # delete any pre-existing group with our name (shouldn't happen)
    nicl . -delete /groups/${NAME} >/dev/null 2>&1

    # Find the next shared UID/GID that's between 100 and 500
    lastid=`(nireport . /users uid ; nireport . /groups gid) | sort -n | uniq | egrep -v '^([0-9]{1,2}|[5-9][0-9]{2,}|[0
    id="`expr $lastid + 1`"

    # in the case that there is no ID over 100 to come after, just start at 200
    if [ ${id} -lt 100 ]; then
        id=200
    fi

    # create new group
    echo "Creating group ${NAME} with GID $id"
    nicl . -create /groups/${NAME}
    nicl . -createprop /groups/${NAME} gid ${id}
    nicl . -createprop /groups/${NAME} passwd '*'
    echo "niutil: group '${NAME}' added."

    echo "Creating user ${NAME} with UID ${id}"
    nicl . -create /users/${NAME}
    nicl . -createprop /users/${NAME} uid ${id}
    nicl . -createprop /users/${NAME} gid ${id}
    nicl . -createprop /users/${NAME} passwd '*'
    nicl . -createprop /users/${NAME} change 0
```

```
    nicl . -createprop /users/${NAME} expire 0
    nicl . -createprop /users/${NAME} realname "PostgreSQL Database"
    nicl . -createprop /users/${NAME} home "${HOME}"
    nicl . -createprop /users/${NAME} shell '/usr/bin/false'
    nicl . -createprop /users/${NAME} _writers_passwd "${NAME}"
    echo "niutil: user '${NAME}' added."
else
    echo "User '${NAME}' already exists; not creating it again"
fi
```

Note that the password is blanked out for security purposes. If you create the script using a GUI tool, you should disable the password.

Finally, you'll need to create two directories and own them to the `postgres` user:

```
mkdir /opt/postgresql-X.Y.Z/data
mkdir /opt/postgresql-X.Y.Z/log

chown -R ${NAME}:${GROUP} /opt/postgresql-X.Y.Z/data
chown -R ${NAME}:${GROUP} /opt/postgresql-X.Y.Z/log

chmod 700 /opt/postgresql-X.Y.Z/data
```

Cluster Creation

A new database cluster must be **initialized** before it can be used. This is done by running the `initdb` command as the `postgres` user:

```
sudo -u postgres /opt/postgresql-X.Y.Z/bin/initdb -D /opt/postgresql-X.Y.Z/data
```

The database is now initialized and ready to run.

PATH setup

By default, the server and client binaries are installed in `/opt/postgresql-X.Y.Z/bin`. To enable local machine users to use these binaries without having to specify the full path, you should add a stanza similar to the following to your `/etc/profile` (for system-wide effect), or to your `~/.profile` (single user):

```
if [ -d /opt/postgresql-X.Y.Z/bin ]; then
    PATH=${PATH}:/Library/PostgreSQL8/bin
    export PATH
fi
```

LaunchDaemon Startup

If you've installed on a machine running Mac OS X 10.4 or later, you may wish to use `launchd` to start and stop the database engine. To do so, follow these steps:

1. Copy our LaunchDaemon plist to the `/Library/LaunchDaemons/` directory on the database server.

2. Customize the plist file to include the correct path to the server (replace `X.Y.Z` with your actual version of PostgreSQL).

3. Start the server using `launchd`:

   ```
   sudo launchctl load -w /Library/LaunchDaemons/org.postgresql.postmaster.plist
   ```

4. Confirm that the server is running:

   ```
   ps auxwww | grep postgres
   ```

   You should see several entries for the database processes.

18.3 Configuration

Once the database is installed, you should take a few minutes to configure the database engine. Most of these configuration options have to do with network security and who is allowed to access the database.

Under Mac OS X, the configuration files live in the `data` folder, relative to where you installed the software:

`/opt/postgresql-X.Y.Z/data`

You'll need to be a super-user or the user `postgres` in order to access this location on the hard drive. To become the `postgres` user, execute the following:

```
sudo -u postgres /bin/bash
```

That will give you a login shell as the `postgres` user. You can then enter the directory listed above.

18.3.1 Schemas

A PostgreSQL cluster may contain one or more **databases**. Each one of these databases is logically separated from the others; tables, functions, triggers, and other objects in a database cannot affect others. This is desirable because it allows different applications to have their own databases without conflicting with each other.

However, this also prevents sharing information between two databases. At Suffield, we have several different applications that rely on a database, and sometimes we'd like to tie these data together. For example, we might have a database with user information, and another database with phone accounting records. Ordinarily, the applications that use these databases know nothing about each other. However, we might wish to run a report that correlates phone usage with user names.

In earlier versions of PostgreSQL, the only way to share data was to assign all tables to the same database. This was problematic, because different applications might each want a table with the same name (*e.g.*, "users").

With recent versions of PostgreSQL, we can use the *schema* feature:

http://www.postgresql.org/docs/current/static/ddl-schemas.html

Schemas allow multiple applications to share the same database, but each have their own namespace for tables and other objects. This way, applications can operate on their own data (just like they would in separate databases), but also cross to other schemas to get data from other applications.

To create a schema, just issue the following:

```
CREATE SCHEMA myschema AUTHORIZATION myuser;
```

That creates a schema and assigns all rights to the given user (you may omit the AUTHORIZATION portion if you don't want to assign the privs to another user).

Now you can create tables (or other objects) within the schema:

```
CREATE TABLE myschema.foo(...);
```

Note the fully-qualified name. If you don't want to specify that for every table, you may set the default schema for a user by executing:

```
ALTER USER myuser SET search_path TO myschema;
```

18.3.2 Tablespaces

PostgreSQL allows you to store different parts of the database on different logical volumes:

http://www.postgresql.org/docs/current/static/manage-ag-tablespaces.html

For systems with multiple disks, this allows you to spread the load to different disks, or to assign different tables to filesystems with different levels of performance, reliability, or size.

To create a new tablespace, make a directory on a logical filesystem and own it to the PostgreSQL database user. Then, issue the following command in the PostgreSQL shell:

```
CREATE TABLESPACE spacename LOCATION '/path/to/spacename/dir';
```

Now, when you create a table in PostgreSQL, supply the tablespace as the final argument. For example:

```
CREATE TABLE foo(bar VARCHAR) TABLESPACE spacename;
```

If you'd like to permanently set the default tablespace for a particular user (so you don't need to specify it for each table), you can issue the following:

```
ALTER USER myuser SET default_tablespace TO spacename;
```

This may be helpful when you wish to set a user to always default to a given tablespace.

18.3.3 Host-based Authentication

Once remote access has been turned on, we PostgreSQL must be told which machines and users may connect to which databases. This is done by editing the `pg_hba.conf` file.

Generally, you should only allow connections from known, trusted machines. Additionally, you should only allow connections from given usernames to specific databases whenever possible. We make an exception for "localhost" so that users can connect directly on the database machine itself.

Below is our sample `pg_hba.conf` file:

```
# Database administrative login by UNIX sockets
```

```
local   all         postgres                          ident sameuser

# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD

# "local" is for Unix domain socket connections only
local   all         all                               ident sameuser
# IPv4 local connections:
host    all         all         127.0.0.1/32          md5
# IPv6 local connections:
host    all         all         ::1/128               md5

# Allow "testuser" to connect from a particular machine
host  suffield  testuser                 172.30.0.40/32  md5

# Reject all others
host    all     all       0.0.0.0      0.0.0.0          reject
```

Here, we allow users on the machine to connect directly without a password, but they cannot specify a different name.

Users can connect to the local loopback interface and specify a username and password (md5 encrypted). We also include IPv6 addresses for completeness.

We then have a list of trusted hosts (well, just one) that allows a particular user access to a particular database from a particular machine. Note that if you use schemas (see above), the database name will most likely be the same for all users.

Finally, we round out the file with an explicit "reject" statement to prevent all other connections.

18.3.4 Allowing Remote Access

By default, the database only listens on the local loopback interface for network connections. This is fine if all your database applications are hosted on the same server (*e.g.*, you're running the database on the same machine as your web server, and the web server is the only client of the database). However, if you need to allow remote machines access to the database, you'll need to change this behavior.

Edit the configuration file `postgresql.conf`. Find the variable `listen_addresses` and set it to `*`:

```
listen_addresses = '*'
```

18.3.5 Autovacuum

From time to time, a PostgreSQL database must be **vacuumed**. This process reclaims lost space in the database, prevents transaction ID wraparound, and generally keeps the database in good working order.

Starting in version 8.1, you can enable an "autovacuum" process that runs concurrently with the database engine. This prevents the need for a manual vacuum by the database administrator.

In the `postgresql.conf` file, change the following lines:

```
stats_start_collector = on
stats_row_level = on

autovacuum = on
```

You may tweak the other settings as you see fit; we currently use the default timeouts and load factors for autovacuuming.

18.3.6 Logging

PostgreSQL can log to STDERR, Syslog, or to a file. To prevent logs from filling up our system logs, we redirect all log output to files.

In the `postgresql.conf` file, change the following lines:

```
log_destination = 'stderr'
redirect_stderr = on
log_directory = '../log'
log_filename = 'postgresql-%Y-%m-%d_%H-%M-%S.log'
log_rotation_age = 86400
client_min_messages = notice
log_min_messages = notice
log_min_error_statement = error
log_line_prefix = '%t %d %u@%h'
```

18.3.7 Tweaking Kernel Memory Parameters

PostgreSQL works quite well "out of the box" but can benefit from some additional tuning. For high database loads, tuning becomes necessary to keep the database running smoothly.

There are many things to tweak in PostgreSQL; an annotated guide to some of these settings are available at the following links:

http://www.varlena.com/GeneralBits/Tidbits/annotated_conf_e.html

http://www.powerpostgresql.com/PerfList/

PostgreSQL benefits from larger amounts of memory being made available to the database engine. Specifically, you may wish to turn up the amount of shared memory available. On Mac OS X, the default amount of shared memory is 4MB; we turn this up to 128MB.

Changing the kernel parameters requires use of the `sysctl` command. To make the changes permanent, you can create (or edit) the `/etc/sysctl.conf` file.

Mac OS X

Under Mac OS X (10.3.9 or later), you should add the following lines to `/etc/sysctl.conf` (the example below assumes 128MB of shared memory):

```
kern.sysv.shmmax=134217728
kern.sysv.shmmin=1
kern.sysv.shmmni=32
kern.sysv.shmseg=8
kern.sysv.shmall=32768
```

Note that you need to add **all five values**, even if you're using the defaults for some of them (in the example above, we are only using non-default values for `shmmax` and `shmall`). Mac OS X triggers on the fifth value to set up the shared memory system, and the changes become permanent (until reboot) after that point. If you don't set all five, the change doesn't take, and will be overwritten by the Mac OS X defaults.

Linux

Under Linux, you should add the following lines to `/etc/sysctl.conf` (the example below assumes 128MB of shared memory):

```
kernel.shmall=134217728
kernel.shmmax=134217728
```

18.3.8 PostgreSQL Shared Memory Parameters

Once you've set the kernel's memory parameters (and rebooted if necessary), you can configure PostgreSQL to take advantage of this memory.

Using our system shared memory amout of 128MB, here are some suggested tweaks to `postgresql.conf`. Note that the settings shown are for PostgreSQL 8.2; if you use an earlier version you may not be able to use the "MB" and "kB"

suffixes for values, and must instead convert them to "pages". See the comments of the file for page size information.

Allocate all but 32MB of the shared memory to the PostgreSQL shared buffer cache:

```
shared_buffers = 96MB
```

This gives a chunk of shared memory over to PostgreSQL to cache table entries for faster searching.

You may also allocate 128MB (assuming your system has enough free memory) to the "maintenance_work" memory:

```
maintenance_work_mem = 128MB
```

This setting is used for VACUM, ANALYZE, and CREATE INDEX statements, and so can speed up those operations.

18.3.9 Speeding Up Disk Access

PostgreSQL uses a write-ahead log (WAL) to commit its changes. To ensure consistency, the database engine writes these log files just before comitting a transaction, and then flushes the log to disk using `fsync`.

While this process preserves data consistency (a good thing), it does slow the database down, as the logs must be flushed.

To speed up this process, there are a few steps we can take to speed up database operations. **Note:** some of these operations are **dangerous** in the sense that they can cause the database to possibly lose data.

- First, mount the `data` directory of the database on its own disk. Ideally, this should be a fast disk (RAID 1+0 is recommended for optimum performance). RAID 0 is fast, but unable to withstand failures. At the same time, RAID 1 is fault-tolerant, but has slow write performance. The risk of doing this is directly linked to the stability of the disk.

  We used to have Apple Xserve hardware, which has a maximum of 3 SCSI drives. Therefore, RAID 1+0 is not possible. We opted for a RAID 1 setup with a write-back cache (and a read-ahead for reads, though this isn't as important). The write-back cache gives reasonable write performance and the RAID 1 gives excellent read performance. In the event of a power failure, we have a UPS and generator to prevent data loss in the cache.

We have since moved to a Linux-based Compaq box with 18 drives, organized into two logical arrays running RAID 1+0. This gives us maximum redundancy with increased speed.

- Next, consider mounting the `pg_xlog` subdirectory of the `data` directory on its own disk. This is the directory that contains the write-ahead logs, so making it independent from the regular database store helps speed access. Again, you must balance the reliability of the drive with the possible speed increases.

- Finally, you can tweak the `fsync` and `wal_sync_method` parameters. These affect how the database writes the WAL out to disk. Note that turning off `fsync` can speed up writes significantly, but can be **very dangerous**. If the WAL files become corrupted, the database may be unable to automatically recover in the event of a system crash or power failure.

18.4 PostgreSQL Usage

In general, interaction with PostgreSQL occurs in three major ways:

1. Server Management commands, used to create or delete entire databases. Creation of users and their permissions is also frequently done through external commands. Finally, full backups and restores using human-readable file formats are performed using external utilities.

2. Manual database interaction, using commands entered at a command prompt via a special database shell. Most often used to test queries, or perform small maintenance tasks.

3. Programatic access using a language that supports direct communication with the database. Usually, these are SQL statements sent from the program to the database, with the program receiving and interpreting the results. Examples of languages that can communicate with PostgreSQL include (but are not limited to) Perl, Java, PHP, and C.

18.4.1 Server Management

Most server management is performed using special binaries that are included with the PostgreSQL distribution. Note that the binaries may reside on a special path on your system (especially under Mac OS X), so if you don't seem to have the binaries installed, be sure to search in the same directory tree as the database itself.

All commands have well-documented manual pages. Copies of the documentation are also available on the PostgreSQL web site.

:`createuser` / `dropuser`

Creates a database user credential. These credentials are frequently used for remote access to a database, and usually include a username/password pair.

The command takes several options, which describe the capabilities of the user. Read the documentation for the command carefully, and only allow the minimum privileges necessary for the user.

:`createdb` / `dropdb`

Creates or destroys a database. A *database* is a related group of tables that are accessed under a common name. It is possible (and highly likely) that you will have several databases run by a single instance of PostgreSQL.

This command supports a few options, including the `--owner` flag, which lets you specify a user who "owns" (has full administrative control over) the database. Usernames must be created using the `createuser` command (described above).

:`pg_dump`/`pg_dumpall`/`pg_restore`

Commands to back up and restore a database, or entire cluster of databases. `pg_dump` only dumps a single database, while `pg_dumpall` saves the contents of all the databases in the cluser. `pg_restore` will restore a database from the output produced from dumping the database.

The dumpfiles used by these commands are in plain text (or may be compressed using `gzip`. As such, they are highly portable, but not terribly space-efficient. Also, they are not easily used for incremental or differential backups (see later sections of this document for alternatives). As such, these utilities should be used only for periodic or transition-based full backups.

18.4.2 Manual Interaction

The command `psql` launches a database shell, which allows the user to send SQL statements directly to the database. Additionally, a small subset of PostgreSQL-specific commands are supported (mostly for querying the schema and data definitions).

To launch `psql`, you normally include a database name to connect to, a username to connect with, and a hostname to connect to (though you may omit any of these options if the defaults suffice). Thus, a typical invocation looks like:

```
psql -d mydatabase -U myusername -h localhost
```

Once connected, you can enter SQL statements directly at the prompt, terminated by a semicolon. The statements will execute, and any results (or errors) will be returned directly to the screen. On recent versions, large result sets will be piped to a paging program (such as `less`) for easy viewing.

To disconnect from the shell, type `^D` (control-d).

18.4.3 Programatic Access

Programatic access usually requires a library call from within the application. Generally, the application must connect to the database, issue SQL commands, retrieve the results of these commands, and then disconnect from the database. Each library handles this process differently, so a full treatment is beyond the scope of this document.

In general, you will need the following information in order to connect to PostgreSQL from an application:

- The name of the database
- A username to connect under
- The password that goes with the username specified
- The hostname to connect to

As a **very basic** example, here is a small Perl program that connects to a fictitious database and executes a small query:

```
use DBI;

# Connect to the DB
my $dbh = DBI->connect('DBI:Pg:dbname=mydatabase;host=127.0.0.1',
                       'myusername', 'mypassword',
                        { PrintError => 0, RaiseError => 0 } )
      or die ("Could not connect to database: " . $DBI::errstr);

# Prepare the query
$sth = $dbh->prepare("SELECT * FROM widgets")
      or die ("Could not prepare statement: " . $dbh->errstr);

# Execute the query
$sth->execute()
      or die ("Could not execute database query: " . $dbh->errstr);

# Fetch the results into a perl hashref
my $results = $sth->fetchall_arrayref({});

# Disconnect when done
$dbh->disconnect();
```

Other languages will use different syntax and library calls, but the general format remains similar to what we've described above. Consult your language's documentation for more information.

18.5 Backing Up

PostgreSQL contains utilities for backing up and restoring an entire database: `pg_dump`, `pg_dumpall`, and `pg_restore`. However, these utilities are focused on backing up entire databases in order to perform complete restores of a database. While these kinds of backups are helpful, they are time-consuming and quickly become out-of-date on a busy database.

PostgreSQL also supports a "hot" version of backup, by relying on its Write-Ahead Logging (WAL) capabilities. For consistency, PostgreSQL always writes any transaction to a special log file before changing the database. In the event of a system crash, this log file can be "replayed" so as to complete any in-progress transactions.

We can use the WAL as a form of incremental backup; if we take a **base** backup of the entire database and store every WAL file generated from that point onwards, we essentially have a collection of data that represents the entire database.

Because the WAL files are generated regularly, we can back each one up as it is created. Since the files are small, we can back each up as it is created, ensuring a near-exact copy of the database at all times.

Suffield uses a hybrid approach for database backups: we take regular full dumps of the database to ensure easy restoration. Additionally, we use the WAL to keep up-to-the-minute backups of the database to prevent data loss.

18.5.1 Suffield Overview

Here at Suffield, we have a single master PostgreSQL server. This server has a spare disk drive in it, which we use for "warm" backups of the database. Additionally, we spool these backups to a remote host on a daily basis for added peace of mind.

Here is a basic overview of how the backups work on our machine. The following process is executed once daily (usually late at night):

- We take a full dump backup of the database to our secondary disk on the database server. This file is also backed up to a remote host.

- We start a new PITR checkpoint backup, archiving the old PITR backup remotely.
- The database archives WAL files one at a time as they are rolled over. As part of the WAL archive, the files are backed up remotely at the same time.

18.5.2 The "postgresql_archive" Script

To support our PostgreSQL backups, we have written a Perl script called `postgresql_backup`. It takes care of all the different types of backups, and includes a mode that runs all the backups in the correct order. Once a day, we invoke this mode to perform the full database backups. The rest of the time, the script is called by the database to archive the WAL files.

The following section describes the setup and use of this script.

Preparation

The script requires a small amount of preparation in order to work properly.

1. First, you must set aside space on a local filesystem to store database backup files. Ideally, this should be on a separate disk from the database to prevent simple hardware failures from corrupting the backups. The space must be readable and writable by the user performing the backups (we suggest using the `postgres` user, as it has full access to the database).

   ```
   sudo mkdir /Volumes/Space/PostgreSQL-Backup
   sudo mkdir /Volumes/Space/PostgreSQL-Backup/dump
   sudo mkdir /Volumes/Space/PostgreSQL-Backup/pitr
   sudo touch /Volumes/Space/PostgreSQL-Backup/lockfile
   sudo chown -R postgres:postgres /Volumes/Space/PostgreSQL-Backup
   ```

   The lines above show the creation of the directory structure, along with a lockfile (necessary for the proper operation of the script). The final line owns the directories to the specified user so they can write to them.

2. Next, you must create space on your remote host to back up the files. We use our own `rsync_snapshot` scripts to perform the remote archiving, so our directory structure is set up to work with those scripts. If you use a different archiving scheme (for example, just a pure `rsync` call), you can use whatever paths work for you.

   If you're using SSH keys to allow the remote logins, you should create those keys at this time and install them properly on the client and server machines.

Our setup also includes `postgresql-*.conf` files that match our remote host and path information. These files are read by the `rsync_snapshot` scripts to perform the backups.

Configuration

Confirm that any external scripts or configuration files (such as those that go with `rsync`, or our `rsync_snapshot` scripts) are correctly installed and configured.

Execution

Copy the LaunchDaemon plist for PostgreSQL backups to the `/Library/LaunchDaemons/` directory on the database server. Once copied, schedule the job for execution using `launchd`:

```
sudo launchctl load -w /Library/LaunchDaemons/org.suffieldacademy.postgresql-backup.plist
```

The job should run at the time specified in the plist file. You can confirm that it is running by checking either its generated log file (logated in `$PGDIR/log`) or the system log (where the script logs by default).

18.6 Restoring From Backup

If the worst should ever happen, you'll need to restore your PostgreSQL database from your backups. Depending on the method(s) you use to back up the database, restoration will either be to the latest full SQL dump of the cluster, or a point-in-time recovery.

18.6.1 Point-In-Time Recovery (PITR)

PITR requires a raw backup of the database cluster files, along with a copy of the write-ahead log (WAL) files generated since the raw backup. **Note:** the WAL files are stored in an architecture-specific format (e.g., PowerPC, Intel x86, *etc*), so you **cannot** restore accross platforms using this method. If you have to move to a different machine, you must use a raw SQL dump (see below).

With PITR, it is possible to restore the database to a specific point in time (rather than just a full backup). For example, if you accidentally blew away an important section of the database, you could recover it up until (but not

including) the bad command. The instructions below do not cover this use of PITR, but the procedure is largely the same. Follow the instructions below, and when the time comes to create a `recovery.conf` file, add the options for the last transaction date you would like to restore (see the official documentation for more information).

The instructions in this section are adopted from the offical PostgreSQL documentation on the subject. For more information, please visit the official documentation site:

http://www.postgresql.org/docs/8.1/static/backup-online.html#BACKUP-PITR-RECOVERY

1. Stop the backup script and database, if they're running:

   ```
   sudo launchctl unload -w /Library/LaunchDaemons/org.suffieldacademy.postgresql-backup.plist
   sudo launchctl unload -w /Library/LaunchDaemons/org.postgresql.postmaster.plist
   ```

2. If you have space, move the current database cluster directory to a safe location (in case you need to get back to it):

   ```
   sudo mv /Library/PostgreSQL8/data /Library/PostgreSQL8/broken-data
   ```

   If you don't have room to keep the whole cluster, you should keep a copy of the `pg_xlog` subdirectory (in the `data` directory), as you may need the WAL files it contains

3. Restore the raw database files (the `data` directory), and ensure that they have the correct file permissions. You will need to copy the database files off of your backup server (or disk) using `rsync`, `scp`, `cp`, or another transfer tool. For example:

   ```
   sudo rsync -auvze ssh root@backup-server:/Snapshot/Backups/postgresql/data/ \
   /Library/PostgreSQL8/data/
   ```

   (The above line should be changed to use your actual host names and paths.)

   Once you have them on the server, own them to the database user:

   ```
   sudo chown -R postgres:postgres /Library/PostgreSQL8/data/
   ```

4. Next, you must create a `recovery.conf` file, which PostgreSQL interprets as a signal to recover from a previous backup. Normally, one must give a recovery command to find the archived WAL files. In our case, we back these files up with the base backup, and so they do not need to be "found" at all. Therefore, we specify a bogus recovery command. Place the following in the file `/Library/PostgreSQL8/data/recovery.conf`:

```
# use your path to the "false" command
restore_command = '/usr/bin/false'
```

Make the file writable by the database owner (the database renames the file when it's done with it):

```
sudo chown postgres /Library/PostgreSQL8/data/recovery.conf
```

1. Optionally, you may wish to modify the `pg_hba.conf` file to disallow logins to the database. This will prevent users from getting access to the data before the restore is complete.
2. Start PostgreSQL:

   ```
   sudo launchctl load -w /Library/LaunchDaemons/org.postgresql.postmaster.plist
   ```

   The postmaster will go into recovery mode and proceed to read through the archived WAL files it needs. Upon completion of the recovery process, the postmaster will rename recovery.conf to recovery.done (to prevent accidentally re-entering recovery mode in case of a crash later) and then commence normal database operations.
3. Inspect the contents of the database to ensure you have recovered to where you want to be. If not, return to the beginning of the recovery instructions and try again. If all is well, let in your users by restoring `pg_hba.conf` to normal.
4. Don't forget to start the backup process again if you stopped it:

   ```
   sudo launchctl load -w /Library/LaunchDaemons/org.suffieldacademy.postgresql-backup.plist
   ```

At this point, your database should be back up and running just as before. Double-check your log files to ensure that operation appears reliable.

18.6.2 SQL Dump Recovery

If you do not (or cannot) use PITR, you can restore the database from a pure SQL dump produced by the `pg_dumpall` command. This procedure is far simpler than PITR, in that you only need to issue a single command. However, there are a few drawbacks to keep in mind:

- The SQL dump is only current to the time it was taken. Depending on the time of the backup, the database may have changed significantly.
- You cannot recover a SQL dump to a particular point in time; you must restore the entire file or nothing at all.

- You cannot perform an incremental restore; the entire database must be emptied and then fully restored from the file.

A SQL dump does have the main advantage that it is portable accross platform architectures, and frequently portable accross versions of PostgreSQL. If you need to move a database from one machine to another, or between different versions of PostgreSQL, you should use the dump method.

Preparing for Restoration

As mentioned above, you cannot restore over an existing database. Therefore, you will need to destroy (or move) the current database before attempting recovery.

1. Stop the backup script and database, if they're running:

   ```
   sudo launchctl unload -w /Library/LaunchDaemons/org.suffieldacademy.postgresql-backup.plist
   sudo launchctl unload -w /Library/LaunchDaemons/org.postgresql.postmaster.plist
   ```

2. If you have space, move the current database cluster directory to a safe location (in case you need to get back to it):

   ```
   sudo mv /Library/PostgreSQL8/data /Library/PostgreSQL8/broken-data
   ```

   If you don't have room to back up the entire cluster, you **must** save any configuration files that contain non-default values. This includes (but is not limited to) `pg_hba.conf` and `postgresql.conf`.

1. Initialize a new database cluster directory:

   ```
   sudo -u postgres /Library/PostgreSQL8/bin/initdb -D /Library/PostgreSQL8/data
   ```

You now have an empty database cluster, and are ready for the next phase of the restore.

Restoring Configuration Files

To restore the cluster to a working state, you must first restore the configuration files from the old cluster directory. This includes files like `pg_hba.conf` and `postgresql.conf`, and may include others as well.

Before restoring, you may wish to limit access to the database by modifying `pg_hba.conf`. This will prevent users from interacting with the database while it is being restored.

Once you've got the configuration files in place, you should start the database engine:

```
sudo launchctl load -w /Library/LaunchDaemons/org.postgresql.postmaster.plist
```

Restoring the Cluster

This section is based heavily on the official PostgreSQL documentation. If you need more information, please consult the following URL:

http://www.postgresql.org/docs/8.1/static/backup.html#BACKUP-DUMP-RESTORE

Your configuration files should be restored at this point, and the database engine should be running. You're now ready to read in the SQL file containing all your data.

Note: loading large amounts of data can take a long time, especially if there are indices, foreign constraints, or other time-consuming consistency checks. You may wish to review the PostgreSQL documentation on large data loads; it discusses performance tips for loading large data sets. It specifically has this to say about using `pg_dump`:

> By default, pg_dump uses COPY, and when it is generating a complete schema-and-data dump, it is careful to load data before creating indexes and foreign keys. So in this case the first several guidelines are handled automatically. What is left for you to do is to set appropriate (i.e., larger than normal) values for maintenance_work_mem and checkpoint_segments before loading the dump script, and then to run ANALYZE afterwards.

For more information, please refer to the documentation:

http://www.postgresql.org/docs/8.1/static/populate.html

When you're ready to perform the load, you must run the restore command as the `postgres` user. You can do this using the `sudo` command:

```
sudo -u postgres psql -f database_backup_file.sql postgres
```

The last `postgres` in the database to initially connect to. Since the cluster is empty at this point, the only database you can connect to is the `postgres` database (which is created automatically as part of the `initdb` command).

The data load may take a long time, depending on the amount of data and the speed of your system. Wait for the load to fully complete before moving on to the next step.

Once the load is complete, you should log in to the database and make sure that the data appear to be restored correctly. At this point, the database is ready for use, though some additional steps should be taken to improve performance. You may allow clients back into the database at this time.

To help with query performance, you should perform a `ANALYZE` on each database to update all the housekeeping information in the database. You can do this with the following command *on every database in the cluster*:

```
sudo -u postgres psql -c 'ANALYZE' <dbname>
```

Substitute the datbase name for <`dbname`>, running the command as many times as is necessary for each database in the cluser.

Your database should now be restored, and ready to go!

Chapter 19

Host-based Firewalls with IPTables

Last updated 2008/03/18

19.1 Introduction

Most organizations have a firewall which prevents machines on the Internet (or other untrusted networks) from accessing protected machines. While these firewalls form an important part of a network's security, they can't do everything. Specifically, they don't guard against access from so-called "trusted" networks (behind the firewall), which is not always safe.

Therefore, host-based firewalls fill the gap in network security. Host-based firewalls allow us to tailor the types of connections we will accept from all hosts (regardless of their location).

This document describes how to protect Linux machines using **iptables**. iptables have been a standard feature of the Linux kernel since 2.4, and provide a robust means for protecting hosts.

This document is heavily geared to the Debian distribution of Linux, though the concepts should readily apply to other flavors of Linux.

Additionally, this document describes **host-based** Linux firewalls; that is, we only discuss firewalls that protect a single machine. If you want to create a Linux firewall to protect multiple machines (and possibly perform other tasks such as NAT), you'll need to look elsewhere.

19.2 Concepts

iptables has the notion of several **tables** of rules which are organized into **chains**. A packet moving through the filter system is assigned to the appropriate table for the operation being performed on it (filtering, NAT, or manipulation), and each table contains chains based upon where the packet is going (in, out, through, etc).

Each table contains a few built-in chains (such as `INPUT` and `OUTPUT`) where rules can be created. Additionally, the user can create their own chains, and reference them from the built-in chains. In this way, complex rulesets can be created and reused.

All of this makes for a very flexible system that allows complex rulesets to be created. Unfortunately, flexibility and brevity are often competing goals when designing firewall rules. For host-based firewalls, this can cause frustration when the underlying task seems so simple (for example, "deny all traffic except for these few ports that I specify").

19.2.1 The Suffield Firewall Scripts

Creating firewall rulesets can be time-consuming and difficult. For host-based firewalls, we often have a simplified subset of requirements; traffic is allowed or denied based on its source host or network and destination port and protocol.

We've designed a small set of scripts that will automatically generate rulesets from a short configuration file. Most of the common tasks for host-based firewalls (setting a default policy, creating user-defined chains, clearing unused rules) can be performed by the scripts, and the user can simply worry about creating a few rules that are uniuqe to the actual host.

We've designed the scripts to work closely with the Debian flavor of Linux. Debian has a framework in place to run scripts when the network state changes (an interface is brought up or down), and we exploit this framework to automatically apply firewall rules when interfaces are changed. If you don't use Debian, the scripts could easily be modified to work under a traditional init-script system (in fact, that's originally how they were authored).

Read on for specifics of how to use our scripts to quickly and easily create host-based firewall rulesets.

19.3 Installation of the Scripts

19.3.1 File Overview

Our firewall scripts are broken into several smaller scripts. The scripts and files are concentrated in two main directories:

- The `/etc/network/*` directories. Scripts in these directories are automatically invoked by Debian when a network interface changes state.
- The `/etc/default` directory, where configuration files are located. Files in this directory are unique to this host.

The scripts themselves are static, and should not normally require modification to run on a particular host. On the other hand, the configuration files are unique to a particular host, and must be customized for proper operation.

19.3.2 Installing the Scripts

The latest version of the scripts are located in our web repository:

Suffield Firewall Scripts

The files are laid out in a directory structure that mirrors that of the root Debian file system. Thus, the files in `etc/default` should be copied to `/etc/default/` on your host.

Copy each file to the correct relative location on the host you wish to protect.

19.3.3 Configuration

All configuration takes place on the files in the `/etc/default/` directory. The firewall configuration scripts all start with the word `firewall`. Three files are included with the distribution:

1. `firewall-default-init`: default rules to apply to all interfaces when the firewall is initialized. Any rules that should apply to the machine as a whole go here.
2. `firewall-default-kill`: any special rules to apply when an interface is brought down. By default, all rules for a given interface are cleared (after all, the interface is down at this point, so you can't even get to it). If you want to "leave behind" a rule, you should add it to this file.

3. `firewall-sampleiface-init`: for machines with multiple interfaces, you may wish to override the default rules with ones for a particular interface. Make a copy of this file, changing the word "sampleiface" to be the name of the interface the rules are for (*e.g.*, `eth1` or `eth0:0`).

19.3.4 Configuration Syntax

In reality, the configuration files are simply shell script fragements that get evaulated by the main firewall script. However, it's best to work with them by creating variables that the main script will use as input to its ruleset generators.

The variables used by the script are whitespace-separated values. Thus, you should take care to only include spaces between multiple values, and not to have any extra space.

Also, when appending a value to an existing rule, you should use the shell syntax for variable expansion. For example, to add a new rule on to the end of the `$TCP_ACCEPT` ruleset, you would do the following:

```
TCP_ACCEPT="$TCP_ACCEPT <new_rule_definition>"
```

Compare with the following:

```
TCP_ACCEPT="<new_rule_definition>"
```

Because no expansion of the previous rule is included, all previous rules get obliterated and replaced with the single rule on the right-hand side of the assignment operator.

19.3.5 Variable Types

In general, there are four classes of variables that affect the creation of firewall rules:

1. "Evil" addresses: defines rules for hosts that should not be allowed to communicate with the host **at all**.
2. ICMP rules: rules specific to accepting, rejecting, or denying ICMP packets.
3. TCP rules: rules specific to accepting, rejecting, or denying TCP packets.
4. UDP rules: rules specific to accepting, rejecting, or denying UDP packets.

Evil Addresses

The `EVIL_IPS` variable contains whitespace-separated values of hosts or networks that should not be allowed to communicate with this machine **at all**. Examples include non-routable, malformed, or private netblocks (the default config contains several of these built-in).

To append a netblock to the ruleset, use a line like this:

```
EVIL_IPS="$EVIL_IPS 169.254.0.0/16"
```

That line appends a rule to block the entire Class B subnet starting with 169.254 (the `/16` syntax is CIDR notation for a Class B subnet). To block a single host, use a CIDR mask of `/32`; otherwise, use a CIDR mask appropriate for the netblock size.

Because these addresses are non-routable, we advise keeping these rules in the "default" ruleset file, so they apply to all interfaces on the machine.

19.4 Protocol Rules

In addition to the Evil address rules, three other classes of variables exist to process the major protocols involved in IPv4 communications: ICMP, TCP, and UDP.

For each of these three protocols, three variables exist to define rules. You should place rules in the variable that corresponds to the action that should take place if the packet matches. The packets are matched agains the three actions in the order shown below:

1. `DROP`: defines rules for packets that should simply be discarded by the filter. No reply is sent to the host that sent the packets; thus, connection attempts that are dropped will cause the sending host to wait for some time before giving up on the connection. Usually, this is the best choice, as it reveals the smallest amount of information about your host. However, legitimate users will notice a long delay if they try to connect to a port they do not have access to.

2. `REJECT`: defines rules for packets that should not be accepted, but for which some reply should be sent to the host sending the packets. Sending hosts will see a "connection refused" or "port closed" message immediately, rather than waiting for the attempt to time out (as would happen with `DROP`). This is often used for ports like **ident**, which some machines will try to connect to as part of a legitimate query to the system. In these

cases, it's better to give an immediate rejection message, as the timeout associated with a `DROP` may cause a delay in other legitimate connections.

3. `ACCEPT`: defines rules for packets that should be accepted by the host, bypassing further inspection by the firewall.

Note that the firewall script includes a default action of `DROP` for any packets not explicitly matched by a rule. Therefore, you only need to create `DROP` rules when you wish to explicitly drop packets before processing other rules that would otherwise accept them. Additionally, you must create `REJECT` and `ACCEPT` rules for **all** packets that you do not wish to drop.

ICMP Rules

Three variables are available for ICMP processing: `ICMP_DROP`, `ICMP_REJECT`, and `ICMP_ACCEPT`.

The form of each rule is as follows:

```
ICMP_RULE="AAA.BBB.CCC.DDD/MM|icmp-type"
```

The first part is an IP address in dotted-quad notation, followed by a slash and the CIDR mask that should apply to the IP address. After the mask, a pipe symbol (|) and the ICMP type should be included.

As an example, suppose we wish to accept all ping requests from the 172.16.0.0/12 netblock:

```
ICMP_ACCEPT="$ICMP_ACCEPT 172.16.0.0/12|echo-request"
```

TCP Rules

Three variables are available for TCP processing: `TCP_DROP`, `TCP_REJECT`, and `TCP_ACCEPT`. **These rulesets only apply to initial TCP connection packets** (*e.g.*, those with the SYN flag set). By default, all "established" TCP connections are allowed, so we only allow the rules to affect new incoming connections. Thus, you don't need to worry about return traffic of outgoing TCP connections.

The form of each rule is as follows:

```
TCP_RULE="AAA.BBB.CCC.DDD/MM|src-port:src-port|dst-port:dst-port"
```

Again, the IP address and CIDR mask come first. Next, a range of source ports for the connection should be supplied (to allow connections originating from

any port, use `0:65535`). Finally, a range of destination ports (or a single port) should be provided. You may use names from `/etc/services` or numeric ports. The address and port specifications are separated by pipe characters (|).

For example, to allow any host to connect to the web server running on the firewalled machine, use a rule like this:

```
TCP_ACCEPT="$TCP_ACCEPT 0/0|0:65535|www"
```

The `0/0` means "any host", the `0:65535` means that the client may initiate the connection from any port, and the `www` means that the connection must be destined for the "www" port on the firewalled machine ("www" in `/etc/services` resolves to port 80).

UDP Rules

Three variables are available for UDP processing: `UDP_DROP`, `UDP_REJECT`, and `UDP_ACCEPT`. Note that UDP is a stateless protocol (it doesn't have the notion of a "connection"), so you may need to create rules to allow return traffic to get back to your machine.

The form of each rule is as follows:

```
UDP_RULE="AAA.BBB.CCC.DDD/MM|src-port:src-port|dst-port:dst-port"
```

Again, the IP address and CIDR mask come first. Next, a range of source ports for the connection should be supplied (to allow connections originating from any port, use `0:65535`). Finally, a range of destination ports (or a single port) should be provided. You may use names from `/etc/services` or numeric ports. The address and port specifications are separated by pipe characters (|).

For example, to allow any host to connect to the DNS server running on the firewalled machine, use a rule like this:

```
UDP_ACCEPT="$UDP_ACCEPT 0/0|1024:65535|domain"
```

The `0/0` means "any host", the `1024:65535` means that the client may initiate the connection from any *unprivileged* (>= 1024) port, and the `domain` means that the connection must be destined for the "domain" port on the firewalled machine ("domain" in `/etc/services` resolves to port 53).

19.5 Usage

Under Debian Linux, usage of the scripts is automatic; their placement in the various subdirectories of `/etc/network/` means that the system will automatically call the appropriate firewall scripts when interfaces are brought up and down. Under Debian, this means whenever `ifup`, `ifdown`, or `/etc/initi.d/network` are run.

There are four basic times that we should invoke the firewall scripts:

1. When we're bringing up the first network interface on the system
2. When we're bringing up any network interface
3. When we're shutting down any network interface
4. When we're shutting down the last network interface on the system

For the "any interface" cases, the scripts that are invoked simply look for rules specific to that interface. They clear out any existing rules for the interface, and (if defined) build new ones.

For the "first" and "last" cases, the script performs some additional housekeeping. Default policies are set for the input and output chains, and some "helper" rulesets (such as those for TCP state tracking) are created or destroyed.

To cover all these cases, we have four smaller scripts, which each invoke the larger `firewall-base` script. Each script corresponds to one of the cases listed above, and take the appropriate action (or passes the appropriate parameters to the main script).

In general, you should not need to invoke the scripts manually, though you can if something goes wrong with a particular script run. Note that the scripts assume that iptables is in a consistent state when they are run; care should be taken to always run the scripts in a full up/down cycle.

Chapter 20

Exile

Last updated 2008/03/18

20.1 Introduction

This document describes the motivation, design, configuration, and use of the **Exile** shaping application. In brief, the software is designed to fairly distribute internet bandwidth to clients on a network by penalizing ("exiling") those who use more than their fair share.

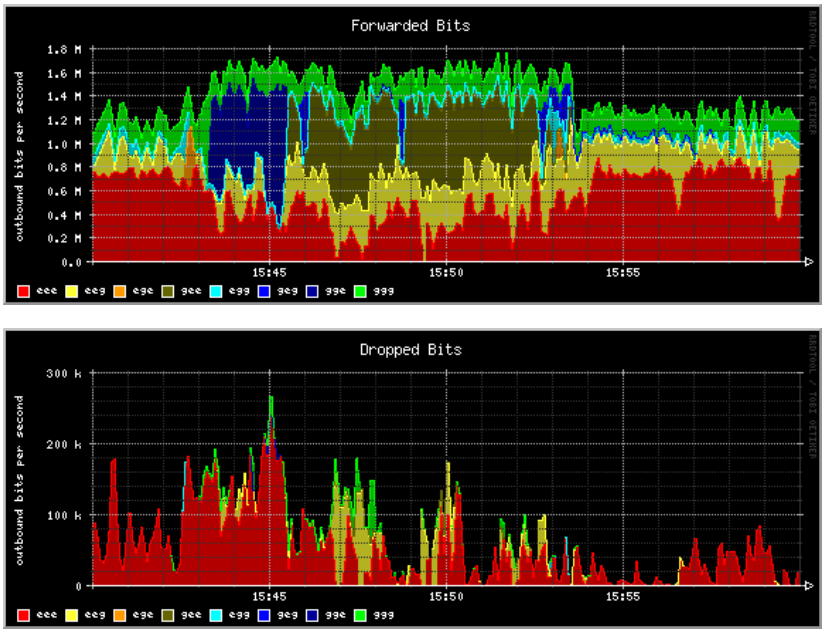
Note: this software is not a content- or filter-based solution, and so may not be appropriate for your needs. Please read the Design section of this document for more information.

In a hurry? Feel free to skip the History section of the document if you don't need to know why we started this project. However, please **do** read the Design portion of this document, as it contains important details about how the software works.

20.1.1 Synopsis

In short, the software works by keeping track of all the bandwidth used by every host (IP address) on your network. If a host crosses a threshold of traffic (which you set), they are "exiled" to a lower tier of service. This in turn allows hosts that haven't used up their "fair share" of bandwidth to have better access to the bandwidth that is available. There are multiple thresholds and tiers, so hosts may be exiled to different tiers depending on how much bandwidth they hog.

To help show this, consider the following graphs from our production system:



The colors range from green (no penalty) to red (maximum penalty). Note that at the beginning (15:40), much of the bandwidth is being used by exiled users. However, at 15:42 a new host starts using bandwidth. Because this new host doesn't have any penalties against it, it receives better access to the bandwidth.

You can see this because the amount of bandwidth given to the penalized users (in red) goes down to "make room" for the unpenalized user (in green). Additionally, looking at the "dropped bits" graph, you can see that the system favors dropping packets from the queues with larger penalties (red).

As time goes by, the new host begins to exceed the thresholds we've set, and they start to fall under increasingly severe penalties (first blue, then yellow). Thus, the system adapts over time to respond to users who hog bandwidth. Had this user continued to exceed the thresholds we had created, they would have eventually fallen into the maximum penalty queue (red) and stayed there until they stopped hogging bandwidth.

Again, the goal of the system is to try to make things "fair", in the sense that hosts who hog bandwidth should be penalized relative to other hosts who behave well. Please read on (especially the Design section) for more detailed information about how the system works, its strengths, and its limitations.

20.2 History

Like most residential schools, Suffield Academy suffers from a shortage of Internet bandwidth. Educational and recreational use of the network soak up bandwidth, and solutions that may work in the corporate world (strict filtering or blocking) don't work when users are in a residential setting.

Various solutions exist for prioritizing network traffic to improve (perceived) performance of the link. Suffield has tried several, with varied results:

PacketShaper and MasterShaper Works by analyzing traffic at all seven layers of the OSI model. Allows prioritization/limiting of traffic on a per-application, per-port, per-address, or per-session level (among others). Very fine-grained control, and the application-layer sniffing prevents some port-hopping by malicious software.

BandwidthArbitrator and NetEqualizer Works by analyzing traffic and "penalizing" bandwidth hogs. As the pipe fills up, penalties are cranked up in order to keep bandwidth utilization at a reasonable level. The basic version is GPL and freely available; the company also sells a ready-made appliance under the NetEqualizer brand.

QoS or Priority Queuing (various implementations) Many firewalls and routers now support at least a basic level of prioritization. While not as sophisticated as layer-7 shaping tools, it's cheap (included), and doesn't require extra hardware. Unfortunately, it only really helps with broad classifications (give entire IP blocks priority over others), as distinctions at the port or application level aren't always practical or possible.

Initially, we favored shaping solutions based on the inspection model. This gave us lots of control over "bad" applications (peer-to-peer, mostly), and kept the network running smoothly for "real work". Unfortunately, the "bad" software continues to try and evade detection through various means (port hopping, encryption, or impersonation). Effective filtering requires up-to-date signatures, and also a lot of horsepower to scan all the traffic.

In the case of free solutions (MasterShaper), keeping the filters up-to-date and tuned correctly proved difficult. In our initial tests, it failed to spot certain key types of traffic that we wanted to filter. On the commercial side, PacketShaper kept up reasonably well with new filters, but as our pipe to the internet got faster, the licensing costs soared.

We then evaluated the Bandwidth Arbitrator. It took a different approach to managing bandwidth. Rather than try to filter every possible use of the pipe, it worked to make sure that users weren't using more than their "fair share" of the bandwidth. The machine uses the Linux kernel (with modifications to the

bridging code) to track usage and apply "penalties" to connections that appear to use too much bandwidth. We tested the system out for a few weeks, with mixed success. We used the GPL version of the software, which came with no support and spotty documentation. The device did run, and it did limit overall bandwidth use. However, it failed to make a major improvement in overall latency, and it was somewhat frustrating to administer. Finally, the documentation was not clear enough on how to configure the device for our specific network (it recommends leaving the settings alone and letting the software decide what to do, but it didn't seem to be doing well enough for us). E-mails to the authors were not returned.

Faced with this, we decided to implement our own system. Convinced that application-layer filtering would only get more difficult over time, we modeled our system on the Bandwidth Arbitrator, though it works on a slightly different principle and is configured in a different way. Please read on for more information.

20.3 Design

This software operates under a few fundamental principles:

- There exists a shared link (which may be asymmetric) which can become oversubscribed.
- Individual users should be free to use most (or all) of this bandwidth without restriction, **so long as there is bandwidth to spare**.
- Users who take up more than their fair share of bandwidth should be given lower priority in the future, and possibly have additional restrictions placed on them.
- **No content or port analysis is performed**.

That last one is important - go back and read it again. Basically, the software takes a very direct approach to managing bandwidth: you get a certain "fair share", and if you go over that you get penalized. If you want to use all your bandwidth up on P2P applications, that's fine, but don't expect your online games to have a decent ping time. We don't care how you spend your bandwidth, and by working this way we free ourselves from worrying about filtering every single type of traffic.

If you need a solution that will prioritize traffic based upon destination, port, or application, **use something else**. The purpose of this software is to improve the overall performance of the link, but it is completely content-agnostic. If you want to prevent users from running P2P apps, you need something else.

20.3.1 Requirements

The exiler software is written to uses existing open-source projects to accomplish its tasks:

- Perl 5.8+. Other versions of Perl may work as well; the features used by the scripts are not terribly esoteric, and the required modules are easy to find.
- OpenBSD 4.0 (the base operating system). We use OpenBSD because it has a built-in packet-queuing system (ALTQ - part of the PF package). Other versions of OpenBSD should work just as well, and it's possible that systems using PF (such as FreeBSD) may work, though this has not been tested.
- Net::Pcap (for traffic capture). This is a Perl module which interfaces directly to the pcap libraries on your system for capturing packets. The script uses this library to capture traffic. OpenBSD 4.0 includes a pre-packaged version of this library in its packages tree.
- Unix::Syslog (for log messages). This is a Perl module which allows logging information to be sent directly to the system. Because the script includes a mode that auto-daemonizes, we include this functionality to automatically log messages. For vehement anti-sysloggers, the logging code is highly centralized and easy to disable, so this software isn't vital to the function of the software. OpenBSD 4.0 includes a pre-packaged version of this library in its packages tree.

Hardware requirements are a little trickier; the primary concerns are having enough processing power to analyze the traffic in real-time, and having enough bus speed to bridge the packets without slowing them down unnecessarily. Memory is not a major concern (scales linearly with the number of active IP addresses), and disk space is almost an afterthought (a base OpenBSD install + 10MB should do it). To this effect, you should concentrate on getting a machine with a fast processor, good NICs, and a solid interconnect (*e.g.*, PCI-X).

You'll need three network interface cards (NICs). One is used for interfacing with the system (Unix hackers can eliminate this NIC if they want), and the other two are used as in/out for the pass-through connection. The two bridge NICs should be as high-quality as possible; some old or cheap NICs flake out when bridged or run under heavy load. Again, YMMV.

As a single data point, consider our system: we have an asymmetric connection that is 16Mb/s inbound and 2Mb/s outbound, and approximately 500 users. We run the script on a 2.4GHz Intel Pentium 4 with 512MB of RAM. We have two gigabit PCI NICs for the bridge, and a no-name 100Mb NIC for management.

The system uses about 15-20% of its CPU time for the shaping (about half of that is processing interrupts), and RAM usage for the script is usually well under 16MB.

As always, each installation will have factors that affect performance. The major impacts on this system are:

- **Packets Per Second**: processing is performed on each packet, so the number of packets per second is more important than raw throughput (in other words, lots of little packets is worse than fewer big ones, even if throughput is lower).
- **Number of Users**: the script tracks each distinct IP it sees during a sample period. The more users you have, the larger the data structures get. Perl was built for fast hash accesses, so performance is good. More users means more time spent processing.
- **Number of Connections**: the script also tracks connections from users on the LAN with remote addresses (in order to penalize users who open too many connections). However, tracking these connections takes time and memory. If you do not wish to track connections, you may wish to comment out the portion of the script dealing with connection tracking.

20.4 Theory of Operation

The software's main goal is to watch the network traffic and assess penalties ("exile") users who cross certain thresholds.

The software runs on a machine configured as a bridge. Normally, a bridging machine simply duplicates all traffic from one interface onto another. In this case, though, the machine filters some of this traffic using OpenBSD's PF-based queuing.

By default, we use a 3x3 setup: there are three different samples that we draw (*seconds*, *minutes*, and *hours*), and for each sample, there are three thresholds (*good*, *bad*, and *evil*). Thus, there are 3*3*3, or 27 possible states that a user's traffic can end up in. There is nothing magical about this number; we could have only 2 decisions at each level for a total of 8 states, for example. However, one driving consideration is that OpenBSD supports only 62 queues for traffic (unless the kernel is recompiled), so we strive for a number that falls within this limit (remember that we have to worry about both inbound and outbound traffic, so we actually have 2 * 27 or 54 queues in the default configuration).

The software listens for traffic using `Net::Pcap`, and keeps statistics for each sample time (by default 9, 180, and 3600 seconds). Each time it draws a sample,

it checks the total traffic for every IP address that has transmitted during that time, and compares it to the thresholds for that time interval. IPs are then assigned to queues based on the thresholds they exceed.

All of this is performed by the script automatically. The administrator must configure the thresholds and penalties associated with each sample. For example, a threshold might be something like "average traffic greater than 60kb/s". This threshold could be applied to a particular sample, such as "inbound traffic, 9 second sample", or "outbound traffic, 3600 second sample". In this way, the administrator determines what a "reasonable" use of the link is, and has some control over how strict depending on the time limit (for example, shorter sample intervals might allow more average traffic, to account for burstiness in short transfers).

The administrator also assigns penalties for each threshold. These penalties can be percentage-based (relative to the amount of bandwidth the link "should" get if there were no penalties), or absolute. There are three types of penalties, based on the Hierarchical Fair Service Curve parameters: realtime (guaranteed bandwidth), linkshare (probability that a queue will be given excess bandwidth), and upperlimit (cap on the amount of bandwidth a queue may ever use). By specifying these parameters, the admin can give lousy service (via realtime), lower priority (via linkshare), or throttling (via upperlimit) to a queue.

This document does not serve as a detailed introduction to HFSC; please visit the following pages for more information:

- http://www.cs.cmu.edu/~hzhang/HFSC/main.html
- http://marc.theaimsgroup.com/?l=openbsd-pf&m=110488079304643&w=2
- http://wiki.pfsense.com/wikka.php?wakka=HFSCBandwidthShapingNotes
- http://www.probsd.net/~exile/hfsc.pf.conf
- http://www.monkey.org/openbsd/archive/misc/0402/msg00666.html

The software runs as a daemon on the machine, running through the process described above.

Because the software integrates tightly with PF, the script includes modes for auto-generating configuration files for PF.

20.5 Installing OpenBSD

Exile is developed and tested on OpenBSD. As part of the installation of the software, you may need to install OpenBSD on the machine that will filter the

traffic. If you're well-versed in OpenBSD, feel free to skim this section to make sure you have all the necessary software installed.

What follows are **very cursory** instructions for getting a basic OpenBSD system installed. Please refer to the official OpenBSD documentation for more information.

The instructions below are written for OpenBSD 4.0. Other versions may work as well (we aren't doing anything too fancy).

20.5.1 Pre-Installation Checklist

Before you start, make sure you have the following:

- A computer that supports OpenBSD (i386 PCs are very well supported)
- The hard drive of the machine must not have any other data that you wish to save (we'll be erasing it)
- 3 network interface cards (NICs) that are OpenBSD-compatible. Even the no-name cards tend to work with OpenBSD, but we recommend checking the OpenBSD web site and list archives to find models that offer the best performance.
- An OpenBSD installation CD (or other installation media; see the OpenBSD documentation for other options)
- A network connection to the internet, and all necessary configuration parameters (IP address, subnet mask, DNS servers, etc)

Install all the NICs into the machine (if you haven't done so already). One of the NICs will be the **management** interface (the one used to connect to the machine and make changes). The other two are the **bridging** interfaces (in/out) that the filtered traffic will cross. The bridging interfaces should be as high-quality as possible (gigabit NICs are cheap, and worth the investment), and should have the same speed (just by two of the same make and model).

Once you have all the items, proceed to the next section.

20.5.2 OpenBSD Installation

Booting

Boot the computer from the installation CD. A bunch of diagnostic text will be displayed as the system searches for devices on the system. Eventually, you should get a prompt that says:

```
(I)nstall, (U)pgrade or (S)hell?
```

Type `I` for install and hit return.

The system will ask for a terminal type; just hit return for the default.

The system will ask for a keyboard mapping; just hit return for the default.

You will be asked to make sure you don't have any data on the disk that you want to keep (you don't, do you?). Type `yes` and hit return to continue.

Partitioning

The system will list all the available hard drives on the machine. Assuming you only have one drive, it should be listed in brackets. Hit return to accept the default disk as the installation target.

Answer `yes` when asked if you want to use the entire disk for OpenBSD.

Assuming there are no existing OpenBSD partitions on the disk, you will now be put into `fdisk` (the partition editor). You need to create two partitions on the disk: one for the main system, and one for virtual memory (swap).

Start by entering `z` to clear the partition table. Now type `p` to get some basic information about the disk. The size of the disk is displayed as partition `c:`. All numbers are in sectors (which are 512 bytes, or half a KiB).

In general, swap should be sized to 2 times physical RAM, though if you have a lot of RAM it can be less. Leave the rest of the disk for OpenBSD. To figure out how much space to set aside, multiply your RAM size in MiB by 4096 (1024 to convert MiB to KiB, 2 to convert KiB to sectors, and 2 to be double physical memory). That will be the size of the swap partition. Subtract this number from the total, and you know how large to make your main partition.

For example, suppose I have an 80GiB drive and 512MiB of RAM. My swap partition should be 512 * 4096, or 2097152 sectors. My total disk space is 80418240 sectors, so my OpenBSD partition should be 78321088 sectors and my swap partition should be 2097152.

To create the partitions, type `a`. For the main OpenBSD partition, select `a` as the partition name (should be the default). Use the default offset (usually 63), and the number of sectors you calculated above for the size (78321088 in our example). For the filesystem, choose `4.2BSD` (should be the default). For the mount point, use the default of `none`.

Now type `a` again to create the swap partition. Choose `b` as the name, use the default offset, and just hit enter for the size (it will default to taking the remaining part of the disk). The filesystem should default to `swap`, which is

what you want.

Once you're done, type `q` to save your changes and quit.

You'll be asked (again) if you're sure that you want to destroy everything. Make sure you're certain, and type `yes`.

The system will format the drive (which may take a few minutes).

Network Configuration

You'll now be prompted to give a hostname to the machine. Pick one according to your system naming scheme.

You'll now be asked if you want to configure the network; say `yes`.

You'll be given a list of interfaces to configure. Choose the one that will be the management interface for this machine. If all your NICs are the same make and model, it may be difficult to guess which one you want. I generally pick the one with the lowest number, configure it, and move the ethernet cables if I guessed wrong.

Work through the options for the interfaces, choosing DHCP options, IP addresses, and routes as necessary.

When you return to the interface menu, enter `done` (we don't need to configure the two bridged interfaces now).

Enter the domain name for this system.

Enter the nameserver(s) this machine should query.

Say `yes` when asked if you want to use the nameservers.

Enter the default route (gateway) for the network you're on.

Say `no` to editing hosts by hand and manual configuration.

Base System Installation

Enter a root password for the machine, and verify it.

You'll now be asked where the base system installation files are. Use `http` (unless you have them somewhere else).

Enter any necessary proxy information when prompted.

If you don't have a preferred mirror server that you use, enter `yes` to show a list of available servers.

When prompted, enter a number from the list, or one of your hand-selected servers. Provide any requested path information (or accept the defaults if you picked from the list).

Now you'll be asked what packages to install. The defaults are fine; we don't need any of the X11 packages (you may install them if you need them for something else). If you're really tight for space, you can deselect the games packages (and possibly others), but only do so if you know what you're losing.

Once you've made any package selections, type `done` to install them. Answer `yes` to signal that you're ready, and the system will download, decompress, and install the requested packages. Depending on the speed of your network connection, you may want to go grab something to drink.

When the installation is complete, the system will prompt you again for the location of any installation media. Type `done` this time, as we've installed everything.

You will now be asked some minor questions about the default configuration of the system. Make whatever choices fit with your usual policies (when in doubt, go with the defaults).

Select a timezone for the system (*e.g.*, `US/Eastern`).

A few final scripts will run, and then you should receive a message that the system has been installed. Issue the `halt` command, and when the system has halted you may eject the installation CD and reboot the computer using its power switch.

The machine will now boot under its "own steam", and you're ready to move on to the next section.

20.5.3 Basic System Configuration

Your machine should now boot by itself to the `login` prompt. You may now log in as `root` and you should get a basic command prompt.

At this point, ping a host to make sure your network is up, and everything else seems in order. You can also make any necessary tweaks that you usually make to the system (configuring `syslog`, changing command shells, installing preferred editors and other tools, *etc.*). Once you've done that, come back to this document (go ahead, we'll wait).

NIC Configuration

You'll need to tell OpenBSD to bring up your network cards at boot time. The management interface should be up and running, so we need to bring up the two bridging interfaces. Because the bridge will simply pass traffic from one interface to the other, they do not need to have IP addresses, or any other configuration.

Your NICs will be listed in the output of the `dmesg` command; just scan for the make of your card and it should show up with an identifier. For example, my gigabit Intel cards show up as "Intel PRO/1000MT", with system identifiers of `em0` and `em1`.

Once you know the names of the bridge cards, type the following for each one, substituting the interface name (*e.g.*, `em0`) for `IFNAME`:

```
echo 'up' > /etc/hostname.IFNAME
```

Bridge Configuration

Next, we need to tell OpenBSD which cards to bridge together. We do this by echoing a few commands to a special bridge configuration file (substitute your interface names for `IFNAME`):

```
rm -f /etc/bridgename.bridge0
echo 'add IFNAME1' >> /etc/bridgename.bridge0
echo 'add IFNAME2' >> /etc/bridgename.bridge0
echo 'up' >> /etc/bridgename.bridge0
```

Enabling Forwarding

OpenBSD must be told to forward packets by default. To do so, edit the file `/etc/sysctl.conf` and change the value of `net.inet.ip.forwarding` to `1`.

Enabling PF

PF is the OpenBSD **P**acket **F**ilter, which includes the queuing code we'll use to slow down IP traffic. To enable it, edit the file `/etc/rc.conf.local` and add the following line:

```
pf=YES
```

Reboot and Check

Now that you've made all the necessary configuration changes, reboot the machine by issuing the following command:

```
shutdown -r now
```

When the machine reboots, log back in and type `ifconfig`. You should see all of your network interfaces and the bridge as "UP,RUNNING". If you see that, you're ready to move on to the next section.

20.5.4 Perl Libraries

All of the Perl libraries that the script requires are available as packages for OpenBSD. Unless you feel like rolling your own, all you need to do is install the pre-compiled packages from your favorite OpenBSD mirror:

```
pkg_add http://openbsd.mirrors.pair.com/ftp/4.0/packages/i386/p5-Net-Pcap-0.04.tgz
pkg_add http://openbsd.mirrors.pair.com/ftp/4.0/packages/i386/p5-Unix-Syslog-0.100.tgz
```

That will install the `Net::Pcap` and `Unix::Syslog` Perl modules.

20.5.5 Wrapping Up

At this point, you have an OpenBSD machine that is correctly configured for use with the software. Please proceed to the Installation section of this document for information on setting up our software.

20.6 Installation

This section describes the installation and configuration of the actual Exile software. **Note:** it is assumed that you already have an OpenBSD machine with PF, bridging, and the required Perl modules installed. If you don't, please read the OpenBSD Installation section of this document and follow the directions there before installing the software.

20.6.1 Downloading

As this is beta software, we currently only support downloads by directly checking the code out from our Subversion repository. Don't worry; it isn't too diffi-

cult!

First, make sure you have a Subversion client installed. OpenBSD users can install a prepackaged version by running:

```
pkg_add http://openbsd.mirrors.pair.com/ftp/4.0/packages/i386/subversion-1.3.2.tgz
```

Next, move into your `/usr/local` directory:

```
cd /usr/local/
```

Now, you're ready to check out the project:

```
svn co svn://svn.suffieldacademy.org/netadmin/trunk/software/exile
```

That will create a directory called `exile` with all of the software contained inside it.

If you ever need to update the software to the latest version, move into the `exile` directory and run:

```
svn up
```

20.6.2 Configuring

Configuration requires a few distinct steps, which are outlined below. Please note that the `conf` folder contains example configurations; you should copy these configurations to your own files and make your changes there.

The example configurations are well-commented. Additionally, information about the configuration parameters can be found in the script itself (by running `perldoc sbin/exiler`).

Exile Parameters

Start by editing the main configuration file (we suggest making a copy of the example.conf, renaming it, and customizing this file).

The example file has detailed comments; in short, you'll need to perform the following actions:

- Specify the paths and interfaces on your system

- Define the parameters of your network (subnets and bandwidth)
- Configure the thresholds and penalties

For first-time configurations, we recommend setting `$DEBUG=1`. This will cause the script to run in the foreground and print more detailed information.

Two of the configuration variables are paths that point to the PF template and local rule files (described below). We recommend using names that are specific to your site, rather than the included "example" files. Again, make copies of the example files and rename them before customizing.

PF Template

The PF template file is a configuration file for OpenBSD's packet filter. The template allows you to define default policies, custom queues, and other rules (such as tables). Additionally, it includes macros that the script will replace with actual configuration variables.

Thus, you set up your custom rules and queues, and also insert the template macros that will expand to the rules and queues that exile uses. If you're not sure what to put, use the example template (it has a reasonable, if not highly secure, configuration).

Local PF Rules

Finally, you must create a "local" PF rules file. This script contains rules that will be executed before any of the exile rules are run. Additionally, these rules will always be in effect, even when the exile rules are not active.

You can use these rules to send certain traffic to "exempt" queues to prevent exiling, or other custom items as you see fit.

If you're not sure what to put here, use the example ruleset and customize the interface options that are specific to your machine.

20.6.3 Running

The script can be run in several modes, depending on what you wish to accomplish. We describe the most common modes of operation below.

Note: all the examples below assume that you are in the `exile` directory (where you installed the script). For example, if you put the script in `/usr/local/exile`, then you should `cd` to that directory before running any of the commands below.

Also, running the script with no arguments will cause it to print a brief synopsis.

Generating Config Files

Before you can run the daemon, you must first generate the PF config files. This can be accomplished by running the script with the "gen" option:

```
./sbin/exiler gen conf/my-config.conf
```

This will parse the configuration file and build the PF configuration files. Inspect the resulting file (which will be in `run/pf/pf.conf` by default) to confirm that they are to your satisfaction.

You can test the generated files by executing:

```
pfctl -nf run/pf/pf.conf
```

If any errors are reported, you'll need to correct them before continuing (most common errors involve specifying too much bandwidth for a queue).

Note: PF may print warnings like:

```
the sum of the child bandwidth higher than parent
```

These errors are harmless, and have to do with the way that HFSC queues operate in OpenBSD. Errors that you need to worry about will say "syntax error" in them.

Now you can load the new ruleset (note that this does not turn on exiling; it just installs the framework that the exiling will plug into). Do this **while logged on through a console** (in case there is an error in the config, which would disconnect you if connected remotely):

```
pfctl -f run/pf/pf.conf
```

If the machine is still connected and traffic is passing normally, you are in good shape.

If you want to make your generated ruleset load automatically on boot, add the following line to `/etc/rc.conf.local`:

```
pf_rules=/path/to/exile/run/pf/pf.conf
```

Listening to Traffic

Now that the PF rules are loaded, you can run the script in "listen" mode. This will run the script in such a way that it does all of its normal work, but doesn't actually assign IP addresses to the penalized queues. It's a good way to test the configuration, and ensure that the machine is fast enough to listen to all the traffic.

Run the script with the "listen" option:

```
./sbin/exiler listen conf/my-config.conf
```

If debugging is enabled, you'll see output as the script scans traffic and adds or removes penalized IP addresses. On a separate terminal, you can consult the files in `run/q` to see which addresses would have been penalized.

Exiling

Once you're sure that your configuration is correct and listening well, you can change to "run" mode. This mode is equivalent to using "listen", but it enables the PF rules to actually assign packets to queues. You can run it by typing:

```
./sbin/exiler run conf/my-config.conf
```

Note that you must have already generated and loaded the PF configuration files.

If you prefer, you may manually achieve the same state as "run" by executing "qon" mode, and then "listen" mode. This can be useful when the script is running in daemonized mode and you don't wish to terminate it. Using the "qon" and "qoff" modes will disengage the actual penalizing process while leaving the daemon in a mode to watch traffic.

You may find it helpful to use the `pftop` program to watch the queue assignments. A pre-compiled version is available for OpenBSD:

```
pkg_add http://openbsd.mirrors.pair.com/ftp/4.0/packages/i386/pftop-0.5.tgz
```

Run `pftop` and switch to the queue assignment screen. You'll see real-time updates of the amount going to each penalty queue.

Additionally, we've created some scripts to graph the performance of the application in near-real-time. Please read on for more information on setting up this service.

20.7 Performance Graphing

20.7.1 Warning

Note: the scripts for performance graphing are still under development, and currently are coded with settings that are specific to Suffield Academy. The script should be easy to modify, but please keep in mind that they are not as simple to configure as the main application.

20.7.2 Overview

To track the performance of the exiler application, we've written a simple daemon that collects data from the exile queues and stores them in RRD (**R**ound **R**obin **D**atabase) files. Another script queries these files and generates graphs of the performance of the application (for an example of the graphs generated, please see the Introduction section of this document).

20.7.3 Installation

The software uses the "rrdtool" package to perform its data capture and graphing. Thus, you should begin by installing RRDtool on your machine. There is a pre-compiled package for OpenBSD; you can install it by running:

```
pkg_add http://openbsd.mirrors.pair.com/ftp/4.0/packages/i386/rrdtool-1.0.49p3.tgz
```

Once you've install RRDtool, you can look at our scripts. If you've checked out the `exile` directory from our Subversion repository, you already have all the necessary scripts. They live in subdirectories of the `rrd` directory.

There are three subdirectories of the `rrd` directory. Briefly:

- The `bin` directory contains the daemon script which collects data from the system and stores it in RRD files.
- The `cgi-bin` directory contains a Perl CGI script which accesses the RRD files and produces graphs of the data.
- The `rrds` directory is where the scripts store the saved RRD files (by default – you may change the path to whatever you wish).

The `exile_queue_rrds` script in the `bin` directory should be launched by root. It will fork to the background and begin collecting statistics about all OpenBSD

PF queues on the system. The results will be stored in RRD files in the directory provided as the first argument to the script.

If you like, you may add a line to launch the script to `/etc/rc.local` so it begins running automatically on startup.

The `exile_queues` script in the `cgi-bin` directory must be run from within a web server. Edit the script to provide the correct path to the RRD files (this should be the same as the argument you passed to `exile_queue_rrds` above), and then run from a web browser.

If you don't have a web server installed, you might try `thttpd` (part of the OpenBSD ports collection). It's a very small and lightweight HTTP server, and can easily be set up to run our CGI. For example, if the exile directory is installed at `/usr/local/exile`, you can execute thttpd as:

```
thttpd -p 80 -d '/usr/local/exile/rrd' -c '/cgi-bin/*' -u www
```

and be able to browse to http://yourhost/cgi-bin/exile_queues to try the script out immediately.

20.7.4 Customization

At this time, all the RRD scripts use hard-coded values for their configuration. If you wish to make changes, you'll need to edit the source code directly. The current values are reasonable for a machine that only runs the exile application, but if you customize the machine you may need to change a few of the settings. Anyone with some experience using Perl should be able to make the necessary changes.

Chapter 21

Failover and High Availability

Last updated 2008/03/18

21.1 Introduction

Network infrastructure occasionally requires maintenance, no matter how well-designed it is. However, many organizations have come to depend on network services to such a degree that downtime adversely affects the entire organization. Thus, we must find a way to increase the **availability** of service, even when parts of the infrastructure are not operating.

In the network world, increased availability usually requires duplicate hardware, and additional expense. We don't have a lot of money to throw at the problem, so the approaches outlined in this document use a minimum of hardware and expense to accomplish their task. Obviously, you must decide what level of downtime your organization can withstand, and balance this with the added cost and complexity of a suitable failover system.

21.2 Terms and Definitions

Before we begin, some quick discussion of terminology:

- When a network service is "up and running", we say that it is **available**. The goal is to achieve the highest **availability** possible, so that users of

the service are not disrupted. When special steps are taken to prevent outages, we aim for **high availiability** (or **HA**).

- High availability usually requires **redundant hardware** (so as to tolerate the loss of a particular system). Several strategies exist for using this redundant hardware.
- **Load balancing** involves having service requests routed to all redundant hardware. If one server dies, the balancer leaves it out of rotation until it returns. This provides maximum hardware utilization, but requires special load-balancing hardware and/or software.
- **Hot-spare failover** involves setting up two duplicate servers, but only one is active at any given time. The two servers constantly communicate, and in the event that the primary becomes unavailable, the secondary server takes over. This setup is relatively simple to configure, but can involve a large amount of "wasted" resources for the backup machine, as it is not frequently used.
- **Warm-spare failover** is similar to a hot-spare, except that the spare might do something else in addition to serve as a backup. When the spare detects a fault in the primary, it reconfigures itself to take over the primary's services in addition to its own.
- **Cold-spare failover** basically means that a duplicate machine is available to take over, but it must be booted and/or configured by hand before services resume. Cold-spare requires no up-front configuration, and can be inexpensive (especially if a single machine is a spare for many others), but it involves the largest amount of downtime while the spare is configured.

21.3 Mac OS X IPFailover

Mac OS X Server 10.4 ships with built-in IP-based failover support. Out of the box, it's a **warm-spare failover** setup; the backup server will take over the IP address of the failed server automatically, but it is up to the administrator to configure the server to start any additional services that may be required.

Failover is implemented using two system-level daemons:

1. `heartbeatd` runs on the primary server, and broadcasts "availability" packets at regular intervals. These packets serve as an announcement that the machine is up and running.
2. `failoverd` runs on the secondary server, and listens for the broadcast packets from the primary. In the event that the broadcasts are not received, the secondary takes over the primary's IP address.

Apple requires the following to implement IP-based failover:

- Both machines be on the same **public** subnet
- Both machines are connected via an independent **private** subnet
- Each machine have its own unique IP address on both subnets

Again, Apple's solution only transfers the IP address from one server to another; you are responsible for ensuring that the secondary server contains the data and configuration necessary to actually take over the network services.

To help with this, Apple uses a customizable scripting framework to help you launch processes during a failover situation (more on this below).

21.3.1 Configuring Failover

For the purposes of this document, we'll be using the following sample names. **You must replace the names and IP addresses with those of your actual equipment.**

- The **primary** server has public IP address `10.0.0.100` and private IP address `192.168.0.1`.
- The **secondary** server has public IP address `10.0.0.200` and private IP address `192.168.0.2`.

Public Network

First, ensure that both machines are on the same public subnet, and are able to reach each other. In our example, both machines are on the `10.0.0.0/24` subnet.

If you have enabled firewall software, ensure that it allows UDP traffic destined for port `1694` to reach the server. **Note**: Apple's built-in firewall entry for "IP Failover" incorrectly defaults to TCP traffic for this rule; you must also enable UDP.

Private Network

Next, connect both servers together via an independent private network. This can be over a second ethernet connection, firewire cable, or other network.

Ensure that the secondary network interface appears **below** the primary interface in the **Network Settings** preference pane; this ensures that the machine will only use the private network when the public network is down.

Also ensure that the private network has **no DNS information** specified. All DNS information should be obtained from the public network.

If you have enabled firewall software, ensure that it allows UDP traffic destined for port `1694` to reach the server. **Note**: Apple's built-in firewall entry for "IP Failover" incorrectly defaults to TCP traffic for this rule; you must also enable UDP.

Configuring the Primary Server

On the primary server, edit the `/etc/hostconfig` file and add a line containing the **broadcast addressess* of both the public and private subnets. Using our sample IPs from above, our line looks like:

```
FAILOVER_BCAST_IPS="192.168.0.255 10.0.0.255"
```

Save the file and reboot the primary server (or manually start the `IPFailover` service using `SystemStarter`).

The primary should now have a `heartbeatd` process running, and a `tcpdump` listening for port `1694` should show regular traffic from the server on both its public and private interfaces. If this is happening, move on to the next step.

Configuring the Secondary Server

On the secondary server, edit the `/etc/hostconfig` file and add lines defining the primary server's IP address, and the interface that should assume the address in the event of a failover. Additionally, you may specify an e-mail address to send notifications to when a failover occurs:

```
FAILOVER_PEER_IP="10.0.0.100"
FAILOVER_PEER_IP_PAIRS="en0:10.0.0.100"
FAILOVER_EMAIL_RECIPIENT="root@example.org"
```

The second line's syntax says that the address `10.0.0.100` should be added to the `en0` interface. If your machine has multiple interfaces, specify the one that should take over the primary server's address.

Save the file and reboot the secondary server (or manually start the `IPFailover` service using `SystemStarter`).

You should now have a `failoverd` process running on the secondary server, ready to take over from the primary. You may test this by unplugging **both** network interfaces from the primary (or simply shutting it down). The secondary server should notice that the server is unavailable and take over the IP address. Additionally, you should receive an e-mail notification about the takover.

If you reconnect the primary server, the secondary should notice and relinquish its address within 15 seconds. Again, an e-mail notification is sent to confirm the change.

21.3.2 Failover Transition Scripting

The process described above handles the takeover of an IP address from a primary server to a secondary one. However, that's all it does; if your secondary server is not already running all the services that the primary uses, then the takeover won't help you.

For this reason, Apple has provided a scriptable framework so you can take specific actions whenever a secondary server takes over (or gives up) the primary's address.

File Locations

Apple's IPFailover scripts look for a directory in `/Library/IPFailover` named after the public IP address of the primary server. In our example, the secondary server would have a directory named:

```
/Library/IPFailover/10.0.0.100
```

This directory can contain several scripts, outlined below:

- `Test`, which is run before any takeover is attempted. If the script returns with zero status, the takeover continues; if it returns non-zero status the takeover is aborted. This allows for conditional takeover depending on other external factors.
- `PreAcq.*` scripts get run before the primary address is added to the secondary. Any script starting with the prefix `PreAcq` is run, and the ordering is determined by the name of the file (*e.g.*, `PreAcq-1` would run before `PreAcq-2`).
- `PostAcq.*` scripts get run after the primary address has been acquired by the secondary server. Execution rules are the same as with `PreAcq` scripts above.

- `PreRel.*` scripts get run just before the secondary server gives up the primary address. Execution rules are the same as with `PreAcq` scripts above.
- `PostRel.*` scripts get run after the primary address has been relinquished by the secondary server. Execution rules are the same as with `PreAcq` scripts above.

You may have as many of each script as you want, and they may perform any scriptable tasks. For example, you might use the scripts to start a service after aquiring the primary's address, and then stop the service when the primary comes back.

We've built a template directory containing simple scripts that are easily customized:

Suffield IPFailover Config Template

21.3.3 Peering Failover

It is possible to set pairs of machines up in a peering configuration such that each acts as the backup for the other. Simply add the requisite lines to `/etc/hostconfig` on both machines and they'll act as a backup for each other.

21.3.4 Using the Suffield Configs

We keep all of our script directories under version control. To use them, you must check out the repository for the primary host you're interested in onto the secondary server that backs it up.

Note: This document assumes you've set up basic IPFailover as described above. That means you've edited `/etc/hostconfig` and "vanilla" failover is working.

Check out the config from our Subversion repository, substituting the actual primary IP address for <PRIMARY_IP>:

```
cd /Library/IPFailover/

sudo svn checkout \
svn://svn.suffieldacademy.org/netadmin/trunk/software/failover/host_configs/<PRIMARY_IP>
```

This will check out the configuration directory named after the IP address and place a working copy in the `/Library/IPFailover` directory. You may change into this directory at any time and run `svn up` to merge in any updates to the configuration.

Chapter 22

NFS (Network File System)

Last updated 2008/03/18

22.1 Introduction

22.2 NFS Exports

To allow other machines to connect to a server, you must **export** directories from the server to the clients.

Traditionally, the list of exported directories has been defined in a file (`/etc/exports`). However, under Mac OS X Server, the list of exports is stored in NetInfo (Apple's directory service).

In general, an **exports** entry must specify the following:

- The directory path(s) to export
- The clients to export to
- Access restrictions on the paths

Note that NFS is an inherently *insecure* protocol, so great care should be taken to prevent abuse. If possible, attempt to follow these guidelines:

- Export filesystems to as few clients as possible. Use a restricted network, or a specific client list to prevent unauthorized access.

- Export filesystems as **read-only** to prevent attacks from other machines.

The examples below follow these guidelines.

22.2.1 Adding an Export

Note: Apple's NFS daemon requires that all shared paths on a single device be mentioned in the same configuration directive. So, if you have three directories on a disk you'd like to share, all three must be in the same configuration directive (you can **not** list them individually).

Let's suppose we wish to export a few directories, all on a single device. The device is mounted at `/Volumes/Snapshots/`, and the folders we wish to share are called `Users`, `Groups`, and `Web`.

Keeping in line with our security recommendations, we'll be exporting these directories only to a specific subnet. Additionally, we'll export the filesystem as read-only to prevent changes from external clients.

To add the export directive, run the following in a terminal:

(**Note:** order is important; the "name" property must be set last!)

```
sudo niutil -create . /exports/Snapshots
sudo niutil -createprop . /exports/Snapshots opts 'ro'
sudo niutil -appendprop . /exports/Snapshots opts 'maproot=root'
sudo niutil -createprop . /exports/Snapshots opts 'network=192.168.1.0'
sudo niutil -createprop . /exports/Snapshots opts 'mask=255.255.255.0'
sudo niutil -createprop . /exports/Snapshots name '/Volumes/Snapshots/Users /Volumes/Snapshots/Groups /Volu
```

Now restart the nfs daemon:

```
sudo killall -HUP mountd
```

Look for any startup errors in the system log:

```
tail -f /var/log/system.log
```

If the logs are clean, you're ready to connect from an NFS client.

22.3 NFS Clients

This section not yet complete.

Chapter 23

AimSniff

23.1 Introduction

This document describes Suffield Academy's use of AimSniff, an open-source system for scanning and classifying AOL Instant Messages.

23.2 Dependencies

These instructions assume a moderate level of Linux system administration experience. Some of the dependencies for AimSniff are not installed by default on many systems, so you must fetch and build them yourself.

For this tutorial, we make several assumptions about the system you are configuring. If your configuration does not match exactly, the instructions may still work. However, you will need to be mindful of any deviations from the following:

- A computer with Debian Linux installed (these instructions written with the pre-release "Sarge" version of Debian). Please see our Debian Installation Instructions for more information on installing this version of Debian.
- A computer with two NICs installed. One will be used for sniffing the network, and the other will be used for regular communication with other hosts. We require this because our switching equipment does not allow a port to pass normal traffic when it is in "monitor" mode.
- A MySQL or PostgreSQL database. We have patched AimSniff to store data into PostgreSQL. If you prefer MySQL, use the stock AimSniff package, which uses MySQL by default.

If you've got all these things, then we're ready to go!

23.2.1 Network Setup

If your swtiching hardware does not allow normal traffic when a port is set into "monitor" (or sniffing) mode, you'll need a second NIC in your machine. Because no normal traffic will pass to this interface, we can give it a bogus static address.

Edit the file `/etc/network/interfaces` and create (or modify) a stanza for your second ethernet card. In our case, the unused card was `eth1`:

```
auto eth1
iface eth1 inet static
      address 127.127.127.127
      netmask 255.255.255.255
```

Now, bring the interface up using `ifup eth1`. You should see the interface appear with the bogus address listed above.

23.2.2 AimSniff Dependencies

In order to run AimSniff, you'll need several Perl modules (all of which are documented in the AimSniff README).

Several of these dependencies exist in Debian, and can be installed directly via apt-get. Others, however, must be fetched and built by hand.

Debian Packages

The easiest dependencies to install are those included in Debian. Run the following command (as root) on your machine:

```
apt-get install smbclient libunicode-string-perl libnet-pcap-perl \
                libdbi-perl libproc-process-perl libproc-daemon-perl \
                libunix-syslog-perl
```

The command should appear all on one line, or you may use backslashes to continue to new lines, as we have done above.

This may require installing other dependencies as Debian sees fit. Install any additional required packages.

SMBClient is required for the resolution of NT usernames on a Windows network, and is not strictly required for AimSniff to operate. If you do not need this functionality, you may omit `smbclient` from the install process.

dh-make-perl

To build additional Perl packages that aren't included in Debian, we'll need to download and compile them ourselves. Fortunately, Debian includes a tool, `dh-make-perl`, which automates much of this process.

To install `dh-make-perl`, simply install it using APT:

```
apt-get install dh-make-perl
```

Again, this may install extra dependencies. Go ahead and accept their installation.

Finally, you may wish to customize the `DEBFULLNAME` and `DEBEMAIL` environment variables. While they won't affect whether your packages build or not, they will set the maintainer information for the packages.

NetPacket Package

The `NetPacket` Perl package is required by AimSniff in order to run.

Lazy people should feel free to download a pre-built Debian package from our `debs` repository. If that doesn't work, follow the instructions below to build your own.

The source package can be downloaded from CPAN at the following link:

http://search.cpan.org/CPAN/authors/id/A/AT/ATRAK/NetPacket-0.04.tar.gz

Download the package source to a suitable directory (`/usr/local/src/` might be a good choice, though the location doesn't matter).

Unpack the source distribution:

```
tar -zxf NetPacket-0.04.tar.gz
```

and change into the resulting directory:

```
cd NetPacket-0.04
```

At this point, we're ready to "debianize" the Perl package. Run the following:

```
dh-make-perl
```

That will turn the source package into a Debian source tree. Now you can run:

```
dpkg-buildpackage -rfakeroot
```

(or `-rsudo` if you prefer to use sudo)

Which will create a debian package in the parent directory. Several files starting with `libnetpacket-perl` will be created. The one we're interested in should be called `libnetpacket-perl_0.04--1_all.deb`

Go ahead and install your new package now, by changing to the parent directory and running `dpkg`:

```
cd ..
dpkg -i libnetpacket-perl_0.04--1_all.deb
```

Proc::Simple Package

The `Proc::Simple` Perl package is required by AimSniff in order to run.

Lazy people should feel free to download a pre-built Debian package from our `debs` repository. If that doesn't work, follow the instructions below to build your own.

The source package can be downloaded from CPAN at the following link:

http://search.cpan.org/CPAN/authors/id/M/MS/MSCHILLI/Proc-Simple-1.21.tar.gz

To build the package, follow the instructions above for installing the `NetPacket` package. This time, however, substitute `Proc-Simple-1.21.tar.gz` for the source package, and `libproc-simple-perl_1.21-1_all.deb` for the name of the resulting Debian package file.

Again, a simple install should finish the job:

```
dpkg -i libproc-simple-perl_1.21-1_all.deb
```

23.2.3 Testing It Out

At this point, you have installed all of the dependencies required to run AimSniff in collector mode.

You should download AimSniff and try to run it. If you're using the stock distribution, you may obtain it here:

http://www.aimsniff.com/releases/aimsniff-0.9d.tar.gz

Suffield Academy has customized the collection script to work with our database server (we use PostgreSQL instead of MySQL). The latest version of the script can be downloaded from our `aimsniff-postgres` repository.

Once you've downloaded (and untarred, if you're using the stock distribution) the files, make the `aimSniff.pl` script executable and run it with the following arguments:

```
chmod 755 aimSniff.pl
./aimSniff.pl --nodb
```

The program should begin running. If you're not root, it will not have access to the network device and will quit. This is OK.

If you get any error messages from Perl about being unable to locate a module, then you need to make sure you installed all the dependencies as described above. The message should give you a hint as to what module is missing (*e.g.*, if it can't find `Proc::Simple.pm` then you'll need to redo the installation of that package).

Once the program starts up normally, you're ready to move on to the next step.

23.3 Database Setup

Now that we've installed the dependencies, we're ready to set up the database. AimSniff can run in one of several modes:

- Display only (sniffed data are printed out, but not stored)
- Local file (sniffed data sent to a flat file)
- Database (sniffed data sent to a relational DB)

We'll focus on the third method, as it provides the most flexibility in reporting later on. If you don't want/need a database, consult the AimSniff documentation on how to use the other collection options.

23.3.1 Selecting a DB Server

AimSniff can connect to a local or remote database server in order to store its data. You can either set up a local database on the same machine as AimSniff, or you can use an existing database on another machine.

At Suffield, we use a separate machine, as we already have a centralized database server. Additionally, our AimSniff machine has limited resources (processor, RAM, HD space), so sending the data to another machine keeps the overhead low on our collection machine.

This tutorial assumes you know how to install and configure one of the following relational database packages:

MySQL MySQL is the default database backend for AimSniff. If you already use MySQL, or if you don't have a database set up yet, you may want to use MySQL. If you choose MySQL, download the stock distribution of AimSniff instead of our version (which is modified to use PostgreSQL instead).

PostgreSQL PostgresSQL is our database of choice at Suffield Academy, for a variety of reasons that are beyond the scope of this document. We have modified AimSniff to work with Postgres instead of MySQL. If you wish to use Postgres as your backend, please download our modified copy of AimSniff.

Once you've selected a database, you should install it on your database server (if you haven't done so already). From this point on, we assume you have a database server up and running, and that you have permission to make administrative changes to the database.

Note that the database server may be the same machine as your AimSniff collector. While these instructions treat the AimSniff machine and the database server as logically separate components, they may reside on the same machine.

23.3.2 Using MySQL as the Backend

If you're using MySQL, you'll need to install Perl support for that database. Run the following commands to install the correct packages:

```
apt-get install libdbd-mysql-perl libdbd-mysql
```

Next, ensure that you have downloaded the correct version of AimSniff. You need the stock (unpatched) version of AimSniff, which can be obtained from:

http://www.aimsniff.com/releases/aimsniff-0.9d.tar.gz

Unpack the distribution, and move into the source directory.

You'll need to perform three steps to prepare the database for AimSniff:

1. Create the actual database by running this command on your database server:

   ```
   mysqladmin create aimsniff
   ```

2. Import the table structure file called `table.struct` from the AimSniff source distribution. If your database server is not on the same machine as the file, you may need to provide additional arguments (or copy the struct file onto the database server and run the command there). A sample local command would be:

   ```
   mysql aimsniff < table.struct
   ```

3. Finally, grant access privileges for a new MySQL user to modify the database. Log into MySQL as the `root` user and issue something like the following:

   ```
   GRANT ALL ON aimsniff.* TO username@hostname IDENTIFIED BY 'password';
   ```

   You should pick a username and password to use. Remember these values; we'll need them later to configure AimSniff. The hostname should be the name or IP address of the computer that will be running the sniffer. If this is the same machine as the database server, use `localhost` or `127.0.0.1`.

To test your setup, try to connect from the collector to the database server using the standard MySQL tools:

```
mysql -u username -p -h hostname
```

Be sure to substitute the correct username and hostname for the database server. You will be prompted to enter a password; use the one you selected earlier.

If the connection is sucessful, you're ready to move on to the next step, and configure AimSniff to run on your machine.

23.3.3 Using PostgreSQL as the Backend

If you're using PostgreSQL, you'll need to install Perl support for that database. Run the following commands to install the correct packages:

```
apt-get install libdbd-pg-perl libdbd-pgsql
```

Next, ensure that you have downloaded the correct version of AimSniff. You need our specially patched version, which can be obtained from our `aimsniff-postgres` repository.

You'll need to perform four steps to prepare the database for AimSniff:

1. Create a user who will own and access the database. On the database server, run the following command:

   ```
   createuser -A -D -P -E aimsniff
   ```

   (You may choose a different username if you wish.)

   You will be prompted to enter a password for this user.

2. Create a database to hold the AimSniff data, and assign ownership to the user you just created:

   ```
   createdb -O aimsniff aimsniff "Database for AimSniff collection data"
   ```

   The username is the first `aimsniff`; the database name is the second. Again, you may use whatever names you see fit.

3. Finally, load in the table structure for AimSniff. It's contained in the file `postgres.struct` from our patched distribution.

   ```
   psql -U user -d database -h server -f postgres.struct
   ```

   As always, susbstitute your values for name, database, and server.

4. You may need to modify your database server's `pg_hba.conf` file to allow database access from your collector machine. If you're on a local machine, this shouldn't be a problem, but remote machines may not be granted remote access over IP by default. Consult the PostgreSQL documentation for more information on granting host access.

At this point, you should be able to connect to the database server from the collector machine. If you have the `postgres-client` package installed on the collector, you should be able to connect to the database using the following:

```
psql -U user -d database -h server
```

Substituting the correct values as before. If it works, you're ready to move on to the next step!

23.4 Configuring AimSniff

At this point, you should be able to start AimSniff on the command line, and you should have your backend database set up correctly.

We're now ready to configure AimSniff, set it to start up on your collector, and configure your switching equipment to forward traffic to the collector.

23.4.1 AimSniff Configuration File

Find the file `aimsniff.config` (stock distribution) or `aimSniff.cfg` (Postgres distribution). Copy this file onto your collector, and store it with your other configuration files (`/etc/` might be a good spot for it).

Edit the file to include the correct parameters for your setup. Be sure to customize the database settings.

23.4.2 AimSniff Startup Script

Find the file `rc.aimsniff` (stock distribution) or `init.d.aimsniff` (Postgres distribution). Copy this file onto your collector, and store it with your other startup scripts (under Debian, `/etc/init.d/aimsniff` should be the final name of the file).

If you wish, you may set up AimSniff to lauch when the computer starts up. On Debian, you can do this by executing the following:

```
update-rc.d aimsniff defaults
```

23.4.3 Configuring Network Equipment

In order for your collector to sniff any traffic, it must be located on the network in such a way that all internet traffic passes to it.

The simplest way to do this is to attache the collector to a hub (or other repeater) on the segment with the traffic you wish to monitor.

For switched environments, however, you may need to configure your switching equipment to forward packets to your collector port in order to analyze them.

For example, the following commands are the ones we use on our Cisco 4507R core switch:

```
monitor session 1 source interface Gi6/1 both
```

```
monitor session 1 destination interface Gi6/2
```

That tells the switch to mirror all traffic **from** port Gi6/1 **to** port Gi6/2 (where our AimSniffer sits).

At this point, you should start seeing traffic on the interface. Use a utility like `tcpdump` to confirm that you're seeing the mirrored traffic.

23.4.4 Testing

You should now be ready to monitor traffic. Start the collector service, and check to see if any messages show up in the database (select from the `logs` or `handles` tables for quickest results).

Chapter 24

SMDR (Phone Switch Call Accounting)

Last updated 2008/03/18

24.1 Introduction

Suffield Academy has its own PBX phone switch, and allows users to make direct local and long-distance calls from the phones on campus. We used to have a 3rd-party vendor who tracked and billed our long distance, but we stopped using them after we cancelled student long-distance accounts (the students were primarily using their cell phones).

In order to keep track of the basic call activity on our lines, we wrote a small script to parse the SMDR (**S**tation **M**essaging **D**etail **R**ecord) data that is output directly by our phone switch. It reads the SMDR data, parses it, and logs it to a SQL database for later analysis. Using simple SQL queries, we can run basic reports for total and per-number usage.

Our phone switch is a NEC NEAX 2400. The SMDR format is specific to this model of switch, but the general discussion of how it works should be roughly the same across other equipment.

24.2 Overview

24.2.1 Basic Operation

To log SMDR data from a phone switch, the following broad steps must be taken:

1. A SQL database (we use PostgreSQL) must be created to store the data.
2. One or more machines with serial ports must have our logging script installed on them (the script is written in Perl, and we use Debian Linux as our OS of choice).
3. The phone switch must be programmed to emit SMDR data on one or more of its serial ports (refer to the manual for the phone switch on how to enable this logging).

Once set up, the basic flow of information works like this:

1. The phone switch detects the termination of a call.
2. The phone switch emits Call Data Record (CDR) on all of its configured serial ports.
3. The machines running the logging script are waiting for serial input. They receive this input, parse it into individual fields, and insert the data into the SQL database.

Once in the database, administrators can run reports on the data using regular SQL queries.

24.2.2 Requirements

Our phone switch logs SMDR data over a slow (9600 baud) serial link. We do not experience a heavy call volume, so the amount of data is very small. Additionally, the phone switch automatically buffers the data on the serial ports, so in the unlikely event of the processing system being unable to keep up with the data flow, it will be buffered until the system can catch up.

Because the amount and rate of data is so small, almost any machine will do for processing the data. We run 3 Cobalt RaQ servers (200MHz MIPS processors) as our processing machines, though almost any machine will do (our previous processors were 486-class machines).

The script is designed to work with multiple processors in simultaneous operation. You can configure the phone switch to emit SMDR data on multiple serial ports, and then connect the ports to different machines for redundancy. The script will properly detect records that have already been inserted by another processor, and not duplicate the records. To help determine if the processors are running, a separate table tracks the records inserted by each processor so that you can ensure that all the machines are working properly.

The processing script is written in Perl, so you must have Perl installed on the processing machine. Additionally several common modules are required by the script:

- Sys::Hostname (built into recent versions of Perl)
- Time::Local (built into recent versions of Perl)
- Data::Dump (built into recent versions of Perl)
- POSIX ('setsid', should be available on POSIX-compliant systems)
- DBI (for inserting data to a SQL database)
- DBD::Pg (for communicating with a PostgreSQL server)
- Device::SerialPort (for listening to the serial port)

All of the modules, if not already built in to your system, are freely available on CPAN. On the "Etch" flavor of Debian Linux, all modules are already installed, or available directly via APT.

24.3 Running the Code

This section gives a (very) brief overview on setting up the processing systems, along with an even briefer overview of the code structure. Note that the script itself is well-commented, and should be easily changed by someone with a basic understanding of Perl.

24.3.1 Setting Up the Database

We have included a SQL file which defines the basic tables. It's written for the PostgreSQL database engine, though it could be made to work with any SQL-compliant database with very simple modifications. You can download the SQL here:

nec_smdr.sql

One table tracks all of the call data, and the other tracks the processing machines that inserted each record. To prevent duplicate records in a multi-processor environment, a UNIQUE constraint is placed on the call table. In the event that a processor attempts to insert a duplicate record, it will detect the duplicate on insertion and instead merely log that it too would have inserted a particular record. This information is stored in the "loggers" table, where the ID of each call record is listed with all the processors that processed the record.

24.3.2 Setting the Script to Run

First, download the actual script:

nec_smdr daemon Perl script

We suggest placing the script in `/usr/local/sbin`, though it can go anywhere. If you do change its location, be sure to update the init script below.

Download and install the package dependencies for the script.

Modify the script to include the correct serial port and database connection information.

Next, create the file `/etc/nec_smdr_postgres_password`, and put the database password inside it.

Next, download the init script, which starts and stops the daemonized Perl script:

nec_smdr init script

This is a Debian-based init script, and should work without modification on that platform. If you don't have Debian, modify it as necessary.

Link the init script into your `rc.d` directories as appropriate. Under Debian, the `update-rc.d` utility can help you with this.

To test the script, you can run it directly with the `-D` flag to run in debugging mode, *e.g.*:

```
/usr/local/sbin/nec_smdr -D
```

Instead of daemonizing, the script will run in the foreground and provide extra information as call records are processed.

If your serial port is hooked up, you should see debugging information about the calls as they are processed. Confirm that the calls are being logged to the database.

If everything is working correctly in debug mode, quit and issue:

```
/etc/init.d/nec_smdr start
```

To launch the daemonized version. The program is now running in normal mode and processing calls. You're all done!

==== Some Hints About the Script ===

The processing script has three major phases in it:

1. Wait for data on the serial line, and accumulate it in a buffer.
2. When a full record has been accumulated, parse it into individual fields.
3. Form the fields into a SQL record and insert it in the database (detecting duplicates if necessary)

If you need to modify the script to understand a different record format, you'll need to modify the record parsing section, and possibly the SQL section (if the number or names of the fields change).

The script has a very well-commented section on the call record format that it understands. Additionally, the script will log any lines it receives that it does not understand, so you can quickly debug the format of the lines.

If you're having trouble with the script, be sure to run it in debug mode (with the `-D` flag). This mode prevents the script from disassociating from the terminal, and provides much more verbose logging of its activity. It's very helpful for seeing what's going on during processing.

Chapter 25

Dance-A-Thon Scripts

Last updated 2008/03/18

25.1 Introduction

This document describes the scripts and procedures for running the technical aspects of Suffield's Dance-A-Thon. The Dance-A-Thon is a charit event put on by the students, faculty, and staff. The event runs all night (until early the next day), and involves a majority of the student body dancing at all times.

The event is a lot of fun, and generates many "media-worthy" moments. We typically take over 1,000 pictures and several short movies at the events. To help organize all of this information, we have created some computer scripts which sort, label, and upload these files to the school's web site.

Note: this document is intended primarily for Suffield staff preparing to use our software for the Dance-A-Thon. Others are welcome to use the software for other purposes, but we make no promises about its adaptability to other tasks.

25.2 Dance-A-Thon Overview

The Dance-A-Thon is a several-hour-long event involving dancing, live music performances, contests, and games. Suffield uses technology during the Dance-A-Thon to accomplish the following tasks:

- Downloading and processing digital photos taken during the event.

- Downloading and processing digital movies taken during the event.
- Streaming a live feed of the audio to other computers.
- Displaying "special features" (trivia contests, music videos) on a large projected screen.
- Displaying a "slide show" of pictures taken at the event so far.

Because of the frantic pace of the Dance-A-Thon, there is not a lot of time nor manpower to handle these tasks. We therefore have created a series of scripts and procedures to help keep things flowing.

25.2.1 Hardware

The Dance-A-Thon requires us to use several pieces of hardware, all of which must work together with each other, and with our software.

Computers

We've managed to hone our hardware requirements down to a single primary machine that does most of the work during the event. The machine has the following minimum requirements:

- Dual-monitor capabilities (two independent screens, not mirroring)
- At least 10GB of free space on the hard drive
- Mac OS X (script tested against 10.4.4)

We currently use a dual-processor G5 computer from Apple, with a video card supporting two DVI outputs.

The computer provides a live streaming feed of the audio to other computers on the network. If you wish to receive this stream and play it, you will need another computer. In this case, the requirements are modest; any machine capable of receiving a QuickTime streaming broadcast will suffice.

Still Cameras

For still photography, our scripts require that all photos have the same dimensions (they assume a statically computed size, to save on computations during

image processing). Therefore, it is best to use the same model camera if multiple cameras are desired.

The scripts also make use of EXIF header information to determin image orientation. Therefore, the cameras **must support** this orientation metadata for the scripts to rotate the pictures correctly. If the cameras lack this feature, the user must manually rotate photos before processing.

We currently use Nikon D70 cameras (several of the people shooting photos have this model camera). The images are consistent, and the camera supports image orientation data.

Video Cameras

For motion photography, we use a camera with the following features:

- Low-light or infrared photography support (the dance floor is not well lit, so we use a camera capable of shooting in very low light).
- Native support for recording into a digital file format (*e.g.*, MPEG)
- Direct video output in a format compatible with the projector (see below)

We currently use a Sony TVR-350 handheld camera. It supports a "nightshot" low-light mode, and can save recorded movies directly to 320x240 pixel 15fps MPEG1 movies.

The quality of the MPEG movies produced by the camera is substantially less than that of DV tape. However, the major benefit is that the movies do not need to be imported into the computer and saved into a compressed format; we simply upload the movies directly from the camera to our web server. Because the web site is the intended delivery mechanism, the file quality is acceptable for this use.

Projector

We provide a live "slide show" of photos from the event, updated throughout the eventing. Additionally, we show music videos, trivia contests, and other media via the primary computer.

Additionally, we sometimes directly attach the video camera to the projector to provide a live shot of the room for everyone to watch. Therefore, the projector should have an additional input that works with the camera (*e.g.*, S-Video or composite). Many projectors provide a simple means for switching between inputs.

25.2.2 Software

We use a few pieces of software to manage the files during the evening. Some of this software is available, and some we have create ourselves.

Photo Sifting

We have written a short perl script called `sifter` which takes a folder of raw photos, renames them by date, and sifts them into directories based on the time they were taken. The script then builds an HTML gallery out of the photos (creating thumbnails as necessary).

The resulting gallery allows web users to quickly see all the pictures, and view larger sizes as needed. Additionally, a direct link to a photo ordering service is provided for each picture.

The script requires the `jhead` program for extracting EXIF information from photos, and the `jpegtran` package for performing lossless rotations of photos. Both programs are freely available, and should compile cleanly on Mac OS X.

Additionally, the script requires the `sips` (scriptable image processing system) binary, which comes pre-installed on all versions of Mac OS X since 10.3. It provides extremely quick image manipulation functions, such as resizing photos. We use it to make thumbnails. SIPS could be easily replaced with other image manipulation software (such as ImageMagick or NetPBM).

Video Sifting

We have written a short perl script called `vidindex` which takes a folder of raw MPEG videos and builds an HTML index to the files. The script looks for an optional file for each movie that contains a timestamp, title, and short description. If the file is found, the script uses these values when building the HTML index.

Because there are not as many videos taken during the night, the video script does not take as complex an approach as the picture filtering script. All the videos are listed on a single page.

Web Syncing

Once the photos and videos have been sifted, a short script called `sync` may be called to upload the resulting galleries to our web server. The script acts as a simple wrapper around `rsync`, which provides a fast and efficient way to update only the files that have changed since the last syncronization.

We use a passwordless SSH key so that the user does not need to enter a password to update the web site.

Additionally, the `sync` script supports a local (or remote, if reconfigured) syncronization of the photos to a "slide show" directory, so that the slide show photos can be easily updated throughout the evening.

Live Streaming

We stream the audio portion of the dance-a-thon over the network to other computers. The primary use of this stream is for computers elsewhere in the building where students may be taking a break between dances. Because they can hear the feed anywhere, they can return to the dance floor if they hear a song or event they'd like to participate in.

We use Apple's Darwin Streaming Server, which provides a quick and easy way to serve streaming media. The project is open-source, and precompiled binaries are available directly from Apple:

http://developer.apple.com/darwin/projects/streaming/

To author the live stream, we use Apple's QuickTime Broadcaster package. This product is available without charge from Apple:

http://www.apple.com/quicktime/broadcaster/

Double-Clickable

As a final note, we have created several `.term` files, which are double-clickable under Mac OS X. These files simply launch the command-line programs with the correct arguments.

25.3 Usage

This section serves as a guide for staff running the Dance-A-Thon using our setup. Please excuse the informal nature; this is intended mostly as "notes to self".

25.3.1 Hardware Preparation

Set up a desktop machine (iMac or tower) with a local admin account (we use "suffieldacademy"). The software config currently assumes "/Users/suffieldacademy/Desktop"

as the base folder for the SVN checkout.

Install the developer tools and MacPorts on the target machine. You'll need to install the `jhead` and `jpeg` to get the `jhead` and `jpegtran` binaries.

For the Eye-Fi wireless SD card, you **must** use a non-enterprise AP. Local computer sharing won't work, and neither will multiple-AP setups where you can roam between points. Set up a dedicated AP and join the card to this.

Set the video camera to "nightshot" (not "super nightshot") mode.

25.3.2 Software Preparation

Check out the scripts from the source repository, and locate the folder on your target machine. If you have a second hard drive with plenty of space, you might want to locate the software there.

Edit the `GLOBALS.pl` file to have the correct paths, dates, and other variables for the scripts to run correctly.

If you want to do a dry run, try setting the year to "test" and running the scripts. There's a `picturefaker` script which will pave over the EXIF dates of a folder of pictures (faking them so they appear to occur during the Dance-A-Thon that is yet to happen), so you can try the scripts out.

On the web server, create the subfolder for the current year, and create an HTML file in the top level with the current year (just copy an existing one and modify). Symlink it as "index.html" in the year folder.

While you're on the web server, edit the config file for the virtual host that contains the Dance-A-Thon files. There should already be a configuration section; just edit what's there (if there isn't, search this repository for a vhost config that should have the settings).

The apache files aren't strictly required, but they help with a few things:

- Cache-refresh settings (mainly for interaction with proxies and Coral Cache).
- Forced-download settings for full-size pictures.

  Also, you may consider editing the `sifter` script and commenting out the Shutterfly order feature. That way, you can post the pictures immediately, but not worry about messed up filenames from interfering with orders. Once the Dance-A-Thon is done, you can confirm that the files are named correctly, uncomment the icon, and issue one more "build" command on the gallery.

  At this point, you should have a functioning upload system, and you're ready to go!

25.3.3 Dance-A-Thon Preparation

On the night of the Dance-A-Thon, things are pretty straightforward. Here's a basic checklist of functionality that should be working before the event starts:

- `sifter dirs` to create the directory structure
- `sifter build` and `vidindex`, which will create the empty directories. Upload these empty directories so there's something to see (instead of a 404) before you put up any pictures.
- Fire up the streaming software, and test the feed from the DJ console to a remote machine. Check the levels!
- **Check dates and times on all cameras**. The scripts rely heavily on the EXIF dates provided by the cameras. If these dates get out of whack, the pictures will appear in the wrong place, or might be rejected outright by the script (the script ignores grossly inaccurate dates). You can fix bad dates with `jhead`, but it's a pain, so it's easier to make sure everything is set before you start.
- Confirm dual-monitor setup on any applications you'll be using on the computer (*e.g.*, Keynote, QuickTime Player). You'll want to set the application's "full screen" mode to use the projector screen, leaving the main screen free for other tasks.

Once you've run through this checklist, you should be good to go. Drink plenty of water, and settle in for the night!

25.3.4 After the Dance-A-Thon

When it's all over, make sure you've downloaded all final pictures, indexed them, and uploaded the results. If you disabled Shutterfly links, confirm that the photos are correct, and then rebuild the gallery with Shutterfly support.

Once you're **certain** that the galleries are correct, re-edit the Apache configs and turn up the cache expiration time.

Part III

Servers

This section contains documentation regarding the physical servers and network equipment used at Suffield Academy. The documentation includes hardware specifications, essential system components, and a brief overview of installed software for each machine.

Chapter 26

Firewall (OpenBSD PF)

Last updated 2008/03/18

26.1 Introduction

Like most networks on the Internet, we have a **firewall** that acts as a boundary between our campus network and the Internet at large. We do this for the purposes of security (preventing hosts from connecting to us that should not be allowed), and also to perform router-type services like bandwidth control and network address translation.

Rather than using specialized hardware for a firewall, we use commodity PCs running the OpenBSD operating system. OpenBSD has a long track record as a secure OS with excellent support for firewalling operations. The firewalling code in OpenBSD is called **PF**, for "packet firewall".

As an added bonus, OpenBSD has a feature called CARP (**C**ommon **A**ccess **R**edundancy **P**rotocol), which allows for hot-standby failover machines. This helps reduce downtime if a firewall needs to be rebooted.

26.1.1 Network Diagram

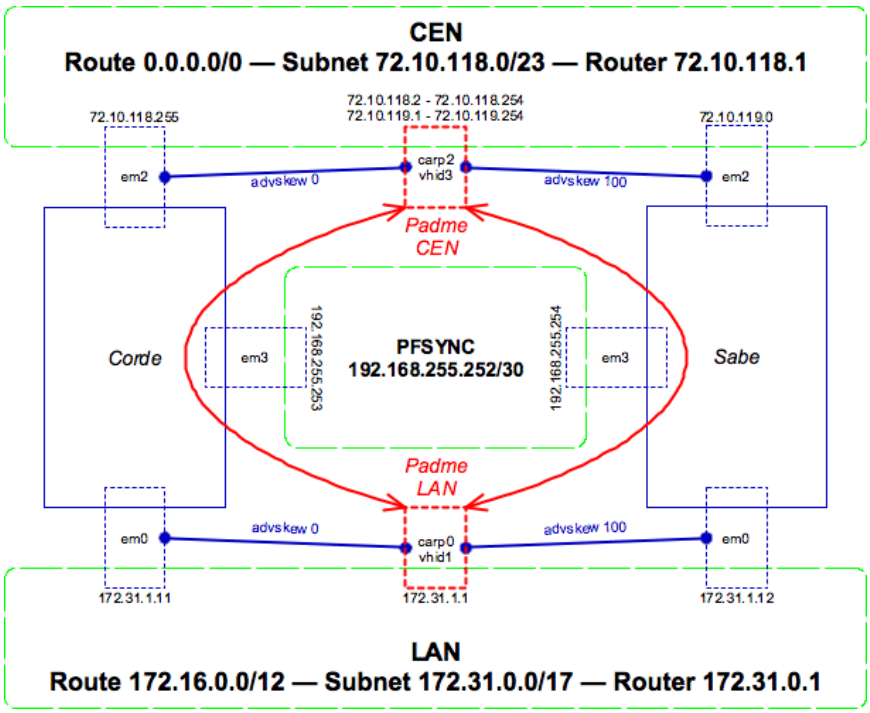
Suffield has a primary connection to the internet through the Connecticut Education Network (CEN). We are provided two class-C networks of routed addresses from CEN. Suffield's internal LAN uses RFC 1918 private addressing, and all of our machines live in this space.

We have two firewall machines: Corde and Sabe. Each has its own connections

to our LAN, WAN, and a crossover connection to the other machine.

Together, these machines form a failover pair to a "virtual" firewall named Padme. It has its own IP addresses distinct from the physical machines that answer for it. To an outside observer, we appear to have a single firewall (Padme), even though there is no physical machine with this name. In reality, we have two physical machines, and one of them is pretending to be Padme at all times (which one doesn't really matter, as they're hot standbys for each other).

The following diagram shows the two machines, their interfaces, addresses, and how they create a virtual firewall:



- The green boxes represent networks (such as our LAN and CEN)
- The blue lines represent physical machines and their connections
- The red lines represent virtual "machines" and their connections
- The dashed lines show network interfaces on the various machines

Essentially, each firewall has its own IP address on each of its physical interfaces (em0, em2, em3) on each network. Additionally, the firewalls each advertise "virtual" IPs on the network on their virtual "carp" interfaces. Only one machine advertises at a time, and the CARP protocol deals with the details of which

machine is the master and which is the backup (the "advskew" parameter determines who is the master by default; in this case Corde is because of the lower value).

The active firewall sends state changes and updates to the backup machine via the "pfsync" network between the two machines. This network is just a direct cable attachment; it is not routed or accessible from outside of the two machines. Should the primary firewall go down, the backup will notice (via CARP) and take over where the other left off (using the state information sent via pfsync).

Note that the private PFSYNC network is only necessary to exchange firewall rule state information, not the failover of addresses. CARP works directly over the real networks (on interfaces em0 and em2 above), and doesn't need an out-of-band channel to work. However, the firewall state information does.

For more information on CARP and PF, please see:

- http://www.openbsd.org/faq/pf/carp.html
- http://www.countersiege.com/doc/pfsync-carp/

26.2 Installation

26.2.1 Hardware Requirements

OpenBSD installs on a wide variety of hardware, and works very well even on "underpowered" hardware. That said, certain hardware choices can help performance.

As our machines are firewalls, network throughput is important. The OpenBSD community appears to agree that gigabit cards perform best (even for small workflows) due to their more advanced features (TCP checksum offload, larger buffers, DMA, *etc.*) We selected Intel dual-port server NICs for our machines (2 onboard, 2 PCIe, for a total of 4 ports per machine). Remember, you need one "inside", one "outside", and one "sync" interface at a minimum.

The firewall code runs inside the OpenBSD kernel, which is not multi-processor aware. Therefore, a dual-core machine doesn't help us. We bought the fastest-clocked single-processor machine we could find.

Hard drive space is almost negligible (600MB or so for the OS install, plus space for swap). PF keeps everything locked in RAM, so speed of the drive is not a huge issue. We did not opt for RAID, because we have two fully redundant machines (which would make a redundant drive, well, redundant...).

26.2.2 Install OpenBSD

If you're a seasoned pro at OpenBSD, just grab the ISO image online, burn it to a CD, and pop it in your machine. If you're less-than-seasoned, I recommend purchasing a full CD set from OpenBSD. The money goes to a good cause, and the sets come with full installation instructions.

You'll want to install the base system and any compiler packages. You do not need to install "games" or the "X11" system.

Set up the primary interface to be on a network that can reach the Internet (we used our LAN as the primary network and interface).

Once the install is complete, you should be logged in as `root` and able to reach the internet from the machine.

26.2.3 Installing Packages

Before you start making configuration changes, you may find it helpful to download a few more pieces of software on the server. OpenBSD has a ton of precompiled packages you can install directly. If you're running on the i386 platform, here is the list for the version of OpenBSD we're using right now (4.3):

http://www.openbsd.org/4.3_packages/i386.html

Simply copy the download URL to the `.tgz` package, and paste it into the following command to automatically download and install the package:

```
pkg_add <URL to package file>
```

You'll need Subversion in order to check out our config files from source control and use them on the machine:

- http://www.openbsd.org/4.3_packages/i386/subversion-1.4.4.tgz-long.html

I like to install Bash (my shell of choice) and Emacs (my editor of choice) on any machine I use:

- http://www.openbsd.org/4.3_packages/i386/bash-3.2.33.tgz-long.html
- http://www.openbsd.org/4.3_packages/i386/emacs-21.4p6-no_x11.tgz-long.html

pftop is a great utility that shows you state information, utilization, and other statistics. pfflowd turns your state information into Netflow-compatible exports, which you can graph with other software:

- http://www.openbsd.org/4.3_packages/i386/pfflowd-0.7.tgz-long.html
- http://www.openbsd.org/4.3_packages/i386/pftop-0.7.tgz-long.html

Install Subversion plus any other packages you're interested in.

26.2.4 OS Tweaks

To prepare the machine, we edit several of the OS configuration files to our liking. All of the files we edit are checked into source control, and so are available. You must have installed Subversion (see above) before you can get these configuration files.

Move to the root of the hard drive, and check out our config file directory:

```
cd /
svn co svn://svn.suffieldacademy.org/netadmin/trunk/servers/pf/etc.suffield
```

Some of these files can be used as-is, while others should be copied into place and then modified.

Static Config Files

You'll want to move the following files out of the way (rename with with ".bak" at the end of their name, or something similar). The files are all located in `/etc`. You'll replace them with symlinks to the files of the same name in the `/etc.suffield` directory.

- mygate (define the router, if not already set up properly)
- resolve.conf (use our name servers)
- syslog.conf (use our local log server)
- newsyslog.conf (add our log rotation rules)
- ntpd.conf (use our local time servers)
- snmpd.conf (be sure to edit and define the password and community string)

Additionally, you'll want to do the same with the following OS config files. These files set defaults for how the operating system behaves, enables the firewall, and tweaks some performance values. All files are commented, so you can find the values we've changed from the defaults:

- sysctl.conf (twiddle kernel configuration knobs for performance)
- rc.conf.local (define options for services that start on the machine)
- rc.local (local startup of daemons installed by us)

26.2.5 Network Configuration

In addition to the files above, we must copy in some "template" config files and then edit them to match the machines they're being installed on.

Refering to the diagram in the Introduction, you can see which IP addresses should be assigned to the various physical interfaces. Copy the `hostname.XXX` files into `/etc/`, where "XXX" is the name of a **physical** interface (*e.g.*, `em0`). Depending on the driver for your NIC, the names may vary. Our Intel adapters use `em` as the prefix, but other prefixes (such as `sk` and `dc`) are also common.

Edit these files to have the correct information. In most cases, we've included comments that tell you what to change and what to leave alone. The interfaces are:

- hostname.em0 (LAN)
- hostname.em2 (CEN/WAN)
- hostname.em3 (PFSYNC)

Note that the LAN interface has route statement added to it. This ensures that packets destined to our LAN get routed "inward" over that interface. Because the default route matches all packets and sends them over the WAN interface, we must add this more specific route to connect to our internal hosts.

Once you've enabled the physical interfaces, you need to activate the virtual interfaces. Copy over the following configuration files, and edit the information in them to match the layout of the host. Assuming the physical interfaces are numbered and named the same as in our setup, the only thing you'll need to chage is the password and `advskew` parameters on each CARP interface. The password should match between the two firewalls, and `advskew` should be 0 on the "primary" firewall and a higher number (we use 100) on the "backup" firewall.

The following config files define the virtual interfaces:

- hostname.carp0 (LAN virtual adapter)
- hostname.carp2 (CEN/WAN virtual adapter)

- hostname.pfsync0 (pfsync device to sync firewall states)

Note that the CARP interfaces define the addresses that the rest of the world should use. In the case of our LAN interface, we have the single address that we use for routing traffic. On the WAN side, the CARP interface advertises all IPs on that interface where we receive traffic (the firewall rules worry about routing it back to machines on the inside via NAT).

26.2.6 Finishing Up

At this point, all the configuration files should be in place for a complete install (including those for our firewall ruleset). Reboot the server:

```
shutdown -r now
```

To enable all the new interfaces and startup config options. **Note:** if you are not physically at the machine, be aware that the reboot will enable the firewall ruleset from source control, and you might be cut off from the machine. Either perform the reboot while physically at the machine, or disable the firewall rules.

26.3 Firewall Settings

The OpenBSD PF code supports many advanced options, turning it into much more than just a firewall. The system can perform other functions, such as routing, redirection, statistics tracking, queuing, and packet normalization.

If you are not familiar with PF, stop and read one of the many documents describing its use, such as the OpenBSD PF FAQ. The remainder of this document assumes you have a working knowledge of PF.

You can download the Suffield pf.conf file to see how we've done things. The file is well-commented, and should be easy to follow if you're used to PF config files. We've highlighted a few features of the configuration below.

26.3.1 Normalization

We use OpenBSD's "scrub" directive to fix up the packets before we process them. This includes fragment reassembly, window cropping, and other sanity checks on the packets (and is generally considered a good idea). We don't apply some of the more esoteric options, as they can cause problems with certain quirks in the TCP/IP stack on some machines.

We also set immediate block rules for "evil" packets (sourced from bogus IP space), and perform checks to ensure that packets with our private IP space only appear on our internal (LAN) interface.

26.3.2 Policy queueing using tags

We have a traffic shaping device on our network, so we do not perform major shaping on the firewall. However, we do set a basic cap to ensure that the traffic level stays below a reasonable level.

Our queueing rules are complicated by the fact that we have two basic tiers of service: "on-net" traffic (which can flow without limit at no extra charge to us), and regular "transit" traffic that must not exceed a threshold.

Ordinarily, we would need to duplicate all of our rules (one for each class of traffic) in order to assign packets to the correct queue. However, PF allows us to "tag" packets as they pass through the firewall, and then act on those tags later on in the ruleset.

You will note that all normal "pass" and "block" rules in our ruleset also assign a tag of "$accept" or "$reject". We can then refer back to this tag at the end of the ruleset and perform queue assignment as the last step (after all of the policy decisions have taken place).

We do not queue inbound traffic at this time, because it interferes with our shaper's policy decisions. Because the firewall comes before the shaper for inbound traffic, turning queuing on for inbound packets on the firewall would have the effect of "pre-empting" the shaper and limiting packets arbitrarily (rather than according to our shaper rules).

26.3.3 Static NAT for servers

Though all our machines live in our private LAN IP space, we do want some of them to be reachable from "the outside". These machines (servers, typically) live in our "DMZ" VLAN. The firewall ruleset defines a NAT mapping from a live external IP to the private internal one (for certain ports), and ensures that any outbound traffic is steered to the correct public IP.

Additionally, the policy section of the firewall rules permits traffic to these same designated ports. For a handful of servers, we restrict access by IP, but most have a global permit for the ports we want others to reach.

26.3.4 Round-robin NAT for users

All non-server machines are forwarded through the firewall and NATed to an arbitrary external IP address (taken from a pool of available addresses). The assignment is "sticky", ensuring that clients are NATed to the same address during a session.

26.4 Reloading the Firewall Ruleset

The firewall rules are loaded in `/etc.suffield/pf.conf` (kept in version control). When you need to make changes, use the following procedure:

1. Edit the firewall ruleset on your local machine in version control.
2. Check in and comment the changes
3. Log on to the firewall
4. Move into the `/etc.suffield` folder
5. Run `svn up` to get the latest version
6. Test the ruleset (without loading) by running:

   ```
   pfctl -nf /etc.suffield/pf.conf
   ```

7. If there are no configuration errors, reload the ruleset:

   ```
   pfctl -nf /etc.suffield/pf.conf
   ```

Note that you'll need to reload the rules on both the primary and backup firewalls. In this case, it's best to load the rules on the backup firewall first, so that any serious errors will not cause a disruption in service on the primary system. Once the rules have loaded successfully on the backup system, log on to the primary system to fetch the latest version and reload.

26.5 Tips and Tricks

Below we list some helpful commands to know about when monitoring the firewall. Some may require installing additional software, usually through the package or ports interface built in to OpenBSD.

26.5.1 pftop

pftop is to firewall information as "top" is to process information. It's a simple utility that shows a live snapshot of the rules on your firewall, the active states, queue statistics, and bandwidth usage.

pftop is installable through the package system in OpenBSD, and so is very easy to get started with. Once installed, just type `pftop` and you'll immediately be taken to the default screen. Use the left and right arrows to move between reports (listing rules, states, queues, *etc.*). Hit "q" to quit.

26.5.2 tcpdump

PF has a "log" keyword for its rules, allowing you to log packets that match those rules. By default, logged packets go to a special device called `pflog0` (you can specify other log devices in the PF configuration file to log certain events elsewhere).

To read this log file, you use `tcpdump`, just as if you were sniffing packets off the wire. We like to see timestamps and other information, so a sample command might be:

```
tcpdump -levvvttti pflog0
```

You can specify other options that tcpdump likes (such as `-n` to leave names unresolved), and you can specify filters to limit matched packets to only those you're interested in.

We usually run this command immediately after loading a new PF ruleset to ensure that we aren't blocking traffic that we used to allow.

26.5.3 Label Statistics

Our firewall ruleset makes uses of "labels", which are a statistics collection tool. Anywhere we define a label in the config file, OpenBSD collects statistics for packets that match the rule and assigns them to counters associated with that label.

You can view the counters for all labels by running the command:

```
pfctl -vs labels
```

This can be inspected by hand, or you can feed it into a script for graphing or other reporting.

26.5.4 Inspecting PF Config

In addition to the label statistics shown above, the `pfctl` utility can show all kinds of information about the currently loaded firewall rules, queuing, state table, and statistics. Simply run `pfctl` with the `-s` flag and a section you wish to see information about (such as `rules` or `states`). Adding a `-v` will give more information, and usually also include detailed packet counters.

For full information, refer to the pfctl manual page.

26.5.5 Killing States

By default, PF "keeps state" on all accepted connections so it doesn't have to evaluate the entire ruleset for each packet (packets that match an existing state are passed immediately). However, if you make a change to the ruleset and wish to have it take effect for all connections immediately, you'll need to kill existing states.

You can do this for specific hosts or networks by using the `-k` option to `pfctl`:

```
pfctl -k 192.168.0.1
```

That kills all states sourced by the given IP address. If you specify `-k` twice, it will kill all states from the first host to the second. You can use hosts or networks to kill multiple states:

```
pfctl -k 192.168.0.1 -k 10.10.10.1
pfctl -k 192.168.0.0/24 -k 10.10.10.2
pfctl -k 0.0.0.0/0 -k 192.168.0.3
```

Note: killing a state may cause an existing connection to be interrupted, and the end user may need to reload or reconnect their sessions. Only kill states when it is absolutely necessary to do so; otherwise, just let the states expire on their own.

Chapter 27

NEAX-2400 (PBX / Phone System)

Last updated 2008/03/18

27.1 Introduction

Suffield Academy has private branch exchange (PBX) system to facilitate on-campus and external phone calls. We provide dial tone to all student and faculty residences, and also have phone service available to administrative and department offices.

This document describes the basic setup for our phone network, including a general network description, numbering plan, and programming instructions. Finally, a quick reference for end users of the phone system is provided.

27.1.1 Network Setup

Suffield Academy has a block of 440 DID (direct inward dial) extensions that we "own". Outside callers may dial (860) 386-XXXX, where XXXX is one of our 4-digit extensions. Currently, our DID allocations are:

- 4400 - 4799
- 1021 - 1060

Internally, we use other extensions outside this range (4000 - 4399 and 4800-4999). While these extensions can make outbound calls, they cannot be dialed directly without being transferred by the switchboard. We keep these numbers for secondary extensions or outward-dial-only devices (automated alarm dialers).

All of these DID numbers are trunked via 2 PRI (primary rate interface) circuits from PAETEC. A PRI is like a T1, but one channel is used for signaling, and is capable of routing a large number of numbers over a smaller number of physical channels (in other words, we only need 46 channels for our 440 DIDs, because the PRI can route and signal the calls on any available channel). The two PRI T1s are sourced from different offices to provide some redundancy. The circuits are:

- Circuit ID: HCGS.748744 / 20.HCGS.301043.110 (route "10" in the PBX)
- Circuit ID: HCGS.748731 / 20.HCGS.301044.110 (route "20" in the PBX)

The circuits have the following numbers assigned to them:

- (860) 752-6526
- (860) 752-6527
- (860) 752-6528

These numbers are not routed or listed, but they do come up as the ANI (Automated Number Information) if you request a trace on the circuit. In general, our PBX should provide correct caller ID for our block of DIDs, so you shouldn't see these numbers popping up anywhere.

In addition to these digital lines, we also have 2 POTS (plain old telephone service) lines from AT&T. These are the school's older published phone numbers, and so do not share our 386 prefix. We have a Central Office Line Card (PA-16COT-BE) that these lines connect to. The numbers are:

- 860 668-7315 (main school number, route "1" in the PBX)
- 860 668-2966 (main school fax, route "2" in the PBX)

  In the event that both PRI interfaces are unavailable, outbound calls will fail over to these analog lines.

We provide "off-premise" extensions to phones outside our school-owned network by renting "dry pairs" (or alarm pairs) from AT&T. These lines connect to our PBX, but pass through AT&T's network to reach buildings off campus that we do not provide direct service to. Those lines are:

- Circuit ID: LCFL.882033 [12,1703,3] (physical plant, Phil Cyr)
- Circuit ID: LCFL.882039 [12,1711,11] (physical plant, Vinnie Bottone)

  Those lines are given on-campus extensions, and in all respects appear to be on-campus analog phones. The only difference is that we don't own the copper connecting the phones to our PBX.

Finally, we also order direct service from AT&T for other buildings not served by our network. These lines do not have 386 extensions, as they are not direct-dial extensions. They simply provide basic service to locations we cannot reach:

- 860 668-1687 (Varnum Physical Plant Building)
- 860 668-2106 (Day Care)
- 860 668-5527 (SOLO barn)

27.1.2 PBX Setup

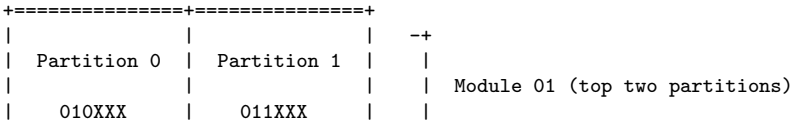
Our PBX is an NEC NEAX-2400 IMX. It serves all on-campus phones, and routes off-campus calls via the PRI interfaces (or, if they are unavailable, any analog lines). The PBX is also capable of many advanced features (caller id, hold, transfers, conference calling, *etc.*).

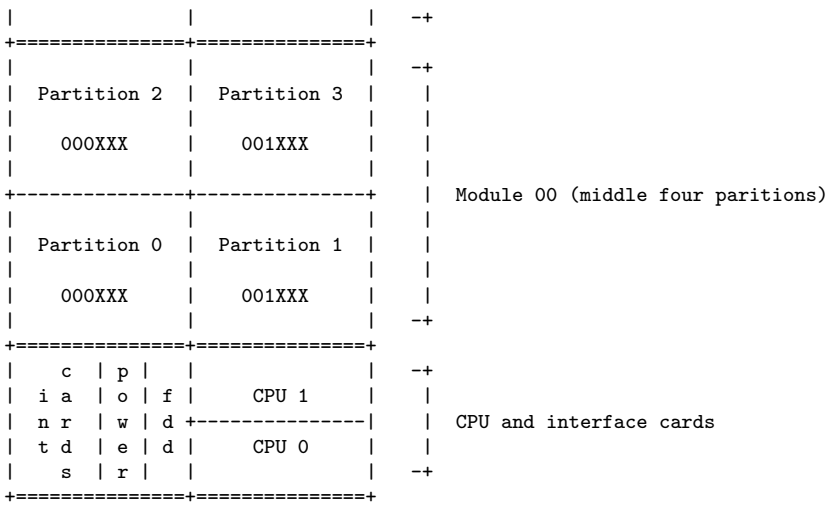
The unit itself is located in the school's main server room. The system is modular, and is capable of having additional cards and cabinets added to it over time.

Every line that connects to the PBX is assigned a **L**ine **E**quipment **N**umber, or **LEN** (or **LENS**). These numbers uniquely identify every station on the phone network, and are how the PBX "knows" which phone number is attached to which piece of equipment.

The LENs are assigned according to the location of the card in the PBX that is connected to the line. Our LENs are six digits long, and are comprised of the **module**, **partition**, **card**, and **line** that serve the particular station.

To determine the LEN, you must determine each of these numbers and concatenate them. If you open the doors to the PBX, you'll see something like the diagram below:





The PBX currently has 4 doors. The bottom door contains the two CPUs for the PBX and the interface cards (serial lines to the MAT, voicemail, and SMDR accounting). The door above that contains the bottom half of module 00 (partitions 0 and 1). The door above that contains the top half of module 00 (partitions 2 and 3). Finally, the topmost door contains module 01 (partitions 0 and 1).

These module and partition numbers form the first 3 numbers of the LEN. They are listed in the diagram above. For example, the top left partition's LENs all start with "010", because they are in the top module (module 01), and on the left side (partition 0): 01 + 0 = 010.

In each partition there are several **line cards**. Above each card there are numbers which determine the group number(s). Sometimes there are 2 group numbers (such as "4 . 5"), sometimes there are 4 group numbers (such as "2 . 3 . 29 . 30"), and sometimes there is a range (such as "12 ˜ 15"). The numbers refer to the groups of 8 lines. If there are 2 group numbers, then the card can have up to 16 lines (2 * 8). If there are 4 group numbers, up to 32 lines are possible, but usually only 16 are used (the higher two numbers are discarded). For a range, 8, 16, 24, or 32 lines are available, using numbers from the top of the range first.

The lines themselves are numbered from 0 to 7 in each group, starting from the bottom of the card.

Some examples are in order. Suppose we have a 16-line card in a slot marked "4 . 5". There would be 16 lines total, with LENs ending with 040, 041, 042, 043, 044, 045, 046, 047, 050, 051, 052, 053, 054, 055, 056, 057. Note that the group number (4 and 5) is simply prepended to the line number (0 - 7) to make the final part of the LEN. In the case of single-digit group numbers, a zero is prepended.

Suppose now that we have a 16-line card in a slot marked ("2 . 3 . 29 . 30"). There are still only 16 lines total, and the prefixes "29" and "30" are ignored. The LENs would end with 02(0-7) and 03(0-7).

Finally, suppose we have a 24-line card in a slot marked "12 ˜ 15". There are 24 lines total, with prefixes 13(0-7), 14(0-7), and 15(0-7). Note that the prefix "12" is unused, as we start from the top of the range (15) and work downwards. Since there are only 24 lines, we don't need the 12 prefix.

To calculate a complete LEN, you concatenate all of these numbers together. For example, a 24-line card in the top-right cabinet in slot 12 ˜ 15 would have LENs in the range 011130 - 011157. You start with the module (01), add the partition (1), then the line card (13, 14, 15, as per the example above), and then the line number (0-7). **Note**: the last digit in a LEN can only be 0-7, as there can only be a total of 8 lines per group.

The cards themselves come in many sizes and types. Cards with a model number starting with "EL" are digital (capable of supporting NEC Dterm phones), while "LC" cards are analog (work with standard telephones from any vendor). These in turn come in 8, 16, and 24-line sizes. Any other cards in the PBX are specialty cards (PRI, T1, CPU, and other control cards).

27.1.3 System Interfaces

The PBX interfaces with several other systems for control and logging purposes. All of these systems interface via a serial connection (DB-25 or DB-9). There are three main types of connections:

1. MAT (control terminal, for programming the PBX) on ports 0 and 5
2. MCI (Message Center Interface, or voicemail) on port 1
3. SMDR (logging of call accounting data) on port 2, 3, and 4

Note: the SMDR ports buffer data if they are not connected to an output device. If an SMDR station fails and you won't be reconnecting it for a while, you must short the pins to prevent the buffers from filling up. To short the connection on a DB-9 interface, connect the following pins together:

- Connect pin 2 to pin 3 (SEND)
- Connect pin 4 to pin 6 (RECEIVE)
- Connect pin 7 to pin 8 (CTS/clear-to-send)

That should discard data on the interface until a permanent connection is available.

27.2 The Frame

27.2.1 Server Room

The frame is where all of the wires from the PBX are connected to the wiring that leads to the individual stations (possibly via other cross-connects in other buildings).

The frame is comprised of numerous 25-pair "split 50" 66 block connectors. Each block has 50 rows with 4 columns of connectors. In every row, the two left connectors are electrically connected, as are the two right connectors. However, the left and right halves are **not** connected together.

Standard punchdown for a station requires two sets of connectors: either the left or right pair of connectors in two sequential rows. We list the LEN for the station on the top row, and the associated station number (*e.g.*, extension) on the bottom row. For standard white-blue wiring, the white wire (tip) goes on top and blue (ring) on the bottom.

Thus, a sample block might look like the diagram below. The left side shows the label and type of wiring; the right side shows sample numbers as they might appear on the frame (1047 would be LEN 001047, and 4401 would be extension x4401).

```
LEN label (White/tip) |   A   B   C   D   | 1047
STN label (Blue/ring) |   A   B   C   D   | 4401

A = punchdown from PBX        (left side)
B = punchdown to jack/station (left side)
C = punchdown to jack/station (right side)
D = punchdown from PBX        (right side)
```

When connecting wiring, you will attach the wires to post "B" for the left-hand side and post "C" for the right-hand side. Posts "A" and "D" are already connected to the PBX and should not be used.

27.2.2 Connecting Station Equipment

Wiring is cross-connected from the frame, though a series of equipment boxes to get to the building where the equipment resides. From there, it is cross-connected again to the horizontal cabling in the building, and thus to the individual jack where the equipment will be connected.

In order to connect equipment to the telephone system, the following procedure should be followed:

1. Activate the extension and LEN in the PBX using the MAT programming software (or, identify an already-active LEN/extension). You will need the LEN to find the cable pair on the frame.

2. Identify the master cable pair (*e.g.*, pair 537), which corresponds to one of the 110 block cabinets to the right of the frame. For existing numbers, the pair should be listed in the FileMaker phone database. If not, you will need to trace an existing wire (if the LEN/station is already cross-connected), or find the cable pair by building.

   To find the cable pair by building, you need to know the cable pair in the 110 block for that specific building (usually, a number between 1 and 100). This is where the physical jack (*e.g.*, 34A) is punched down, and should be clearly labed with the jack number and the pair number.

   Once you know the local building pair (between 1 and 100), you can find the master cable pair. The master pairs run from 1 to 1200, with a contiguous block assigned to each building (the building ranges are marked on the 110 block cabinets). Simply correspond the start of the building block with the number from the building to find the actual master number. For example, Memorial Hall has the master cable pairs 26-100, and building cable pairs 1 - 75. So Memorial's building cable pair 1 corresponds to master cable pair 26, 2 to 27, 3 to 28, and so on.

3. Cross-connect the master cable pair to the frame using UTP wiring. White wire goes on the top (66 block) and left (110 block), and blue goes on the bottom/right.

4. If necessary, cross-connect the 110 block to the jack in the building where the equipment should be connected (again, most building pairs are already terminated, so this is only necessary for jacks that don't already have service).

You should now have service on the desired jack.

27.3 Programming

Some system changes only require wiring to accomplish (*e.g.*, switching an extension to a new jack). However, most other changes require programming the phone system. This includes assigning new extensions, changing phone type (digital/analog), changing the name display on the phone screens, and many others.

Programming the PBX requires the IMX MAT software from NEC. This software is installed on a machine with a direct serial connection to the PBX. The commands are somewhat arcane, so this section describes frequenly-used commands for common tasks.

27.3.1 Terminology

Several of the commands share input values. Some of these values are always the same (for example "tennant" is always 1), and others use one of a pre-set range of values. We discuss these below.

Restriction Class (RSC)

Our PBX has been set up to allow or deny certain features based on the restriction class of a particular number. Historically, this was used to prevent students from using their phones at certain times of day (study hall and lights out), but these restrictions are currently not used (students can dial at any time).

Restrictions can also be used to allow certain phones to dial numbers, require a PIN for long distance, or perform other tasks. The following table lists the restriction classes in our PBX:

RSC	Description
5	Admin (default/PIN required for long-distance)
6	Admin (no PIN for long-distance)
7	Student (enabled - normal operation)
8	Student (default - emergency calls only)
9	Dorm Faculty (no PIN for local calls)

Most lines will either be 5 (admin phones) or 7 (student phones).

Note: RSC 8 ("student default") **is not currently used**. This restriction class does not allow any calls, except to 911. This class exists for lights out and study hall, but we no longer have the policy of restricting calls during this time.

Service Feature Class (SFC)

The service feature class limits the type of service that a particular phone receives. For example, digital phones support putting calls on hold, transfers, and conference calls. The following table defines the service classes for the types of phones we use:

SFC	Description
3	Admin (digital phone)
4	Admin (analog phone)
6	Student (enabled - normal operation)
15	Student (default - emergency calls only)

Most lines will either be 3 (admin phones) or 6 (student phones).

Note: SFC 15 ("student default") **is not currently used**. This class is a holdover from when we restricted calls during study hall and lights out.

Phone Type (TEC)

Our PBX supports two kinds of phones: standard analog phones (any make or model) and digital phones (NEC Dterm). The digital phones support many additional features (*e.g.*, multiple lines, conference calls, name display).

When a station is created, the type of the line must be specified. One of the following two values must be used:

TEC	Description
3	Analog
12	Digital

Station/Extension (STN)

The **station** is the primary extension of the phone (or, for analog phones, the only extension). We use "station" and "extension" interchangably in this documentation.

Tennancy (TN)

On our system, the **tennant** is always **1**.

Automatic Number Identification (ANI / CPN / Caller ID)

Our PBX connects to the phone company's central office (CO) via a PRI, which is a type of ISDN digital line. As such, our system is more like its own central office than an end customer of the phone company. When our system places calls, it sends out ANI data via the digital line to the CO informing the CO of the station that placed the call.

Because we use "internal-only" (non-DID) extensions, we have to be careful not to send out ANI data for numbers that we do not own. By default, the PBX will send out 860386XXXX, where XXXX is the extension number. We must configure the PBX to override this behavior when we want to send out a different number.

To make matters more confusing, there are 3 types of "caller ID-ish" info that get sent out:

- ANI (Automatic Number Identification) is used internally by the phone company to determine the owner of the circuit. In our case, ANI is the number of the digital circuit we use (8607526526 from PAETEC). This number isn't even dialable, but it does indicate the owner.
- CPN (Calling Party Number) is what our PBX has control over. This number is sent out as a data packet when you place a call, and the receiving equipment can see what number you're calling from. This number can be "spoofed" by the PBX (for example, to map outgoing calls back to a main number). This is what we use on our "private" extensions.
- e911 data is stored in a database with the phone company. It provides additional information (ALI, or Automated Location Information) to emergency dispaters based on the CPN provided by the switch (**not** ANI). We provide a spreadsheet to the phone company with address, floor, and room data associated with each DID so the dispatcher knows where to send help.

You can call 1-866-MY-ANI-IS to hear an automated recording of your CPN. Despite the name, it really is your CPN (ANI is only reported if CPN is unavailable).

If you want to hear true ANI info, call MCI at 1-800-444-4444. The voice should read you the ANI unless you have specific caller ID information sent via the phone company.

See the **ACNP**, **ACND**, and **ACPNCL** commands below for information on setting the CPN.

27.3.2 Commands

Backup Phone Database

Changes are held in RAM and will not survive a full system reset unless they are stored to the hard drive. Whenever significant changes to the system have been made, you should back them up.

From the MAT software, choose **System Backup** -> **MEM_HDD** (Data control between Memory and HDD).

Choose **Memory to Hard Disk** as the direction.

Check all the boxes for the data that have changed. We don't use **Wireless Call Data** or **ACD Data Memory**, so leave those items unchecked.

Some data are associated with a specific **Line Processor** (LP), and you must enter the number of the LP to back up the data, which is **0** (zero).

Choose **Start**. The process will begin, and a timer will count. Each individual checkbox will cause a separate process, and will be shown in the log menu. When **all** of the processes have finised, the screen will un-grey (accept input) and you can return to the main menu.

Display Station Data (DSTN)

If you know a station number (extension), you can find out all associated information (such as LEN, classes, key assignments, *etc.*). Simply enter the station number and click **Get** to fetch the associated information.

If you want a listing for a range of stations (instead of just a single number), use **LSDT** instead to generate a report.

Display of LENS Data (DLEN)

If you know a LEN, you can find all the associated information (such as extension/station number, phone type, and classes).

Simply input the full 6-digit LEN and you'll get back all the associated information.

If you want a listing for a range of LENS (instead of just a single one), use **LLEN** instead to generate a report.

Assignment of Name Display Data (ANDD)

Use this command to change the name display that screen phones will show for incoming or outgoing calls.

Enter tennant and station. If a name is already assigned, you must delete it first before entering a new value and saving it.

Assignment of Key Data for DTERM (AKYD)

This command allows you to program the 8 or 16 programmable line buttons on a digital phone. The buttons can be programmed to ring, flash, or dial other lines (allowing people to easily answer other lines or transfer calls).

Enter the primary station for the phone you'd like to program. In the grid below the phone information, you can enter operations for the line buttons. The

buttons are numbers left-to-right, top-to-bottom, with the top-left being line 1. Line 1 should be the primary extension for the phone.

In general, just copy an existing line to make another line ring on that phone. If you want the line to flash, but not ring, enter "0" in the **RG** column (instead of 7). If you want the button to speed-dial a number (rather than answer or join the line), enter "2" in the **KD** column.

We program all digital phones to have the "last" line button (either 8 or 16, depending on model of phone) dial voicemail. So that number should be programmed to speed dial (KD=2) voicemail (STN=4405) but not ring (RG=0).

Assignment of Station Data (ASDT)

This command allows you to create an extension by assigning a station number to a LEN. The LEN must be unassigned; if you need to unassign the LEN, put in the current station number and then say "Y" under **DEL**. You can then assign a new station to the LEN.

When you create a new extension, you must choose the phone type (TEC), restriction class (RSC), and service feature class (SFC). See the **terminology** section above for valid values. **Note**: the LEN is tied to a specific card in the PBX, and the card type determines the kind of phone that can be attached (analog or digital).

If you need to change the restriction classes, but don't need to reassign the LEN or TEC, use the **ASCL** command below.

If you need to change extension numbers, but don't need to change anything else (LEN, restriction, phone type, *etc.*), use the **ASTN** command below.

Assignment of Station Class Data (ASCL)

Assign phone type (TEC), restriction class (RSC), and service feature class (SFC) for an existing extension (if you need to change these values). More frequently, you will use the **ASDT** command above to create a new extension and simultaneously assign these values.

Assignment of Station Number (ASTN)

If you have an existing station number that you'd like to change, but do not need to change any other features (such as restriction class or phone type), use this command. Simply provide the existing extension and the new extension and the numbers will switch.

If you need to assign a station to a new LEN, use the **ASDT** command above.

Assignment of Call Forwarding Station Data (ACFS)

Users can set their phone to forward calls to another number if certain conditions are met (all, busy, no answer). This is useful for forwarding calls to voicemail, to another party, or to another number (*e.g.*, a cell phone).

Users can forward their phone by picking up the handset and dialing *5 (all calls), *6 (busy), or *7 (no answer), followed by the number to forward to (4405 is voicemail, or the extension of your choice).

We have assigned the system parameter that dictates that the FORWARDED station's restriction class is used when a call is forwarded. Thus, if Station A is forwarded to a long distance number, and Station B calls Station A, Station B will be prompted for a PIN if Station A would normally need one. If Station A is in a restriction class that does not require a PIN, then all calls (regardless of origin) will forward without needing a PIN.

To cancel forwarding, dial #5 (all calls), #6 (busy), or #7 (no answer).

For this command, enter 1 for **TYPE**, and use one of the following for **SRV**:

SRV	**Description**
V	All calls
D	Don't Answer
B	Busy
N	Not reachable (unsure what this is for)

You can list the current assigned forwarding status for a range of numbers by using the **LCFS** command.

Assignment of Authorization Code Data (AATC)

This command allows you to create PINs for users to dial restricted (long-distance) numbers. Depending on the restriction classes defined, the PIN allows dialing to local, domestic, or international numbers.

For **ACR**, enter 1.

For **RSC**, use one of the following restrictions, which define the class of numbers that may be dialed with this PIN (note that RSC 6 should always be used with SFC 3 (Admin), while RSC 2, 3, and 9 should only be used with SFC 6 (Student)):

RSC	Description
2	Student domestic and local (*e.g.*, all but international)
3	Student international, domestic, and local (*e.g.*, all numbers)
9	Student local only
6	Admin all numbers (use with SFC 3 below)

For **SFC**, use one of the following:

SFC	Description
3	Admin
6	Student

To examine existing PINs, use the **LATC** command (you can give a range like "0000000 - 9999999", though the report will take several minutes to generate).

Assignment of Date and Time (ATIM)

The clock in the PBX does drift over time, and also requires manual setting during the switch to/from daylight savings time. This command allows you to set the exact date and time.

Assignment of Calling Number Data (ACND)

When the PBX makes a call, it sets the Calling Party Number (CPN) and sends it to the called party (which uses it as caller ID). Additionally, this is the number shown to emergency services when 911 is dialed.

When the PBX makes a call, it sends the station number out as the CPN. However, we only use 4-digit extensions, so these numbers are not really valid as CPN.

ACND allows you to define rules for turning extensions into full CPN data.

The first field is **CNP**, which is a number from 1 - 255 defining the pattern that you'd like to define (you can define multiple patterns and then assign them with the **ACNP** command below). We use "1" as our default plan.

The next field is the **SKIP**, which is how many digits of the extension to get rid of (working left to right). If the skip is 0, the whole extension will be appended to the CPN data. If skip is the same length as the extension (in our case, 4), then the extension will not appear in the CNP at all (useful if you want to wholly replace a number with another one).

Next there is the **ADD** field, which tells how many digits will be prepended to the extension. This number should equal 10 - (length of extension) + SKIP.

In other words, this prefix plus whatever part of the extension you're sending should be 10.

Finally, there is the **DC** field, which defines the prefix to put in front of all CNP. It should be the same length as the **ADD** field.

Examples:

If you use 4-digit extensions, and want to send out a standard area code and prefix before the extension number, you would do:

```
 CNP: (number between 1 and 255)
SKIP: 0 (use whole extension)
 ADD: 6
  DC: 860386
```

A call placed from extension 4567 would come through as 8603864567.

If you want to wholly replace a number with another, you say:

```
 CNP: (number between 1 and 255)
SKIP: 4 (skip all 4 digits of extension)
 ADD: 10
  DC: 8603864400
```

Any number that matches this rule will be replaced with the full 10-digit number specified.

Our current ACND values are:

CNP	SKIP	ADD	DC	Notes
1	0	6	860386	Default rule (add are and prefix to station number)

See the **ACNP** command for information about assigning CPN based on the route, and **ACPNCL** command for overriding the CPN based on the dial prefix.

Assignment of Calling Number Pattern (ACNP)

Once you've defined one or more CNP using **ACND** above, you need to tell the switch which CNP apply to which numbers. ACNP does this based on the route that the call goes through.

First, you must say whether the route applies to **I**ncoming or **O**utgoing routes. We want to affect the CPN for outbound calls, so enter **O** (the letter "oh") here.

(Inbound routes would be for a site-to-site setup where you wanted to modify inbound caller ID from a trusted site.)

For **RT**, you must provide the route number that the calls are going through.

Finally, specify the **CNP** variable (1 - 255) that you created using the **ACND** command above.

Once committed, all calls going through the route will apply the pattern you specified.

Our current ACNP values are:

IC/OG	RT	CNP	Notes
O	10	1	Use default CNP for outgoing calls on PRI 1
O	20	1	Use default CNP for outgoing calls on PRI 2

See the **ACPNCL** command for a way to assign CNP based on the prefix of the station and override this command.

Assignment of Calling Party Number Conversion (ACPNCL)

This command is like a combination of **ACND** and **ACNP**; see those commands for a more detailed description of their options.

This command is useful to override the values set by ACNP (which is like a system-wide default) based on the number of the station placing the call. In this way, you can "exempt" a block of numbers from caller ID, or reassign a block to a standard number.

As we use both DID and non-DID numbers, we need to be careful not to send out non-DID extensions as part of our CPN. Our system-wide default is to send this info, so we use ACPNCL to override this default for the blocks of numbers that are **not** DIDs.

To use the command, you must specify an outgoing route (**OGRT**) and station prefix (**RCPN**). RCPN can be a full extension number (so it only applies to that station), or a prefix (for example, "49" would apply to all extensions starting with the digits "49", including 4901, 4902, *etc.*).

Note: RCPN numbers cannot overlap or override each other. In other words, you cannot make a rule for "42XX", and then try to override with "4234". If you need to do something like this, you need to break the RCPN into smaller non-overlapping prefixes and specify each one.

Click the **Get** button to look up the information.

You can now specify a **SKIP** and **ADD**, just like for **ACND**.

Finally, you must provide the **ACPN**, which is the pattern to prepend to the portion of the extension that hasn't been skipped (see the **DC** field of **ACND**).

For example, to replace the CPN for all extensions 4900-4999 with a single number, you would use:

```
OGRT: (route number)
RCPN: 49 (all numbers starting with 49)
SKIP: 4 (remove all 4 digits of the extension from the CPN)
 ADD: 10 (prepend with full 10-digit number)
ACPN: 8603864400
```

Below are our current settings for ACPNCL. Keep in mind that any pattern **not** matched below will use the default rules specified in **ACPN** above (in other words, "860386" will be prepended to the station number).

OGRT	RCPN	SKIP	ADD	ACPN	Notes
10	40	4	10	8603684400	4000-4099 internal-only (PRI 1)
20	40	4	10	8603864400	4000-4099 internal-only (PRI 2)
10	42	2	8	86038644	4200-4299 remap to 4400-4499 (PRI 1)
20	42	2	8	86038644	4200-4299 remap to 4400-4499 (PRI 2)
10	43	2	8	86038645	4300-4399 remap to 4500-4599 (PRI 1)
20	43	2	8	86038645	4300-4399 remap to 4500-4599 (PRI 2)
10	49	4	10	8603864400	4900-4999 internal-only (PRI 1)
20	49	4	10	8603864400	4900-4999 internal-only (PRI 2)

Assignment of Terminal Application Data (ADAI)

This command allows you to (amont other things) change the number of rings before the station forwards to another extension (*e.g.*, to voicemail).

The main things to look for here are the **TC** fields (timer) and **MTC** fields (multiplier). The **TC** fields have two possible values: **3**, which means "2 seconds", and **7**, which means "8 seconds" (no, that isn't a typo; the keys and their values have nothing to do with each other).

The multiplier fields determine how many total seconds the phone will ring for, based on the **TC** field.

For example, if **TC** is 3 and **MTC** is 8 (our system default), then the phone will ring for 2 x 8 or 16 seconds (approximately). A general rule of thumb is that the called station will hear a full ring cycle (1 ring plus 1 pause at the least, though possibly more if the ring tone is short) about every six seconds. The caller may hear different numbers of rings depending on their provider and some other timing issues.

The field you most likely want to override is 139, which controls the main answer time. You can also tweak 140 ("no answer recall"), or more likely 247 ("blind transfer no answer"), which gets involved if you transfer to another station. The PBX's call-forwarding mechanism does not appear to count as a "transfer", however.

All of the system defaults can be seen by running **ASYD** and probing the indices listed above (139, 140, 247). The **DATA** will list a hex value, where the first digit is 3 or 7 (the **TC**) and the second digit is anywhere from 1 to F (the **MTC**). ADAI will override these system-wide defaults.

Enter 1 for the **MODE**, the tennant and station number, and then zero for all other fields (except those you wish to modify). So, to modify the standard ring time, you'd enter a value for **139MTC** and **139TC** and zero for all the rest.

27.4 Voicemail

FIXME: describe recording vacation greeting, changing menus, etc.

27.5 User Commands

Our PBX allows users to customize their extension to provide special features or settings. Please note, sections marked "digital phones only" only apply to NEC Dterm multi-line phones (like those found in our administrative offices).

27.5.1 Forwarding Calls (all phones)

Users can forward their calls to another extension, or to our voicemail extension (4405). There are three conditions when you can forward calls, and each is set independently. The conditions are **all calls** (forward immediately), **busy** (forward only if you are on the line), and **don't answer** (forward only if you don't pick up after several rings).

Note: most users forward to **voicemail** for **busy** (*5 4405) and **don't answer** (*6 4405) only.

Pick up the handset and dial one of the following codes, depending on which kind of forwarding you want (if you want more than one kind of forwarding you must repeat these steps for each type you want):

Dial	Forwarding type
*5	All calls
*6	Busy
*7	Don't answer

After dialing the code, wait for broken dial tone. Then dial the number you wish to forward to (to forward to voicemail, enter **4405**). You will hear a confirmation tone. Hang up the handset.

To cancel forwarding, pick up the handset and dial one of the following codes, depending on which kind of forward you want to cancel (if you want to cancel more than one kind of forwarding you must repeat these steps for each type):

Dial	Forwarding type to cancel
#5	Cancel all calls forward
#6	Cancel busy forward
#7	Canclel don't answer forward

You will hear a confirmation tone, at which point the forward has been canceled.

27.5.2 Voicemail (all phones)

To check your voicemail from **on campus** from **your own extension** dial 4405 and follow the prompts.

To check your voicemail from **on campus** from **someon else's phone** dial 4205, wait for the recorded message to start, and then dial 9 plus your extension.

To check your voicemail from **off campus**, dial (860) 386-4405. Wait for the recorded message to start, an then dial 9 plus your extension.

27.5.3 Telephone Setup Functions (digital phones only)

Up/Down Arrows

Used to adjust LCD contrast, speaker/receiver volume, and ringer volume.

Feature Key

This key is used to set various features and functions on your telephone. To activate or cancel, press the **Feature** button first, then the appropriate number.

- **Feature +1** Turns microphone on or off. The **MIC** button lights when microphone is on.

- **Feature +2** Adjusts handset receiver volume on current call. The LCD displays the current volume (LARGE OR SMALL). Press **Feature** and **2** to alternate.

- **Feature +3** Selects ringer tone. Your telephone has 4 different ring tones. After pressing **Feature** and **3**, continue to press **3** until desired tone is heard.

- **Feature +4** Adjusts default handset volume. Options are SMALL (low) or LARGE (high). Press **Feature** and **4** to change.

- **Feature +5** Activates speakerphone capability.

- **Feature +6** Deactivates speakerphone capability.

- **Feature +7** Turns call indicator lamp on or off for incoming call notification. (If turned off, this lamp will still light to indicate message waiting).

27.5.4 System Features

Hold

To place a call on hold: Press **Hold** button; held line wink flashes. Note: If held line appears on other D-Term stations, the associated LED flashes red slowly.

To retrieve: Lift handset or press Speaker. Press held line. Note: Any station with this line appearance can retrieve the call.

If unanswered: After approximately two minutes, held call will automatically ring back to the phone that places it on hold. Caller does not hear this ringing. To retrieve call, press button for held line. To continue holding call without hearing automatic ringing, press button for held line, then press **Hold**.

To place a call on Exclusive Hold (cannot be retrieved from other phones): Press **Hold** twice; line light winks. (Other D-Term telephones show steady red light). To retrieve call: Lift handset; press button for held line. Call can be retrieved only from telephone that put it on Exclusive Hold.

To place call on hold and consult with third party: Ask party to hold (but do not press **Hold**); press **Transfer**, hear broken dial tone. Dial number of party you wish to consult. Converse with third party; when they hang up, you will automatically be connected with holding party. If third party does not answer, reconnect to holding party by pressing **Transfer** again.

Transfering Calls

To transfer a call: Press **Transfer**. Broken dial tone indicates caller is placed on hold. Dial destination number, wait one second for call to connect, then hang up.

(Optional): You may stay on the line to announce transfer of call. If party does not answer or line is busy, press **Transfer** to return to holding party.

Conference Calling (aka three-way calls)

To set up three-way conferencing: Establish first call (incoming or outgoing); ask party to hold.

Press **Transfer**; receive interrupted dial tone (caller is on hold). Dial third party. (If no answer or busy, press **Transfer** to reconnect to held party.)

When answered, press **Conf** Button (Conf LED lights). Three-way conference is established. If one party hangs up, other two remain connected. (Conf LED goes out).

Chapter 28

Catalyst 3550 Common

Last updated 2008/03/18

28.1 Introduction

Suffield Academy currently has a core network built primarily from equipment built by Cisco Systems. Many of our edge switches (such as those found in dormatories, classroom buildings, and other wiring closets) are the Catalyst 3550 model. These switches have gigabit uplink ports, and 10/100 ethernet switch ports for clients.

Because we have so many of this particular model, much of the configuration is the same from switch to switch. The purpose of this document is to describe the common configuration options for these switches in a single location. For switch-specific information (such as port naming, VLAN assignment, or other unique information), please see the documentation for that particular switch.

This document discusses site-wide configuration policy, such as authentication, DNS settings, NTP servers, and edge filtering policy.

28.1.1 Intended Audience

We assume that readers of this document understand basic command-line interaction with Cisco equipment. You should already be familiar with entering enable mode, editing configuration parameters, and saving them to flash memory. If you are not familiar (or comfortable) with these procedures, do **not** proceed.

28.1.2 Shortcut to Configs

This document describes the theory behind all of the configuration settings. If you just want to jump straight to our config files, you can find them here:

Catalyst 3550 Common Config Repository

28.2 Banner (MOTD and Login)

When users connect to the switch for management purposes, they can be shown a **message of the day** (MOTD) and/or login banner. Frequently, this area is used to display a warning about unauthorized access to the system, and possibly other useful information.

28.2.1 Suffield Config Files

The following sections discuss the theory behind the configuration setting we use at Suffield. If you're just interested in a copy of the actual config file sections, you can find them here:

Catalyst 3550 Common Banner Settings

28.2.2 Setting the MOTD Banner

The message of the day banner is shown to the user as soon as they connect to the switch's management interface.

To set the motd banner:

1. Enter enable mode:

   ```
   > enable
   ```

2. Enter configuration mode:

   ```
   # configure terminal
   ```

3. Announce that you wish to set the banner:

   ```
   (config)# banner motd #
   ```

The "**#**" mark is the **delimiter** used to mark the end of the banner. The switch will continue reading input until it sees this character again. If your MOTD has a "#" in it, use a different delimiter.

4. You will be prompted to enter your banner:

```
Enter TEXT message.  End with the character '#'.

This is my motd.
#
```

5. (Optional: exit config mode and save your changes.)

28.2.3 Setting the Login Banner

The message of the day banner is shown after the MOTD banner, and before the user authenticates to the switch.

To set the login banner:

1. Enter enable mode:

```
> enable
```

2. Enter configuration mode:

```
# configure terminal
```

3. Announce that you wish to set the banner:

```
(config)# banner login #
```

The "**#**" mark is the **delimiter** used to mark the end of the banner. The switch will continue reading input until it sees this character again. If your login banner has a "#" in it, use a different delimiter.

4. You will be prompted to enter your banner:

```
Enter TEXT message.  End with the character '#'.

This is my login banner.
#
```

5. (Optional: exit config mode and save your changes.)

28.2.4 Testing It Out

The next time you connect to the switch in command-line mode, you should see the MOTD and Login banners.

28.3 DNS

The 3550 is capable of performing its switching duties without being able to resolve Internet domain names. However, without name-to-address resolution, certain tasks become more difficult (such as configuring Kerberos). Therefore, all switches should have a proper DNS configuration.

28.3.1 Suffield Config Files

The following sections discuss the theory behind the configuration setting we use at Suffield. If you're just interested in a copy of the actual config file sections, you can find them here:

Catalyst 3550 Common DNS Settings

28.3.2 Setting the Hostname

All machines on the network should have a hostname, and this name should agree with the name provided by the DNS system.

To set the host name of the switch:

1. Enter enable mode:

   ```
   > enable
   ```

2. Enter configuration mode:

   ```
   # configure terminal
   ```

3. Set the host name: (replace `myname` with desired hostname)

   ```
   (config)# hostname myname
   ```

4. (Optional: exit config mode and save your changes.)

This changes the name of the switch. Note that this **does not** change the name in DNS! This is only the switch's local notion of its hostname; if you wish to change the DNS name you must do so on the DNS server itself.

28.3.3 Setting the DNS Name Servers

In order to resolve hostnames, the switch must be given a list of servers to contact for DNS information. The DNS servers must be given by their IP address.

To set the name servers for the switch:

1. Enter enable mode:

   ```
   > enable
   ```

2. Enter configuration mode:

   ```
   # configure terminal
   ```

3. Add a name server:

   ```
   (config)# ip name-server 172.30.0.2
   ```

4. Optionally, add more name servers if your site uses more than one:

   ```
   (config)# ip name-server 172.30.0.3
   ```

5. (Optional: exit config mode and save your changes.)

To remove old or outdated name server entries, use the "`no`" version of the command above.

28.3.4 Setting the Domain Suffix Search

Normally, DNS lookups are performed on fully-qualified domain names (FQDNs). These names have a full host and domain specification (*e.g.*, `gandalf.suffieldacademy.org`). However, many systems allow the user to omit the domain part and have it "guess" the correct suffix when none is supplied.

If your site has a flat domain structure, you will only have one suffix to use for unqualified names. For more complicated domains, you will need to supply a list of possible suffixes to try.

To set the list of domain suffixes for the switch:

1. Enter enable mode:

   ```
   > enable
   ```

2. Enter configuration mode:

```
# configure terminal
```

3. Add the first domain suffix to use for unqualified names:

```
(config)# ip domain-name suffieldacademy.org
```

4. Optionally, if you wish to use a list of domain suffixes, use multiple versions of this line (you must include the default domain, even if you added it with `domain-name` above):

```
(config)# ip domain-list suffieldacademy.org
(config)# ip domain-list net.suffieldacademy.org
(config)# ip domain-list gear.suffieldacademy.org
```

5. (Optional: exit config mode and save your changes.)

28.3.5 Testing It Out

With proper DNS configuration settings, you should be able to `ping` and `telnet` to other servers on the network, both by their FQDN or by just their hostname. You should see the hostname query going out and being resolved; it will look something like this:

```
Translating "gandalf"...domain server (172.30.0.2) [OK]
```

When you see that, you know DNS is working properly on the switch.

28.4 NTP

The 3550 has an internal system clock, which it uses to mark log entries, keep statistics, and perform other time-related functions. Additionally, Kerberos functionality requires that the switch's notion of time agree with that of other machines on the network. Therefore, the clock must both be very accurate, and kept in sync with the other clocks on the network.

The Network Time Protocol (NTP) allows hosts to communicate with a trusted server and get a reliable source of time information. The 3550 supports NTP natively, so syncronizing with a time server is relatively simple.

Note: this section assumes that you already have one or more stable NTP servers configured on your network. If you need to set up an NTP server, you

will need to refer to the documentation for the server operating system of your choice. Many machines can act as NTP servers (including some Cisco equipment); determining which system to use and setting it up is beyond the scope of this document.

28.4.1 Suffield Config Files

The following sections discuss the theory behind the configuration setting we use at Suffield. If you're just interested in a copy of the actual config file sections, you can find them here:

Catalyst 3550 Common NTP Settings

28.4.2 Setting the Clock Options

Before enabling NTP, you should ensure that the system clock options are correct for your site. This includes (optionally) setting the time zone, or leaving it as UTC.

To set the local clock options for the switch:

1. Enter enable mode:

   ```
   > enable
   ```

2. Enter configuration mode:

   ```
   # configure terminal
   ```

3. Set the "standard" time zone name and offset from UTC

   ```
   (config)# clock timezone EST -5
   ```

4. Set the "summer" (daylight savings) time zone name and offset. Additionally, you may optionally specify the dates and times when it goes into effect (this will become more important in 2007 when the US switches to a new set of DST rules):

   ```
   (config)# clock summer-time EDT recurring 1 Sunday April 2:00 last Sunday October 2:00 60
   ```

5. (Optional: exit config mode and save your changes.)

28.4.3 Setting the NTP Servers

You must provide at least one (and preferably more) NTP servers to use as a stable source of time information.

To set the time servers for the switch:

1. Enter enable mode:

   ```
   > enable
   ```

2. Enter configuration mode:

   ```
   # configure terminal
   ```

3. Add a time server:

   ```
   (config)# ntp server 172.30.0.2
   ```

4. Optionally, add more time servers if your site uses more than one:

   ```
   (config)# ntp server 172.30.0.3
   ```

5. (Optional: exit config mode and save your changes.)

To remove old or outdated time server entries, use the "`no`" version of the command above.

28.4.4 Testing It Out

Once you've entered the time server information, you can query the switch for its status:

```
# show ntp status
```

The output should show that the clock is synchronized, and list one of your servers as its primary reference.

Additionally, you can view the list of servers and see what the syncronization status is for each of them:

```
# show ntp associations
```

28.5 Kerberos

(**FIXME**: not yet written.)

28.5.1 Suffield Config Files

The following sections discuss the theory behind the configuration setting we use at Suffield. If you're just interested in a copy of the actual config file sections, you can find them here:

Catalyst 3550 Common Kerberos Settings

28.6 Syslog

The 3550 has the capability to send log information to a remote `syslog` server. This allows for centralized collection of log messages, and also allows for log analysis long after the switch's internal logging buffer may have cleared.

This document assumes you have already set up a centralized syslog server that is ready to receive messages from the switch.

28.6.1 Suffield Config Files

The following sections discuss the theory behind the configuration setting we use at Suffield. If you're just interested in a copy of the actual config file sections, you can find them here:

Catalyst 3550 Common Syslog Settings

28.6.2 Basic Logging Setup

Begin by defining the global settings for logging that you would like to use. This means turning on logging and defining the format that log messages should take.

To set the logging options on the switch:

1. Enter enable mode:

   ```
   > enable
   ```

2. Enter configuration mode:

```
# configure terminal
```

3. Enable logging:

```
(config)# logging on
```

4. Set the timestamp format to include any information you'd like (time, zone, etc):

```
(config)# service timestamps log datetime msec localtime show-timezone
```

5. Disable sequence numbers in the log entries if you don't want them (we use millisecond stamps in the timestamp format above, and so don't need sequence numbers):

```
(config)# no service sequence-numbers
```

6. (Optional: exit config mode and save your changes.)

28.6.3 Local Logging Options

The switch retains some logging information that can be viewed with the `show logging` command. You can determine how much information to retain and its format.

To set the local logging options for the switch:

1. Enter enable mode:

```
> enable
```

2. Enter configuration mode:

```
# configure terminal
```

3. Set the local log buffer to 8Kb (default is 4Kb):

```
(config)# logging buffered 8192
```

4. (Optional: exit config mode and save your changes.)

28.6.4 Remote Syslog Options

The best use of the logging system is to send the messages to a remote host for collection, archiving, and possible analysis. You may specify more than one server, if desired.

To set the remote syslog server options for the switch:

1. Enter enable mode:

   ```
   > enable
   ```

2. Enter configuration mode:

   ```
   # configure terminal
   ```

3. Specify the remote server to send log information to (you may specify more than one line for multiple servers):

   ```
   (config)# logging 172.30.0.10
   ```

4. Set the maximum syslog level of messages that will be sent to the server. Levels range from 7 (debugging) to 0 (emergencies), with each level including all levels below it. Note that "debugging" may generate a very large amount of messages!

   ```
   (config)# logging trap informational
   ```

5. Set the syslog facility to use when sending messages. The remote server may use this facility to sort or group messages. In general, you should select one of the "local" facilities (numbered 0 - 7):

   ```
   (config)# logging facility local7
   ```

6. (Optional: exit config mode and save your changes.)

28.6.5 Testing It Out

As soon as the logging settings have been established, your remote logging host should begin to receive messages from the switch. Additionallly, running `show logging` on the switch should show a list of the messages in the local cache. They should have the new timestamp format you defined.

28.7 Edge Filtering

We use various methods for filtering traffic on our internal LAN. Some of this filtering is done for policy reasons (to prevent access to certain network segments or hosts), and some is done for performance or protocol reasons.

Most of the security-related configuration is performed on our core switch, where traffic is switched between VLANs. Additionally, the core performs policy routing to send packets to particular destinations (for example, routing them over a different internet connection, or forcing them to use a web proxy).

However, low-level policy filtering needs to be applied at the VLAN level (rather than only when packets travel between VLANs). These types of rules prevent unwanted packets from ever making it on to the network, and as such they must be supplied at the edge switches.

28.7.1 Suffield Config Files

The following sections discuss the theory behind the configuration setting we use at Suffield. If you're just interested in a copy of the actual config file sections, you can find them here:

Catalyst 3550 Common Filter Settings

28.7.2 Types of Unwanted Traffic

At Suffield, we run a relatively open network. However, we do prevent some troublesome protocols from being used on the network in an effort to make the network more robust.

Currently, Suffield uses edge filtering to **prevent** the following from being used on the network:

1. AppleTalk ethernet frames (AFP via IP is now preferred over AppleTalk, and most printers now support IPP or LPD for print services). AppleTalk is very "chatty", and also difficult to filter for purposes of access control.
2. "Rogue" DHCP broadcasts. Users who connect 3rd-party routers, or who enable "Internet connection sharing" may accidentally broadcast DHCP responses onto the network. This can cause other clients on the network to use incorrect configurations and be unable to access the network properly. We block all DHCP server packets from "untrusted" machines.
3. "Rogue" IPP broadcasts. Users frequently enable "Printer sharing" on their computers, advertising themselves as a route to a printer that is

better reached directly. By discarding these advertisements, we ensure that clients always connect directly to printers rather than selecting an untrusted machine.

4. Multicast DNS advertisements. We run a well-configured network, including a static DNS Service-Discovery configuration to allow users to find printers and other resources. To prevent users from registering their own machines (or duplicates of our services), we quash ad-hoc advertisements.

5. Spoofed addresses. Our internal LAN uses a specific segment of RFC 1918 private IP ddresses, so we automatically drop any packets that are not sourced or destined to this block of addresses.

28.7.3 Quick Background: ACLs and VLAN Access Maps

The **only way** to filter traffic **on** a VLAN (as opposed to **between** VLANs) is to use a combination of Access Control Lists (ACLs) and VLAN access-maps.

First, we build several ACLs which identify packets that we wish to match. We may wish to match packets to explicitly deny, or to specifically allow, so the ACLs do not actually decide what to do with the packets; they simply identify the packets we're interested in.

Next, we create VLAN access maps, which decide what to do with packets that match a particular ACL. **Note:** access maps apply to all traffic on a VLAN, **regardless of direction**, so your ACLs must match traffic in both directions (ingress and egress).

Finally, we apply the VLAN access maps to one or more VLANs on the edge switch. If the map is generic enough, it can be applied to several VLANs, making administration simpler.

28.7.4 ACLs

To identify packets, we must create ACLs that match these packets. Two types of ACL are available: "`ip`" access lists, and "`mac`" access lists. IP access lists only match IP protocol information such as IP address, protocol (TCP/UDP), and port number. MAC access lists can match physical Ethernet addresses (48-bit hexadecimal), as well as Ethernet protocol families (AppleTalk, DECNET, IPX, VINES).

Because of the way VLAN access maps are applied, you **must** create an ACL for each type of traffic you wish to **allow** (access maps deny packets by default whenever an ACL is present). Also keep in mind that ACLs are applied in order, with the first matching ACL being the one that gets applied. You may need to create several ACLs to properly match certain types of traffic.

Finally, all ACLs are comprised of rules that either "permit" or "deny" packets. In the context of VLAN access maps, "permit" means that the packet matches the current access map entry, and "deny" means that it does not. **The VLAN access map decides what to do with a matching packet**, not the ACL! Therefore, it is possible to have a packet "permitted" by an ACL, but have the action in the map set to "drop". Similarly, just because you "deny" a packet in an ACL does not mean it is dropped; it simply means that the packet is no longer considered for that VLAN access map entry.

Default ACLs

In order to allow traffic on the VLAN, its helpful to create a few default ACLs to permit all (or most) traffic. Because we run a fairly open network, we have selected to allow all traffic that we do not explicitly deny. You may wish to change this to a more restrictive policy.

VLAN access maps explicitly deny unmatched packets whenever any matching rule for that packet type exists. In other words, if you specify at least one IP ACL, then any IP packets **not** matching that ACL are dropped. Similarly, if you specifiy at least one MAC ACL, any non-IP packets that do **not** match that ACL are dropped. Therefore, if you plan to use both IP and MAC ACLs, at least one of each type must be used to permit traffic, or you'll end up denying everything by default.

Also note that Cisco gear treats fragmented packets differently from normal (header) packets. If you don't explicitly match fragments, they may end up getting denied because fragments do not include full port information.

Here is a simple IP ACL which matches all IP fragments:

```
ip access-list extended vlan_filter_ip_fragments
 permit ip any any fragments
```

Here is a simple IP ACL which matches all (non fragmented!) IP-based traffic:

```
ip access-list extended vlan_filter_ip_any
 permit ip any any
```

And here is a simple MAC ACL which matches all non-IP (Ethernet) traffic:

```
mac access-list extended vlan_filter_mac_any
 permit any any
```

Note that these rules say nothing about blocking or passing traffic; they simply match packets. We will use these lists later in our VLAN access maps to make policy decisions.

AppleTalk

AppleTalk is a protocol developed by Apple Computer. It provides a broadcast-based system for advertising services and hosts. Because of this, the protocol is very "chatty". Additionally, it requires an entirely separate set of controls (*i.e.*, "zones"), as it is not IP-based.

The primary uses of AppleTalk (file sharing and printing) have been ported to IP-based protocols (AFP over TCP for file sharing, and IPP or LPD for printing). Therefore, AppleTalk is unecessary in a modern network. However, many devices (especially printers) still ship with AppleTalk turned on by default, so filtering helps prevent misconfigured devices from flooding the network.

Because AppleTalk is a non-IP protocol, we must use a MAC ACL. Here is an ACL that matches AppleTalk packets:

```
mac access-list extended vlan_filter_appletalk
 permit any any aarp
 permit any any appletalk
```

AARP is the AppleTalk Address Resolution Protocol, and is the most vital to block (as it prevents services from finding each other). It serves the same purpose as ARP packets on an IP network. Without it, address mappings cannot take place, and communication is prohibited.

Rogue DHCP Servers

Many sites use DHCP for automatic assignment of IP addresses, as well as other configuration information (subnet mask, gateway addresses, name servers, etc).

Because the purpose of DHCP is to provide configuration to clients with no knowledge of the network topology, the service model is very insecure. Clients simply broadcast a query on the network and hope for a response. Meanwhile, any listening server broadcasts back a response for the client to use.

The system works well, provided that the only DHCP servers on the network are the ones you set up. Unfortunately, numerous consumer devices (wireless access points, routers, or even computers with "Internet sharing" enabled) are configured to act as DHCP servers as well. If these devices are connected to the network, they will attempt to answer DHCP queries. Because the client does not know who to trust, it simply chooses the first response as its source of authoritative information.

We can remedy this situation by preventing the broadcast of DHCP server packets from all computers except those that we trust. We use an ACL which

matches all **bad** servers, and ignores thost that we trust, so that we can take specific action on the untrusted packets.

Note that you must list any servers which provide DHCP service, as well as any DHCP relays on your network (if you use relays to forward the broadcast packets accross VLANs).

Here is a sample ACL for identifying rogue DHCP servers:

```
ip access-list extended vlan_filter_dhcp_rogue
 remark Allow Suffield servers to broadcast
 deny   udp host 172.30.0.2 eq bootps any eq bootpc
 deny   udp host 172.30.0.3 eq bootps any eq bootpc
 deny   udp host 172.24.48.4 eq bootps any eq bootpc
 remark Allow DHCP helper relay addresses (gateway addresses)
 deny   udp host 172.22.0.1 eq bootps any eq bootpc
 deny   udp host 172.24.0.1 eq bootps any eq bootpc
 deny   udp host 172.28.0.1 eq bootps any eq bootpc
 deny   udp host 172.30.0.1 eq bootps any eq bootpc
 deny   udp host 172.31.0.1 eq bootps any eq bootpc
 remark Prevent all unauthorized clients from broadcasting
 permit udp any eq bootps any eq bootpc
```

Note the use of "deny" for the trusted hosts; this prevents the ACL from matching packets from our known servers. Later, when we apply the VLAN access map, we will use it to drop packets that match this ACL.

IPP Broadcasts

At Suffield, we use the Internet Printing Protocol (IPP) as our preferred means of submitting jobs to networked printers. IPP has many features, including a way to share printers between computers by broadcasting their availability on the network.

Some users turn on this printer sharing, and begin advertising themselves as a route to all the networked printers. Obviously, this is not a desirable state, as other users should be printing directly to the printers, not through other computers.

To prevent this advertisement of printers, we have an ACL that prevents these broadcast messages from being sent:

```
ip access-list extended vlan_filter_ipp
 permit udp any any eq 631
```

Note that if you use print servers on your network, you may need to specifically exempt them from this ACL (using "deny" rules) so that they can still broadcast to the network.

mDNS Advertisements

Mac OS X automatically advertises a computer on the network using ad-hoc DNS Service Discovery (sometimes branded as "Bonjour", "Rendezvous", or "Zeroconf"). This is great for ad-hoc networks with no formal DNS structure, but on a well-run network it becomes a hinderance when user-advertised services conflict with legitimate ones.

Services are advertised via multicast DNS, which uses a specific multicast address and port. Therefore, we can easily deny it by blocking all traffic with this address/port combination:

```
ip access-list extended vlan_filter_mdns
 permit udp any host 224.0.0.251 eq 5353
 permit tcp any host 224.0.0.251 eq 5353
```

Rogue IP Filter

As an added layer of security, we filter all IP traffic to ensure that it originates from, or is destined to, a valid IP address. Our internal network uses a single block of RFC 1918 private IP addresses, so we filter out any packets that do not use these addresses. This prevents misconfigured clients from sending packets on the network.

Note that this approach requires two additional steps to work correctly:

1. Broadcast traffic from unconfigured DHCP clients must be explicitly allowed, regardless of the source address. Unconfigured clients may use 0.0.0.0 as an address, or part of the auto-assigned 169.254.0.0/16 block (or any other address), so we must be careful not to filter these requests for DHCP.

2. Multicast traffic uses a separate special IP block, known as the Class D block, at 224.0.0.0/4. We must also allow addresses to/from this block if we wish to allow multicast IP on our network. Of course, if you don't want multicast on your network, you can leave this out.

Here is our ACL to prevent unwanted IP traffic. Again, the rule is to "deny" packets that we want, as this ACL will be used in an access map rule that drops the matched packets.

```
ip access-list extended vlan_filter_ip_rogue
 remark Allow traffic to/from our internal addresses
 deny   ip 172.16.0.0 0.15.255.255 any
 deny   ip any 172.16.0.0 0.15.255.255
```

```
remark Allow multicast traffic (224.0.0.0/4)
deny   ip 224.0.0.0 15.255.255.255 any
deny   ip any 224.0.0.0 15.255.255.255
remark Allow anyone to make a DHCP request
deny   udp any eq bootpc any eq bootps
remark Prevent non-Suffield IP addresses from getting on the network
permit ip any any
```

Finally, note that if you use other/additional IP ranges on your network, you must change the values in the rule above.

28.7.5 VLAN Access Maps

Once we've written our ACLs, we need to match traffic against them and take specific actions on those matched packets. The way we do this is by using a **VLAN access map**. Just as a reminder: access maps do not differentiate between "ingress" and "egress" filtering (which is different from some other Cisco ACL-based commands). All traffic on the VLAN is matched against these maps.

The basic operation of the access map is simple. The map consists of one or more **tests**, with each test having an **action** associated with it. A packet is checked against each test in order, and the first one that matches causes the action to be taken on it. Packets that do not match any tests are dropped by default (unless there are no tests, but we'll assume that is not the case here). Therefore, you must have at least one test which allows packets, or you'll end up blocking everything.

Since the matching rules are considered in order, you must apply the rules so that packets always match more specific rules first. In our case, we have a set of specific "drop" rules, followed by some permissive "forward" rules. In this sense, we eliminate all traffic we know we don't want, and allow everything else by default.

Here is our access map entry. Note the rule numbers applied to each matching condition. Also, note that the action for each rule; sometimes we explicitly forward packets, and sometimes we drop them. The ACL names given here match those discussed in previous sections.

```
vlan access-map vlan_broadcast_suppress 100
match mac address vlan_filter_appletalk
action drop

vlan access-map vlan_broadcast_suppress 200
match ip address vlan_filter_dhcp_rogue
action drop

vlan access-map vlan_broadcast_suppress 300
match ip address vlan_filter_ipp
```

```
action drop

vlan access-map vlan_broadcast_suppress 400
match ip address vlan_filter_ip_rogue
action drop

vlan access-map vlan_broadcast_suppress 65533
match ip address vlan_filter_ip_any
action forward

vlan access-map vlan_broadcast_suppress 65534
match mac address vlan_filter_mac_any
action forward
```

28.7.6 Applying Access Maps and Testing

Once the VLAN access map has been created, we apply it to the VLANs where the policy should have effect. You may specify each VLAN individually, or provide ranges of VLANs.

Warning: improperly configured access maps may block all traffic on a given VLAN. If you are accessing your switch remotely, make sure to test your configuration on a VLAN other than the one you're on. Otherwise, you may disconnect yourself from the switch and be unable to revert the changes remotely.

Our VLANs all fall within a well-specified range, so we simply apply the range operator to cover all our VLANs:

```
vlan filter vlan_broadcast_suppress vlan-list 500 - 899
```

The command should take effect immediately. Confirm that normal traffic is still being passed, and that unwanted traffic is being blocked. If necessary, you can remove the filter simply by using the "no" version of the command above.

28.8 Switch Installation / Replacement

This section describes how to prepare a "factory-fresh" switch for use on our network. Usually, this will occur when a failed switch is replaced with a new unit.

28.8.1 Hardware Setup

1. Attach all GBICs and external interfaces to the switch

2. Attach the primary cabling to the switch (usually, this will be the fiber optic link from the core)

3. Attach a serial cable to the switch's monitor port, and start a terminal emulation program on your computer

4. Power up the switch

28.8.2 Initial Configuration

1. When asked "Would you like to enter the initial configuration dialog", say **no**.

2. When asked "Would you like to terminate autoinstall", say **yes**.

3. You can now enter a command prompt. Type **en** to enter enable mode so you can begin configuration.

28.8.3 VLAN Configuration

In order to properly configure the switch, it must be connected to the core, and it must download the VLAN database.

1. Start by moving into configuration mode:

```
conf t
```

2. Set the trunk interface to the correct mode (substitute the correct interface):

```
int gi0/1
switchport trunk encapsulation dot1q
switchport mode trunk
no shut
exit
```

3. Next, we need to get the VLAN database from the core switch. We do this by joining the VTP domain, which slaves us to the master switch:

```
vtp domain suffield
vtp password <use actual password here>
vtp version 2
vtp mode client
```

4. Set an IP address for this switch:

```
int vlan901
ip address 172.31.0.XXX 255.255.128.0
no shut
exit
```

5. Finally, exit configuration mode:

   ```
   exit
   ```

   After several seconds, the management VLAN should come up, and you should be able to ping the core switch. At this point, the switch is configured for network use, and ready for the next step.

   Do not proceed unless you have completed a successful ping test.

28.8.4 Upgrading the IOS Software

Depending on the age of the switch you're installing, you may need to replace the software with the version currently in production on our other switches.

Preparing to Upgrade

1. Confirm the software version on the switch:

   ```
   show version
   ```

   The first line of output should show the IOS version for the switch. If it is not identical to the line on our production switches, you must upgrade the IOS software.

2. Confirm that there is enough free space on the flash disk to perform an upgrade:

   ```
   dir flash:
   ```

   You'll get an accounting of free space from this command. If there is not enough room for the new image, delete files using the `delete flash:<file name>` command. There should be ample room for both the current and upgraded IOS images.

Setting Up a TFTP Server

The easiest way to get the new software image to the switch is via TFTP. Several TFTP servers exist for UNIX and Windows computers. We use Mac OS X, which comes with a built-in TFTP server. If you are not using Mac OS X 10.4 or better, please find other documentation on setting up a TFTP server.

1. To see if tftpd is running, type the following in the Mac OS X terminal:

   ```
   sudo launchctl list | grep -i tftp
   ```

   If you see `com.apple.tftpd`, then the service is running, and you're ready to go.

2. If you need to start the service, type the following:

   ```
   sudo launchctl load -w /System/Library/LaunchDaemons/tftp.plist
   ```

   Then, run the first command again to confirm that it's started.

   If you wish to stop the TFTP service, simply run:

   ```
   sudo launchctl unload -w /System/Library/LaunchDaemons/tftp.plist
   ```

3. The TFTP server serves files from the `/private/tftpboot` directory. You'll need to copy any software images you'd like to serve to that directory:

   ```
   sudo cp <path to image> /private/tftpboot/
   ```

   The "path to image" should be a **bin** file downloaded from Cisco (or from our repository). The name should be something like `c3550-ipservicesk9-mz.122-25.SEE2.bin`.

4. Finally, make note of the IP address of your computer. You'll need this to connect from the swtich to your computer to download the files.

Upgrading the IOS Software

1. On the switch, in enable mode, copy the IOS image to the flash disk:

   ```
   copy tftp flash:
   ```

   You will be prompted for the address of the remote host. Enter the IP address of the machine with the TFTP server.

You will be prompted for a filename to copy. Enter the name of the IOS image. You should not need to enter any leading path information (such as `/private/tftpboot`).

You will be prompted for a destination filename. Just hit return to accept the default (which matches the source filename).

The switch will show a progress indicator as it downloads the image.

2. Verify that the image was correctly transferred:

```
verify flash:<name of image>
```

3. Set the boot variable on the switch to use the new image:

```
conf t
boot system flash:<name of image>
exit
```

4. Verify that the boot parameter is correct:

```
show boot
```

5. Write the configuration to memory:

```
write memory
```

6. Reload the switch:

```
reload
```

7. Once the switch has rebooted, confirm that the IOS version has changed:

```
show version
```

28.8.5 Restoring the Configuration

If this is a replacement switch, you'll want to load a backed up copy of the configuration onto the new switch.

The best way to restore the configuration is via TFTP. Make sure you have a functioning TFTP server (as described in the previous section).

1. Copy the backup of the switch configuration into the TFTP directory:

```
sudo cp <path to config> /private/tftpboot/
```

2. On the switch, in enable mode, copy the configuration on to the switch:

   ```
   copy tftp running-config
   ```

   You will be prompted for the address of the remote host. Enter the IP address of the machine with the TFTP server.

   You will be prompted for a filename to copy. Enter the name of the configuration file. You should not need to enter any leading path information (such as `/private/tftpboot`).

   You will be prompted for the destination filename. Just press return to accept the default (**running-config**).

   A progress indicator will show the load of the config file. Additional, the configuration file may cause one-time events to fire (such as the generation of PKI SSH keys, NTP queries, or DNS probes).

3. At this point, the running configuration of the switch should be correctly loaded from the backup configuration. Spot-check the configuration (interface names, VLAN assignments, connectivity, DNS resolution). If everything appears to be working properly, save the configuration permanently:

   ```
   write memory
   ```

At this point, the switch has been properly restored, and is ready for active service.

Part IV

Diagrams

Contains diagrams and configurations of our core network, switches, racks, server room, phones, and other hardwired infrastructure.

In the future, this may contain other architecture diagrams, such as database schema, DNS maps, and DHCP scopes.

This section not yet populated