SQUID Frequently Asked Questions

c 2001 Duane Wessels, wessels@squid-cache.org

Frequently Asked Questions (with answers!) about the Squid Internet Object Cache software.

Contents

1 About Squid, this FAQ, and other Squid information resources **15**

1.1 What is Squid? . . . 15
1.2 What is Internet object caching? . . . 15
1.3 Why is it called Squid? . . . 15
1.4 What is the latest version of Squid? . . . 16
1.5 Who is responsible for Squid? . . . 16
1.6 Where can I get Squid? . . . 16
1.7 What Operating Systems does Squid support? . . . 16
1.8 Does Squid run on Windows NT? . . . 17
1.9 What Squid mailing lists are available? . . . 17
1.10 I can't gure out how to unsubscribe from your mailing list. . . . 17
1.11 What Squid web pages are available? . . . 17
1.12 Does Squid support SSL/HTTPS/TLS? . . . 18
1.13 What's the legal status of Squid? . . . 18
1.14 Is Squid year-2000 compliant? . . . 19
1.15 Can I pay someone for Squid support? . . . 19
1.16 Squid FAQ contributors . . . 19
1.17 About This Document . . . 21
1.17.1 Want to contribute? Please write in SGML... . . . 21

2 Getting and Compiling Squid **22**

2.1 Which le do I download to get Squid? . . . 22
2.2 How do I compile Squid? . . . 22
2.3 What kind of compiler do I need? . . . 22
2.4 What else do I need to compile Squid? . . . 22
2.5 Do you have pre-compiled binaries available? . . . 22
2.6 How do I apply a patch or a di? . . . 23
2.7 *congure* options . . . 23
2.8 undened reference to __inet_ntoa . . . 24
2.9 How can I get true DNS TTL info into Squid's IP cache? . . . 24

2.10 My platform is BSD/OS or BSDI and I can't compile Squid . . . 26
2.11 Problems compiling *libmiscutil.a* on Solaris . . . 26
2.12 I have problems compiling Squid on Platform Foo. . . . 27
2.13 I see a lot warnings while compiling Squid. . . . 27
2.14 Building Squid on OS/2 . . . 27

3 Installing and Running Squid **28**
3.1 How big of a system do I need to run Squid? . . . 28
3.2 How do I install Squid? . . . 28
3.3 What does the *squid.conf* le do? . . . 29
3.4 Do you have a *squid.conf* example? . . . 29
3.5 How do I start Squid? . . . 29
3.6 How do I start Squid automatically when the system boots? . . . 30
3.6.1 From inittab . . . 30
3.6.2 From rc.local . . . 31
3.6.3 From init.d . . . 31
3.7 How do I tell if Squid is running? . . . 32
3.8 *squid* command line options . . . 32
3.9 How do I see how Squid works? . . . 33

4 Conguration issues **34**
4.1 How do I join a cache hierarchy? . . . 34
4.2 How do I join NLANR's cache hierarchy? . . . 34
4.3 Why should I want to join NLANR's cache hierarchy? . . . 35
4.4 How do I register my cache with NLANR's registration service? . . . 35
4.5 How do I nd other caches close to me and arrange parent/child/sibling relationships with them? . . . 35
4.6 My cache registration is not appearing in the Tracker database. . . . 35
4.7 What is the httpd-accelerator mode? . . . 35
4.8 How do I congure Squid to work behind a rewall? . . . 35
4.9 How do I congure Squid forward all requests to another proxy? . . . 36
4.10 I have *dnsserver* processes that aren't being used, should I lower the number in *squid.conf*? . . . 36
4.11 My *dnsserver* average/median service time seems high, how can I reduce it? . . . 37
4.12 How can I easily change the default HTTP port? . . . 37
4.13 Is it possible to control how big each *cache_dir* is? . . . 37
4.14 What *cache_dir* size should I use? . . . 37
4.15 I'm adding a new *cache_dir*. Will I lose my cache? . . . 38
4.16 Squid and *http-gw* from the TIS toolkit. . . . 38

4.16.1 Firewall conguration: . . . 38
4.16.2 Squid conguration: . . . 38
4.17 What is "HTTP_X_FORWARDED_FOR"? Why does squid provide it to WWW servers, and how can I stop it? . . . 39
4.18 Can Squid anonymize HTTP requests? . . . 40
4.18.1 Squid 2.2 . . . 40
4.18.2 Squid 2.1 and Earlier . . . 40
4.19 Can I make Squid go direct for some sites? . . . 42
4.20 Can I make Squid proxy only, without caching anything? . . . 42
4.21 Can I prevent users from downloading large les? . . . 42

5 Communication between browsers and Squid 42

5.1 Netscape manual conguration . . . 43
5.2 Netscape automatic conguration . . . 43
5.3 Lynx and Mosaic conguration . . . 44
5.4 Redundant Proxy Auto-Conguration . . . 45
5.5 Proxy Auto-Conguration with URL Hashing . . . 45
5.6 Microsoft Internet Explorer conguration . . . 46
5.7 Netmanage Internet Chameleon WebSurfer conguration . . . 46
5.8 Opera 2.12 proxy conguration . . . 46
5.9 How do I tell Squid to use a specic username for FTP urls? . . . 47
5.10 Conguring Browsers for WPAD . . . 47
5.11 Conguring Browsers for WPAD with DHCP . . . 48
5.12 IE 5.0x crops trailing slashes from FTP URL's . . . 48
5.13 IE 6.0 SP1 fails when using basic authentication . . . 49

6 Squid Log Files 49

6.1 *squid.out* . . . 49
6.2 *cache.log* . . . 50
6.3 *useragent.log* . . . 50
6.4 *store.log* . . . 50
6.5 *hierarchy.log* . . . 51
6.6 *access.log* . . . 52
6.6.1 *The common log le format* . . . 52
6.6.2 *The native log le format* . . . 52
6.6.3 *access.log native format in detail* . . . 52
6.7 Squid result codes . . . 54
6.8 HTTP status codes . . . 56

6.9 Request methods . . . 57
6.10 Hierarchy Codes . . . 58
6.11 *cache/log* (Squid-1.x) . . . 60
6.12 *swap.state* (Squid-2.x) . . . 60
6.13 Which log les can I delete safely? . . . 61
6.14 How can I disable Squid's log les? . . . 61
6.15 My log les get very big! . . . 61
6.16 I want to use another tool to maintain the log les. . . . 61
6.17 Managing log les . . . 62
6.18 Why do I get ERR_NO_CLIENTS_BIG_OBJ messages so often? . . . 62
6.19 What does ERR_LIFETIME_EXP mean? . . . 63
6.20 Retrieving "lost" les from the cache . . . 63
6.21 Can I use *store.log* to gure out if a response was cachable? . . . 63

7 Operational issues **63**

7.1 How do I see system level Squid statistics? . . . 63
7.2 How can I nd the biggest objects in my cache? . . . 63
7.3 I want to restart Squid with a clean cache . . . 64
7.4 How can I proxy/cache Real Audio? . . . 64
7.5 How can I purge an object from my cache? . . . 65
7.6 Using ICMP to Measure the Network . . . 66
7.6.1 Supporting ICMP in your Squid cache . . . 66
7.6.2 Utilizing your parents database . . . 66
7.6.3 Inspecting the database . . . 67
7.7 Why are so few requests logged as TCP_IMS_MISS? . . . 67
7.8 How can I make Squid NOT cache some servers or URLs? . . . 67
7.9 How can I delete and recreate a cache directory? . . . 68
7.10 Why can't I run Squid as root? . . . 69
7.11 Can you tell me a good way to upgrade Squid with minimal downtime? . . . 69
7.12 Can Squid listen on more than one HTTP port? . . . 69
7.13 Can I make origin servers see the client's IP address when going through Squid? . . . 70

8 Memory **70**

8.1 Why does Squid use so much memory!? . . . 70
8.2 How can I tell how much memory my Squid process is using? . . . 70
8.3 My Squid process grows without bounds. . . . 71
8.4 I set *cache_mem* to XX, but the process grows beyond that! . . . 71
8.5 How do I analyze memory usage from the cache manger output? . . . 72

8.6 The "Total memory accounted" value is less than the size of my Squid process. 73
8.7 xmalloc: Unable to allocate 4096 bytes! 73
8.7.1 BSD/OS 73
8.7.2 FreeBSD (2.2.X) 74
8.7.3 OSF, Digital Unix 75
8.8 fork: (12) Cannot allocate memory 75
8.9 What can I do to reduce Squid's memory usage? 76
8.10 Using an alternate *malloc* library. 76
8.10.1 Using GNU malloc 76
8.10.2 dlmalloc 77
8.11 How much memory do I need in my Squid server? 77

9 The Cache Manager 78

9.1 What is the cache manager? 78
9.2 How do you set it up? 78
9.3 Cache manager conguration for CERN httpd 3.0 78
9.4 Cache manager conguration for Apache 78
9.5 Cache manager conguration for Roxen 2.0 and later 79
9.6 Cache manager ACLs in *squid.conf* 80
9.7 Why does it say I need a password and a URL? 81
9.8 I want to shutdown the cache remotely. What's the password? 81
9.9 How do I make the cache host default to *my* cache? 81
9.10 What's the dierence between Squid TCP connections and Squid UDP connections? 81
9.11 It says the storage expiration will happen in 1970! 81
9.12 What do the Meta Data entries mean? 81
9.13 In the utilization section, what is `Other`? 82
9.14 In the utilization section, why is the `Transfer KB/sec` column always zero? 82
9.15 In the utilization section, what is the `Object Count`? 82
9.16 In the utilization section, what is the `Max/Current/Min KB`? 82
9.17 What is the `I/O` section about? 82
9.18 What is the `Objects` section for? 82
9.19 What is the `VM Objects` section for? 83
9.20 What does `AVG RTT` mean? 83
9.21 In the IP cache section, what's the dierence between a hit, a negative hit and a miss? 83
9.22 What do the IP cache contents mean anyway? 83
9.23 What is the fqdncache and how is it dierent from the ipcache? 83
9.24 What does "Page faults with physical i/o: 4897" mean? 84

9.24.1 Ok, so what is unusually high? . . . 86
9.25 What does the IGNORED eld mean in the 'cache server list'? . . . 86

10 Access Controls **86**

10.1 Introduction . . . 86
10.1.1 ACL elements . . . 86
10.1.2 Access Lists . . . 87
10.2 How do I allow my clients to use the cache? . . . 88
10.3 how do I congure Squid not to cache a specic server? . . . 88
10.4 How do I implement an ACL ban list? . . . 89
10.5 How do I block specic users or groups from accessing my cache? . . . 89
10.5.1 Ident . . . 89
10.5.2 Proxy Authentication . . . 89
10.6 Do you have a CGI program which lets users change their own proxy passwords? . . . 90
10.7 Is there a way to do ident lookups only for a certain host and compare the result with a userlist in squid.conf? . . . 90
10.8 Common Mistakes . . . 90
10.8.1 And/Or logic . . . 90
10.8.2 allow/deny mixups . . . 91
10.8.3 Dierences between *src* and *srcdomain* ACL types. . . . 92
10.9 I set up my access controls, but they don't work! why? . . . 92
10.10 Proxy-authentication and neighbor caches . . . 92
10.11 Is there an easy way of banning all Destination addresses except one? . . . 93
10.12 Does anyone have a ban list of porn sites and such? . . . 93
10.13 Squid doesn't match my subdomains . . . 93
10.14 Why does Squid deny some port numbers? . . . 94
10.15 Does Squid support the use of a database such as mySQL for storing the ACL list? . . . 94
10.16 How can I allow a single address to access a specic URL? . . . 94
10.17 How can I allow some clients to use the cache at specic times? . . . 95
10.18 How can I allow some users to use the cache at specic times? . . . 95
10.19 Problems with IP ACL's that have complicated netmasks . . . 95
10.20 Can I set up ACL's based on MAC address rather than IP? . . . 95
10.21 Debugging ACLs . . . 96
10.22 Can I limit the number of connections from a client? . . . 96
10.23 I'm trying to deny *foo.com*, but it's not working. . . . 96
10.24 I want to customize, or make my own error messages. . . . 97
10.25 I want to use local time zone in error messages . . . 97

11 Troubleshooting **97**

11.1 Why am I getting "Proxy Access Denied?" . . . 97

11.2 I can't get `local_domain` to work; *Squid* is caching the objects from my local servers. . . . 97

11.3 I get `Connection Refused` when the cache tries to retrieve an object located on a sibling, even though the sibling thinks it delivered the object to my cache. . . . 98

11.4 Running out of ledescriptors . . . 98

11.4.1 Linux . . . 98

11.4.2 Solaris . . . 98

11.4.3 FreeBSD . . . 99

11.4.4 General BSD . . . 99

11.4.5 Recongure afterwards . . . 100

11.5 What are these strange lines about removing objects? . . . 100

11.6 Can I change a Windows NT FTP server to list directories in Unix format? . . . 101

11.7 Why am I getting "Ignoring MISS from non-peer x.x.x.x?" . . . 101

11.8 DNS lookups for domain names with underscores (_) always fail. . . . 101

11.9 Why does Squid say: "Illegal character in hostname; underscores are not allowed?' . . . 102

11.10Why am I getting access denied from a sibling cache? . . . 102

11.11Cannot bind socket FD NN to *:8080 (125) Address already in use . . . 103

11.12icpDetectClientClose: ERROR xxx.xxx.xxx.xxx: (32) Broken pipe . . . 103

11.13icpDetectClientClose: FD 135, 255 unexpected bytes . . . 104

11.14Does Squid work with NTLM Authentication? . . . 104

11.15The *default* parent option isn't working! . . . 104

11.16"Hot Mail" complains about: Intrusion Logged. Access denied. . . . 105

11.17My Squid becomes very slow after it has been running for some time. . . . 105

11.18WARNING: Failed to start 'dnsserver' . . . 105

11.19Sending in Squid bug reports . . . 106

11.19.1 crashes and core dumps . . . 106

11.20Debugging Squid . . . 109

11.21FATAL: ipcache_init: DNS name lookup tests failed . . . 109

11.22FATAL: Failed to make swap directory /var/spool/cache: (13) Permission denied . . . 110

11.23FATAL: Cannot open HTTP Port . . . 110

11.24FATAL: All redirectors have exited! . . . 110

11.25FATAL: le _map_allocate: Exceeded lemap limit . . . 110

11.26FATAL: You've run out of swap le numbers. . . . 111

11.27I am using up over 95% of the lemap bits?!! . . . 111

11.28FATAL: Cannot open /usr/local/squid/logs/access.log: (13) Permission denied . . . 112

11.29When using a username and password, I can not access some les. . . . 112

11.30pingerOpen: icmp_sock: (13) Permission denied . . . 112
11.31What is a forwarding loop? . . . 112
11.32accept failure: (71) Protocol error . . . 113
11.33storeSwapInFileOpened: ... Size mismatch . . . 113
11.34Why do I get *fwdDispatch: Cannot retrieve 'https://www.buy.com/corp/ordertracking.asp'* . . . 114
11.35Squid can't access URLs like http://3626046468/ab2/cybercards/moreinfo.html . . . 114
11.36I get a lot of "URI has whitespace" error messages in my cache log, what should I do? . . . 115
11.37commBind: Cannot bind socket FD 5 to 127.0.0.1:0: (49) Can't assign requested address . . . 115
11.38Unknown cache_dir type '/var/squid/cache' . . . 115
11.39unrecognized: 'cache_dns_program /usr/local/squid/bin/dnsserver' . . . 116
11.40Is *dns_defnames* broken in 2.3.STABLE1 and STABLE2? . . . 116
11.41What does *sslReadClient: FD 14: read failure: (104) Connection reset by peer* mean? . . . 116
11.42What does *Connection refused* mean? . . . 116
11.43squid: ERROR: no running copy . . . 117
11.44FATAL: getgrnam failed to nd groupid for eective group 'nogroup' . . . 117
11.45"Unsupported Request Method and Protocol" for *https* URLs. . . . 117
11.46Squid uses 100% CPU . . . 118
11.47Webmin's *cachemgr.cgi* crashes the operating system . . . 118
11.48Segment Violation at startup or upon rst request . . . 118
11.49urlParse: Illegal character in hostname 'proxy.mydomain.com:8080proxy.mydomain.com' . . . 119
11.50Requests for international domain names does not work . . . 119
11.51Why do I sometimes get "Zero Sized Reply"? . . . 119

12 How does Squid work? 121

12.1 What are cachable objects? . . . 121
12.2 What is the ICP protocol? . . . 121
12.3 What is the *dnsserver*? . . . 121
12.4 What is the *ftpget* program for? . . . 121
12.5 FTP PUT's don't work! . . . 121
12.6 What is a cache hierarchy? What are parents and siblings? . . . 121
12.7 What is the Squid cache resolution algorithm? . . . 122
12.8 What features are Squid developers currently working on? . . . 122
12.9 Tell me more about Internet trac workloads . . . 122
12.10What are the tradeos of caching with the NLANR cache system? . . . 122
12.11Where can I nd out more about rewalls? . . . 123
12.12What is the "Storage LRU Expiration Age?" . . . 123
12.13What is "Failure Ratio at 1.01; Going into hit-only-mode for 5 minutes"? . . . 123

12.14Does squid periodically re-read its conguration le? . . . 123
12.15How does *unlinkd* work? . . . 123
12.16What is an icon URL? . . . 124
12.17Can I make my regular FTP clients use a Squid cache? . . . 124
12.18Why is the select loop average time so high? . . . 124
12.19How does Squid deal with Cookies? . . . 125
12.20How does Squid decide when to refresh a cached object? . . . 125
12.20.1 Squid-1.1 and Squid-1.NOVM algorithm . . . 126
12.20.2 Squid-2 algorithm . . . 126
12.21What exactly is a *deferred read*? . . . 127
12.22Why is my cache's inbound trac equal to the outbound trac? . . . 127
12.23How come some objects do not get cached? . . . 128
12.24What does *keep-alive ratio* mean? . . . 129
12.25How does Squid's cache replacement algorithm work? . . . 129
12.25.1 Squid 1.1 . . . 130
12.25.2 Squid 2 . . . 130
12.26What are private and public keys? . . . 130
12.27What is FORW_VIA_DB for? . . . 130
12.28Does Squid send packets to port 7 (echo)? If so, why? . . . 130
12.29What does "WARNING: Reply from unknown nameserver [a.b.c.d]" mean? . . . 131
12.30How does Squid distribute cache les among the available directories? . . . 131
12.31Why do I see negative byte hit ratio? . . . 131
12.32What does "Disabling use of private keys" mean? . . . 132
12.33What is a half-closed ledescriptor? . . . 132
12.34What does –enable-heap-replacement do? . . . 133
12.35Why is actual lesystem space used greater than what Squid thinks? . . . 133
12.36How do *positive_dns_ttl* and *negative_dns_ttl* work? . . . 133
12.37What does *swapin MD5 mismatch* mean? . . . 134
12.38What does *failed to unpack swaple meta data* mean? . . . 134
12.39Why doesn't Squid make *ident* lookups in interception mode? . . . 134
12.40dnsSubmit: queue overload, rejecting blah . . . 135
12.41What are FTP passive connections? . . . 135

13 Multicast **135**

13.1 What is Multicast? . . . 135
13.2 How do I know if my network has multicast? . . . 135
13.3 Should I be using Multicast ICP? . . . 136

13.4 How do I congure Squid to send Multicast ICP queries? . . . 136
13.5 How do I know what Multicast TTL to use? . . . 137
13.6 How do I congure Squid to receive and respond to Multicast ICP? . . . 137

14 System-Dependent Weirdnesses 138

14.1 Solaris . . . 138
14.1.1 TCP incompatibility? . . . 138
14.1.2 select() . . . 138
14.1.3 malloc . . . 138
14.1.4 DNS lookups and *nscd* . . . 138
14.1.5 DNS lookups and */etc/nsswitch.conf* . . . 139
14.1.6 DNS lookups and NIS . . . 139
14.1.7 Tuning . . . 140
14.1.8 disk write error: (28) No space left on device . . . 140
14.1.9 Solaris X86 and IPFilter . . . 140
14.1.10 Changing the directory lookup cache size . . . 141
14.1.11 The priority_paging algorithm . . . 141
14.2 FreeBSD . . . 141
14.2.1 T/TCP bugs . . . 141
14.2.2 mbuf size . . . 142
14.2.3 Dealing with NIS . . . 143
14.2.4 FreeBSD 3.3: The lo0 (loop-back) device is not congured on startup . . . 143
14.2.5 FreeBSD 3.x or newer: Speed up disk writes using Softupdates . . . 143
14.2.6 Internal DNS problems with jail environment . . . 144
14.3 OSF1/3.2 . . . 144
14.4 BSD/OS . . . 144
14.4.1 gcc/yacc . . . 144
14.4.2 process priority . . . 144
14.5 Linux . . . 145
14.5.1 Cannot bind socket FD 5 to 127.0.0.1:0: (49) Can't assign requested address . . . 145
14.5.2 FATAL: Don't run Squid as root, set 'cache_eective _user'! . . . 145
14.5.3 Large ACL lists make Squid slow . . . 145
14.5.4 gethostbyname() leaks memory in RedHat 6.0 with glibc 2.1.1. . . . 145
14.5.5 assertion failed: StatHist.c:91: 'statHistBin(H, max) == H->capacity - 1' on Alpha system. . . . 145
14.5.6 tools.c:605: storage size of 'rl' isn't known . . . 146
14.5.7 Can't connect to some sites through Squid . . . 146

14.6 HP-UX . . . 147
14.6.1 StatHist.c:74: failed assertion 'statHistBin(H, min) == 0' . . . 147
14.7 IRIX . . . 147
14.7.1 *dnsserver* always returns 255.255.255.255 . . . 147
14.8 SCO-UNIX . . . 147
14.9 AIX . . . 147
14.9.1 "shmat failed" errors with *diskd* . . . 147
14.9.2 Core dumps when squid process grows to 256MB . . . 147

15 Redirectors **148**
15.1 What is a redirector? . . . 148
15.2 Why use a redirector? . . . 148
15.3 How does it work? . . . 148
15.4 Do you have any examples? . . . 148
15.5 Can I use the redirector to return HTTP redirect messages? . . . 149
15.6 FATAL: All redirectors have exited! . . . 149
15.7 Redirector interface is broken re IDENT values . . . 149

16 Cache Digests **150**
16.1 What is a Cache Digest? . . . 150
16.2 How and why are they used? . . . 150
16.3 What is the theory behind Cache Digests? . . . 150
16.3.1 Adding a Key . . . 150
16.3.2 Querying a Key . . . 151
16.3.3 Deleting a Key . . . 151
16.4 How is the size of the Cache Digest in Squid determined? . . . 151
16.5 What hash functions (and how many of them) does Squid use? . . . 152
16.6 How are objects added to the Cache Digest in Squid? . . . 152
16.7 Does Squid support deletions in Cache Digests? What are dis/deltas? . . . 152
16.8 When and how often is the local digest built? . . . 153
16.9 How are Cache Digests transferred between peers? . . . 153
16.10 How and where are Cache Digests stored? . . . 153
16.10.1 Cache Digest built locally . . . 153
16.10.2 Cache Digest fetched from peer . . . 154
16.11 How are the Cache Digest statistics in the Cache Manager to be interpreted? . . . 154
16.12 What are False Hits and how should they be handled? . . . 155
16.13 How can Cache Digest related activity be traced/debugged? . . . 156
16.13.1 Enabling Cache Digests . . . 156

16.13.2 What do the access.log entries look like? . . . 156
16.13.3 What does a False Hit look like? . . . 156
16.13.4 How is the cause of a False Hit determined? . . . 156
16.13.5 Use The Source . . . 157
16.14 What about ICP? . . . 157
16.15 Is there a Cache Digest Specication? . . . 157
16.16 Would it be possible to stagger the timings when cache_digests are retrieved from peers? . . . 157

17 Interception Caching/Proxying 158

17.1 Interception caching for Solaris, SunOS, and BSD systems . . . 159
17.1.1 Install IP Filter . . . 159
17.1.2 Congure ipnat . . . 159
17.1.3 Congure Squid . . . 159
17.2 Interception caching with Linux 2.0 and ipfwadm . . . 160
17.3 Interception caching with Linux 2.2 and ipchains . . . 162
17.4 Interception caching with Linux 2.4 and netlter . . . 164
17.5 Interception caching with Cisco routers . . . 164
17.5.1 possible bugs . . . 165
17.6 Interception caching with LINUX 2.0.29 and CISCO IOS 11.1 . . . 166
17.7 The cache is trying to connect to itself... . . . 167
17.8 Interception caching with FreeBSD . . . 168
17.9 Interception caching with ACC Tigris digital access server . . . 169
17.10 "Connection reset by peer" and Cisco policy routing . . . 170
17.11 WCCP - Web Cache Coordination Protocol . . . 170
17.11.1 Does Squid support WCCP? . . . 170
17.11.2 Conguring your Router . . . 170
17.11.3 IOS 12.x problems . . . 171
17.11.4 Conguring FreeBSD . . . 171
17.11.5 Conguring Linux 2.2 . . . 172
17.11.6 Conguring Others . . . 173
17.12 Can someone tell me what version of cisco IOS WCCP is added in? . . . 173
17.13 What about WCCPv2? . . . 173
17.14 Interception caching with Foundry L4 switches . . . 173
17.15 Can I use *proxy_auth* with interception? . . . 174

18 SNMP 175

18.1 Does Squid support SNMP? . . . 175
18.2 Enabling SNMP in Squid . . . 175

18.3 Conguring Squid 2.2 . . . 175
18.4 Conguring Squid 2.1 . . . 176
18.5 How can I query the Squid SNMP Agent . . . 176
18.6 What can I use SNMP and Squid for? . . . 177
18.7 How can I use SNMP with Squid? . . . 177
18.8 Where can I get more information/discussion about Squid and SNMP? . . . 177
18.9 Monitoring Squid with MRTG . . . 177

19 Squid version 2 **178**

19.1 What are the new features? . . . 178
19.2 How do I congure 'ssl _proxy' now? . . . 178
19.3 Logle rotation doesn't work with Async I/O . . . 179
19.4 Adding a new cache disk . . . 179
19.5 Squid 2 performs badly on Linux . . . 179
19.6 How do I congure proxy authentication with Squid-2? . . . 179
19.7 Why does proxy-auth reject all users with Squid-2.2? . . . 180
19.8 Delay Pools . . . 180
19.8.1 How can I limit Squid's total bandwidth to, say, 512 Kbps? . . . 182
19.8.2 How to limit a single connection to 128 Kbps? . . . 182
19.8.3 How do you personally use delay pools? . . . 182
19.8.4 Where else can I nd out about delay pools? . . . 184
19.9 Can I preserve my cache when upgrading from 1.1 to 2? . . . 186
19.10Customizable Error Messages . . . 187
19.11My squid.conf from version 1.1 doesn't work! . . . 189

20 httpd-accelerator mode **190**

20.1 What is the httpd-accelerator mode? . . . 190
20.2 How do I set it up? . . . 190
20.3 When using an httpd-accelerator, the port number for redirects is wrong . . . 191

21 Related Software **191**

21.1 Clients . . . 191
21.1.1 Wget . . . 191
21.1.2 echoping . . . 192
21.2 Logle Analysis . . . 192
21.3 Conguration Tools . . . 192
21.3.1 3Dhierarchy.pl . . . 192
21.4 Squid add-ons . . . 192

21.4.1 transproxy . . . 192
21.4.2 Iain's redirector package . . . 192
21.4.3 Junkbusters . . . 192
21.4.4 Squirm . . . 192
21.4.5 chpasswd.cgi . . . 193
21.4.6 jesred . . . 193
21.4.7 squidGuard . . . 193
21.4.8 Central Squid Server . . . 193
21.5 Ident Servers . . . 193

22 DISKD **193**

22.1 What is DISKD? . . . 193
22.2 Does it perform better? . . . 193
22.3 How do I use it? . . . 194
22.4 FATAL: Unknown cache_dir type 'diskd' . . . 194
22.5 If I use DISKD, do I have to wipe out my current cache? . . . 194
22.6 How do I congure message queues? . . . 194
22.6.1 FreeBSD . . . 195
22.6.2 OpenBSD . . . 195
22.6.3 Digital Unix . . . 195
22.6.4 Linux . . . 196
22.6.5 Solaris . . . 196
22.7 How do I congure shared memory? . . . 197
22.7.1 FreeBSD . . . 197
22.7.2 Digital Unix . . . 197
22.7.3 Linux . . . 198
22.7.4 Solaris . . . 198
22.8 Sometimes shared memory and message queues aren't released when Squid exits. . . . 198
22.9 What are the Q1 and Q2 parameters? . . . 199

23 Authentication **199**

23.1 How does Proxy Authentication work in Squid? . . . 199
23.2 How do I use authentication in access controls? . . . 200
23.3 Does Squid cache authentication lookups? . . . 200
23.4 Are passwords stored in clear text or encrypted? . . . 200
23.5 How do I use the Winbind authenticators? . . . 201
23.5.1 Supported Samba Releases . . . 201
23.5.2 Congure Samba . . . 201

23.5.3 Congure Squid 203

24 Terms and Denitions **204**

24.1 Neighbor 204

24.2 Regular Expression 204

25 Security Concerns **204**

25.1 Open-access proxies 204

25.2 Mail relaying 205

You can download the FAQ as *PDF* `<FAQ.pdf>`, *compressed Postscript* `<FAQ.ps.gz>`, *plain text* `<FAQ.txt>`, *linuxdoc SGML source* `<FAQ.sgml>` or as a *compressed tar of HTML* `<FAQ.tar.gz>`.

1 About Squid, this FAQ, and other Squid information resources

1.1 What is Squid?

Squid is a high-performance proxy caching server for web clients, supporting FTP, gopher, and HTTP data objects. Unlike traditional caching software, Squid handles all requests in a single, non-blocking, I/O-driven process.

Squid keeps meta data and especially hot objects cached in RAM, caches DNS lookups, supports non-blocking DNS lookups, and implements negative caching of failed requests.

Squid supports SSL, extensive access controls, and full request logging. By using the lightweight Internet Cache Protocol, Squid caches can be arranged in a hierarchy or mesh for additional bandwidth savings.

Squid consists of a main server program *squid*, a Domain Name System lookup program *dnsserver*, some optional programs for rewriting requests and performing authentication, and some management and client tools. When *squid* starts up, it spawns a congurable number of *dnsserver* processes, each of which can perform a single, blocking Domain Name System (DNS) lookup. This reduces the amount of time the cache waits for DNS lookups.

Squid is derived from the ARPA-funded *Harvest project* `<http://webharvest.sourceforge.net/ng/>`.

1.2 What is Internet object caching?

Internet object caching is a way to store requested Internet objects (i.e., data available via the HTTP, FTP, and gopher protocols) on a system closer to the requesting site than to the source. Web browsers can then use the local Squid cache as a proxy HTTP server, reducing access time as well as bandwidth consumption.

1.3 Why is it called Squid?

Harris' Lament says, "All the good ones are taken."

We needed to distinguish this new version from the Harvest cache software. Squid was the code name for initial development, and it stuck.

1.4 What is the latest version of Squid?

Squid is updated often; please see *the Squid home page* <http://www.squid-cache.org/> for the most recent versions.

1.5 Who is responsible for Squid?

Squid is the result of eorts by numerous individuals from the Internet community. *Duane Wessels* <mailto:wessels@squid-cache.org> of the National Laboratory for Applied Network Research (funded by the National Science Foundation) leads code development. Please see *the CONTRIBUTORS le* <http://www.squid-cache.org/CONTRIBUTORS> for a list of our excellent contributors.

1.6 Where can I get Squid?

You can download Squid via FTP from *the primary FTP site* <ftp://ftp.squid-cache.org/pub/> or one of the many worldwide *mirror sites* <http://www.squid-cache.org/mirrors.html>.

Many sushi bars also have Squid.

1.7 What Operating Systems does Squid support?

The software is designed to operate on any modern Unix system, and is known to work on at least the following platforms:

Linux

FreeBSD

NetBSD

BSDI

Mac OS/X

OSF and Digital Unix

IRIX

SunOS/Solaris

NeXTStep

SCO Unix

AIX

HP-UX

2.14

For more specic information, please see *platforms.html* <http://www.squid-cache.org/platforms.html>. If you encounter any platform-specic problems, please let us know by registering a entry in our *bug database* <http://www.squid-cache.org/bugs/>.

1.8 Does Squid run on Windows NT?

Recent versions of Squid will *compile and run* on Windows/NT with the *Cygwin* <http://cygwin.com/> / *GnuWin32* <http://gnuwin32.sourceforge.net/> packages.

Guido Serassio <http://www.serassio.it/SquidNT/index.htm> have Squid NT pages and is actively working on having the needed changes integrated into the standard Squid distribution. Partially based on earlier NT port by *Romeo Anghelache* <http://www.phys-iasi.ro/users/romeo/squidnt.htm>.

LogiSense <http://www.logisense.com/> has ported Squid to Windows NT and sells a supported version. You can also download the source from *their FTP site* <ftp://ftp.logisense.com/cachexpress/>. Thanks to LogiSense for making the code available as required by the GPL terms.

1.9 What Squid mailing lists are available?

squid-users@squid-cache.org: general discussions about the Squid cache software. Subscribe via *squid-users-subscribe@squid-cache.org*.

Previous messages are available for browsing at *the Squid Users Archive* <http://www.squid-cache.org/mail-archive/squid-users/>, and also at *theaimsgroup.com* <http://marc.theaimsgroup.com/?l=squid-users&r=1&w=2>.

squid-users-digest: digested (daily) version of above. Subscribe via *squid-users-digest-subscribe@squid-cache.org*.

squid-announce@squid-cache.org: A receive-only list for announcements of new versions. Subscribe via *squid-announce-subscribe@squid-cache.org*.

squid-bugs@squid-cache.org: A closed list for sending us bug reports. Bug reports received here are given priority over those mentioned on squid-users.

squid@squid-cache.org: A closed list for sending us feed-back and ideas.

squid-faq@squid-cache.org: A closed list for sending us feed-back, updates, and additions to the Squid FAQ.

We also have a few other mailing lists which are not strictly Squid-related.

cache-snmp@ircache.net: A public list for discussion of Web Caching and SNMP issues and developments. Eventually we hope to put forth a standard Web Caching MIB.

icp-wg@ircache.net: Mostly-idle mailing list for the nonexistent ICP Working Group within the IETF. It may be resurrected some day, you never know!

1.10 I can't gure out how to unsubscribe from your mailing list.

All of our mailing lists have "-subscribe" and "-unsubscribe" addresses that you must use for subscribe and unsubscribe requests. To unsubscribe from the squid-users list, you send a message to *squid-users-unsubscribe@squid-cache.org*.

1.11 What Squid web pages are available?

Several Squid and Caching-related web pages are available:

The Squid home page <http://www.squid-cache.org/> for information on the Squid software

The IRCache Mesh <http://www.ircache.net/Cache/> gives information on our operational mesh of caches.

The Squid FAQ <http://www.squid-cache.org/Doc/FAQ/> (uh, you're reading it).

Oskar's Squid Users Guide <http://squid-docs.sourceforge.net/latest/html/book1.html>.

The Information Resource Caching FAQ <http://www.ircache.net/Cache/FAQ/>

Squid Programmers Guide <http://www.squid-cache.org/Doc/Prog-Guide/prog-guide.html>. Yeah, its extremely incomplete. I assure you this is the most recent version.

Web Caching Resources <http://www.web-cache.com>

Squid-1.0 Release Notes </Versions/1.0/Release-Notes-1.0.txt>

Squid-1.1 Release Notes </Versions/1.1/Release-Notes-1.1.txt>

Tutorial on Conguring Hierarchical Squid Caches <http://www.squid-cache.org/Doc/Hierarchy-Tutorial/>

RFC 2186 <ftp://ftp.isi.edu/in-notes/rfc2616.txt> ICPv2 – Protocol

RFC 2187 <ftp://ftp.isi.edu/in-notes/rfc2187.txt> ICPv2 – Application

RFC 1016 <ftp://ftp.isi.edu/in-notes/rfc1016.txt>

1.12 Does Squid support SSL/HTTPS/TLS?

As of version 2.5, Squid can terminate SSL connections. This is perhaps only useful in a surrogate (http accelerator) conguration. You must run congure with *–enable-ssl*. See *https_port* in squid.conf for more information.

Squid also supports these encrypted protocols by "tunelling" trac between clients and servers. In this case, Squid can relay the encrypted bits between a client and a server.

Normally, when your browser comes across an *https* URL, it does one of two things:

1. The browser opens an SSL connection directly to the origin server.
2. The browser tunnels the request through Squid with the *CONNECT* request method.

The *CONNECT* method is a way to tunnel any kind of connection through an HTTP proxy. The proxy doesn't understand or interpret the contents. It just passes bytes back and forth between the client and server. For the gory details on tunnelling and the CONNECT method, please see *RFC 2817* <ftp://ftp.isi.edu/in-notes/rfc2817.txt> and *Tunneling TCP based protocols through Web proxy servers* <http://www.web-cache.com/Writings/Internet-Drafts/draft-luotonen-web-proxy-tunneling-01.txt> (expired).

1.13 What's the legal status of Squid?

Squid is *copyrighted* <squid-copyright.txt> by the University of California San Diego. Squid uses some *code developed by others* <squid-credits.txt>.

Squid is *Free Software* <http://www.gnu.org/philosophy/free-sw.html>.

Squid is licensed under the terms of the *GNU General Public License* <http://www.gnu.org/copyleft/gpl.html>.

1.14 Is Squid year-2000 compliant?

We think so. Squid uses the Unix time format for all internal time representations. Potential problem areas are in printing and parsing other time representations. We have made the following xes in to address the year 2000:

cache.log timestamps use 4-digit years instead of just 2 digits.

parse_rfc1123() assumes years less than "70" are after 2000.

parse_iso3307_time() checks all four year digits.

Year-2000 xes were applied to the following Squid versions:

squid-2.1 </Versions/v2/2.1/>: Year parsing bug xed for dates in the "Wed Jun 9 01:29:59 1993 GMT" format (Richard Kettlewell).

squid-1.1.22: Fixed likely year-2000 bug in ftpget's timestamp parsing (Henrik Nordstrom).

squid-1.1.20: Misc xes (Arjan de Vet).

Patches:

Richard's lib/rfc1123.c patch <../Y2K/patch3>. If you are still running 1.1.X, then you should apply this patch to your source and recompile.

Henrik's src/ftpget.c patch <../Y2K/patch2>.

Arjan's lib/rfc1123.c patch <../Y2K/patch1>.

Squid-2.2 and earlier versions have a *New Year bug* <http://www.squid-cache.org/Versions/v2/2.2/bugs/index.html#sq
This is not strictly a Year-2000 bug; it would happen on the rst day of any year.

1.15 Can I pay someone for Squid support?

Yep. Please see the *commercial support page* </Support/services.html>.

1.16 Squid FAQ contributors

The following people have made contributions to this document:

Jonathan Larmour <mailto:JLarmour@origin-at.co.uk>

Cord Beermann <mailto:cord@Wunder-Nett.org>

Tony Sterrett <mailto:tony@nlanr.net>

Gerard Hynes <mailto:ghynes@compusult.nf.ca>

Katayama, Takeo <mailto:tkatayam@pi.titech.ac.jp>

Duane Wessels <mailto:wessels@ircache.net>

K Clay <mailto:kc@caida.org>

Paul Southworth <mailto:pauls@etext.org>

Oskar Pearson <mailto:oskar@is.co.za>

Ong Beng Hui <mailto:ongbh@zpoprp.zpo.dec.com>

Torsten Sturm <mailto:torsten.sturm@axis.de>

James R Grinter <mailto:jrg@blodwen.demon.co.uk>

Rodney van den Oever <mailto:roever@nse.simac.nl>

Kolics Bertold <mailto:bertold@tohotom.vein.hu>

Carson Gaspar <mailto:carson@cugc.org>

Michael O'Reilly <mailto:michael@metal.iinet.net.au>

Hume Smith <mailto:hclsmith@tallships.istar.ca>

Richard Ayres <mailto:RichardA@noho.co.uk>

John Saunders <mailto:John.Saunders@scitec.com.au>

Miquel van Smoorenburg <mailto:miquels@cistron.nl>

David J N Begley <mailto:david@avarice.nepean.uws.edu.au>

Kevin Sartorelli <mailto:SarKev@topnz.ac.nz>

Andreas Doering <mailto:doering@usf.uni-kassel.de>

Mark Visser <mailto:mark@cal026031.student.utwente.nl>

tom minchin <mailto:tom@interact.net.au>

Jens-S. Vöckler <mailto:voeckler@rvs.uni-hannover.de>

Andre Albsmeier <mailto:andre.albsmeier@mchp.siemens.de>

Doug Nazar <mailto:nazard@man-assoc.on.ca>

Henrik Nordstrom <mailto:hno@squid-cache.org>

Mark Reynolds <mailto:mark@rts.com.au>

Arjan de Vet <mailto:Arjan.deVet@adv.IAEhv.nl>

Peter Wemm <mailto:peter@spinner.dialix.com.au>

John Line <mailto:webadm@info.cam.ac.uk>

Jason Armistead <mailto:ARMISTEJ@oeca.otis.com>

Chris Tilbury <mailto:cudch@csv.warwick.ac.uk>

Je Madison <mailto:jeff@sisna.com>

Mike Batchelor <mailto:mbatchelor@citysearch.com>

Bill Bogstad <mailto:bogstad@pobox.com>

Radu Greab <mailto:radu at netsoft dot ro>

F.J. Bosscha <mailto:f.j.bosscha@nhl.nl>

Brian Feeny <mailto:signal@shreve.net>

Martin Lyons <mailto:Support@dnet.co.uk>

David Luyer <mailto:david@luyer.net>

Chris Foote <mailto:chris@senet.com.au>

Jens Elkner <mailto:elkner@wotan.cs.Uni-Magdeburg.DE>

Simon White <mailto:simon@mtds.com>

Jerry Murdock <mailto: jmurdoc at itraktech dot com>

Gerard Eviston <mailto: geviston at bigpond dot net dot au>

Rob Poe <mailto: rob at poeweb dot com>

Please send corrections, updates, and comments to: *squid-faq@squid-cache.org* <mailto:squid-faq@squid-cache.org>.

1.17 About This Document

This document is copyrighted (2000) by Duane Wessels.

This document was written in SGML and converted with the *SGML-Tools package* <http://www.sgmltools.org/>.

Most current version of this document can always be found at *http://www.squid-cache.org/Doc/FAQ/* <http://www.squid-cache.org/Doc/FAQ/> in HTML, Plain Text, Postscript and SGML formats.

1.17.1 Want to contribute? Please write in SGML...

It is easier for us if you send us text which is close to "correct" SGML. The SQUID FAQ currently uses the LINUXDOC DTD. Its probably easiest to follow examples in the this le. Here are the basics:

Use the <url> tag for links, instead of HTML <A HREF ...>

```
<url url="http://www.squid-cache.org" name="Squid Home Page">
```

Use for emphasis, cong options, and pathnames:

```
<em>usr/local/squid/etc/squid.conf</em>
<em/cache_peer/
```

Here is how you do lists:

```
<itemize>
<item>foo
<item>bar
</itemize>
```

Use <verb>, just like HTML's <PRE> to show unformatted text.

2 Getting and Compiling Squid

2.1 Which le do I download to get Squid?

You must download a source archive le of the form squid-x.y.z-src.tar.gz (eg, squid-1.1.6-src.tar.gz) from *the Squid home page* <http://www.squid-cache.org/>, or. *the Squid FTP site* <ftp://www.squid-cache.org/pub/>. Context dis are available for upgrading to new versions. These can be applied with the *patch* program (available from *the GNU FTP site* <ftp://ftp.gnu.org/gnu/patch>).

2.2 How do I compile Squid?

For **Squid-1.0** and **Squid-1.1** versions, you can just type *make* from the top-level directory after unpacking the source les. For example:

```
% tar xzf squid-1.1.21-src.tar.gz
% cd squid-1.1.21
% make
```

For **Squid-2** you must run the *congure* script yourself before running *make*:

```
% tar xzf squid-2.0.RELEASE-src.tar.gz
% cd squid-2.0.RELEASE
% ./configure
% make
```

2.3 What kind of compiler do I need?

To compile Squid, you will need an ANSI C compiler. Almost all modern Unix systems come with pre-installed compilers which work just ne. The old *SunOS* compilers do not have support for ANSI C, and the Sun compiler for *Solaris* is a product which must be purchased separately.

If you are uncertain about your system's C compiler, The GNU C compiler is available at *the GNU FTP site* <ftp://ftp.gnu.org/gnu/gcc>. In addition to gcc, you may also want or need to install the *binutils* package.

2.4 What else do I need to compile Squid?

You will need *Perl* <http://www.perl.com/> installed on your system.

2.5 Do you have pre-compiled binaries available?

The developers do not have the resources to make pre-compiled binaries available. Instead, we invest eort into making the source code very portable. Some people have made binary packages available. Please see our *Platforms Page* <http://www.squid-cache.org/platforms.html>.

The *SGI Freeware* <http://freeware.sgi.com/> site has pre-compiled packages for SGI IRIX.

Squid binaries for *FreeBSD on Alpha and Intel* <http://www.freebsd.org/cgi/ports.cgi?query=squid-2&stype=all>.

Squid binaries for *NetBSD on everything* <ftp://ftp.netbsd.org/pub/NetBSD/packages/pkgsrc/www/squid/README.html

Gurkan Sengun has some *Sparc/Solaris packages* <http://www.linuks.mine.nu/solaris/> available.

2.6 How do I apply a patch or a di?

You need the `patch` program. You should probably duplicate the entire directory structure before applying the patch. For example, if you are upgrading from squid-1.1.10 to 1.1.11, you would run these commands:

```
cd squid-1.1.10
mkdir ../squid-1.1.11
find . -depth -print | cpio -pdv ../squid-1.1.11
cd ../squid-1.1.11
patch < /tmp/diff-1.1.10-1.1.11
```

After the patch has been applied, you must rebuild Squid from the very beginning, i.e.:

```
make realclean
./configure
make
make install
```

Note, In later distributions (Squid 2), 'realclean' has been changed to 'distclean'.

If patch keeps asking for a le name, try adding "-p0":

```
patch -p0 < filename
```

If your `patch` program seems to complain or refuses to work, you should get a more recent version, from the *GNU FTP site* `<ftp://ftp.gnu.ai.mit.edu/pub/gnu/>`, for example.

2.7 *congure* options

The congure script can take numerous options. The most useful is `--prefix` to install it in a dierent directory. The default installation directory is */usr/local/squid/*. To change the default, you could do:

```
% cd squid-x.y.z
% ./configure --prefix=/some/other/directory/squid
```

Type

```
% ./configure --help
```

to see all available options. You will need to specify some of these options to enable or disable certain features. Some options which are used often include:

```
--prefix=PREFIX         install architecture-independent files in PREFIX
                        [/usr/local/squid]
--enable-dlmalloc[=LIB] Compile & use the malloc package by Doug Lea
--enable-gnuregex       Compile GNUregex
--enable-splaytree      Use SPLAY trees to store ACL lists
--enable-xmalloc-debug  Do some simple malloc debugging
--enable-xmalloc-debug-trace
                        Detailed trace of memory allocations
--enable-xmalloc-statistics
                        Show malloc statistics in status page
```

```
--enable-carp           Enable CARP support
--enable-async-io       Do ASYNC disk I/O using threads
--enable-icmp           Enable ICMP pinging
--enable-delay-pools    Enable delay pools to limit bandwith usage
--enable-mem-gen-trace  Do trace of memory stuff
--enable-useragent-log  Enable logging of User-Agent header
--enable-kill-parent-hack
                        Kill parent on shutdown
--enable-snmp           Enable SNMP monitoring
--enable-cachemgr-hostname[=hostname]
                        Make cachemgr.cgi default to this host
--enable-arp-acl        Enable use of ARP ACL lists (ether address)
--enable-htpc           Enable HTCP protocol
--enable-forw-via-db    Enable Forw/Via database
--enable-cache-digests  Use Cache Digests
                        see http://www.squid-cache.org/Doc/FAQ/FAQ-16.html
--enable-err-language=lang
                        Select language for Error pages (see errors dir)
```

2.8 undened reference to __inet_ntoa

by *Kevin Sartorelli* <mailto:SarKev@topnz.ac.nz> and *Andreas Doering* <mailto:doering@usf.uni-kassel.de>.

Probably you've recently installed bind 8.x. There is a mismatch between the header les and DNS library that Squid has found. There are a couple of things you can try.

First, try adding `-lbind` to *XTRA_LIBS* in *src/Makele* . If `-lresolv` is already there, remove it.

If that doesn't seem to work, edit your *arpa/inet.h* le and comment out the following:

```
#define inet_addr               __inet_addr
#define inet_aton               __inet_aton
#define inet_lnaof              __inet_lnaof
#define inet_makeaddr           __inet_makeaddr
#define inet_neta               __inet_neta
#define inet_netof              __inet_netof
#define inet_network            __inet_network
#define inet_net_ntop           __inet_net_ntop
#define inet_net_pton           __inet_net_pton
#define inet_ntoa               __inet_ntoa
#define inet_pton               __inet_pton
#define inet_ntop               __inet_ntop
#define inet_nsap_addr          __inet_nsap_addr
#define inet_nsap_ntoa          __inet_nsap_ntoa
```

2.9 How can I get true DNS TTL info into Squid's IP cache?

If you have source for BIND, you can modify it as indicated in the di below. It causes the global variable _dns_ttl_ to be set with the TTL of the most recent lookup. Then, when you compile Squid, the congure script will look for the _dns_ttl_ symbol in libresolv.a. If found, dnsserver will return the TTL value for every lookup.

This hack was contributed by *Endre Balint Nagy* <mailto:bne@CareNet.hu>.

```
diff -ru bind-4.9.4-orig/res/gethnamaddr.c bind-4.9.4/res/gethnamaddr.c
--- bind-4.9.4-orig/res/gethnamaddr.c   Mon Aug  5 02:31:35 1996
+++ bind-4.9.4/res/gethnamaddr.c        Tue Aug 27 15:33:11 1996
@@ -133,6 +133,7 @@
 } align;

 extern int h_errno;
+int _dns_ttl_;

 #ifdef DEBUG
 static void
@@ -223,6 +224,7 @@
        host.h_addr_list = h_addr_ptrs;
        haveanswer = 0;
        had_error = 0;
+       _dns_ttl_ = -1;
        while (ancount-- > 0 && cp < eom && !had_error) {
                n = dn_expand(answer->buf, eom, cp, bp, buflen);
                if ((n < 0) || !(*name_ok)(bp)) {
@@ -232,8 +234,11 @@
                cp += n;                        /* name */
                type = _getshort(cp);
                cp += INT16SZ;                  /* type */
-               class = _getshort(cp);
-               cp += INT16SZ + INT32SZ;        /* class, TTL */
+               class = _getshort(cp);
+               cp += INT16SZ;                  /* class */
+               if (qtype == T_A  && type == T_A)
+                       _dns_ttl_ = _getlong(cp);
+               cp += INT32SZ;                  /* TTL */
                n = _getshort(cp);
                cp += INT16SZ;                  /* len */
                if (class != C_IN) {
```

And here is a patch for BIND-8:

```
*** src/lib/irs/dns_ho.c.orig   Tue May 26 21:55:51 1998
--- src/lib/irs/dns_ho.c        Tue May 26 21:59:57 1998
***************
*** 87,92 ****
--- 87,93 ----
  #endif

  extern int h_errno;
+ int _dns_ttl_;

  /* Definitions. */

***************
```

```
*** 395,400 ****
--- 396,402 ----
        pvt->host.h_addr_list = pvt->h_addr_ptrs;
        haveanswer = 0;
        had_error = 0;
+       _dns_ttl_ = -1;
        while (ancount-- > 0 && cp < eom && !had_error) {
                n = dn_expand(ansbuf, eom, cp, bp, buflen);
                if ((n < 0) || !(*name_ok)(bp)) {
***************
*** 404,411 ****
                cp += n;                        /* name */
                type = ns_get16(cp);
                cp += INT16SZ;                  /* type */
!               class = ns_get16(cp);
!               cp += INT16SZ + INT32SZ;        /* class, TTL */
                n = ns_get16(cp);
                cp += INT16SZ;                  /* len */
                if (class != C_IN) {
--- 406,416 ----
                cp += n;                        /* name */
                type = ns_get16(cp);
                cp += INT16SZ;                  /* type */
!               class = _getshort(cp);
!               cp += INT16SZ;                  /* class */
!               if (qtype == T_A  && type == T_A)
!                       _dns_ttl_ = _getlong(cp);
!               cp += INT32SZ;                  /* TTL */
                n = ns_get16(cp);
                cp += INT16SZ;                  /* len */
                if (class != C_IN) {
```

2.10 My platform is BSD/OS or BSDI and I can't compile Squid

```
cache_cf.c: In function `parseConfigFile':
cache_cf.c:1353: yacc stack overflow before `token'
...
```

You may need to upgrade your gcc installation to a more recent version. Check your gcc version with

```
gcc -v
```

If it is earlier than 2.7.2, you might consider upgrading.

2.11 Problems compiling *libmiscutil.a* on Solaris

The following error occurs on Solaris systems using gcc when the Solaris C compiler is not installed:

```
/usr/bin/rm -f libmiscutil.a
/usr/bin/false r libmiscutil.a rfc1123.o rfc1738.o util.o ...
```

```
make[1]: *** [libmiscutil.a] Error 255
make[1]: Leaving directory `/tmp/squid-1.1.11/lib'
make: *** [all] Error 1
```

Note on the second line the **/usr/bin/false**. This is supposed to be a path to the *ar* program. If *congure* cannot nd *ar* on your system, then it substitues *false*.

To x this you either need to:

Add */usr/ccs/bin* to your PATH. This is where the *ar* command should be. You need to install SUNWbtool if *ar* is not there. Otherwise,

Install the **binutils** package from *the GNU FTP site* <ftp://ftp.gnu.org/gnu/binutils>. This package includes programs such as *ar*, *as*, and *ld*.

2.12 I have problems compiling Squid on Platform Foo.

Please check the *page of platforms* </platforms.html> on which Squid is known to compile. Your problem might be listed there together with a solution. If it isn't listed there, mail us what you are trying, your Squid version, and the problems you encounter.

2.13 I see a lot warnings while compiling Squid.

Warnings are usually not a big concern, and can be common with software designed to operate on multiple platforms. If you feel like xing compile-time warnings, please do so and send us the patches.

2.14 Building Squid on OS/2

by *Doug Nazar* <mailto:nazard@man-assoc.on.ca>

In order in compile squid, you need to have a reasonable facsimile of a Unix system installed. This includes *bash*, *make*, *sed*, *emx*, various le utilities and a few more. I've setup a TVFS drive that matches a Unix le system but this probably isn't strictly necessary.

I made a few modications to the pristine EMX 0.9d install.

1. added denes for *strcasecmp()* & *strncasecmp()* to *string.h*
2. changed all occurrences of time_t to signed long instead of unsigned long
3. hacked ld.exe
   (a) to search for both xxxx.a and libxxxx.a
   (b) to produce the correct lename when using the -Zexe option

You will need to run *scripts/convert.congure.to.os2* (in the Squid source distribution) to modify the congure script so that it can search for the various programs.

Next, you need to set a few environment variables (see EMX docs for meaning):

```
export EMXOPT="-h256 -c"
export LDFLAGS="-Zexe -Zbin -s"
```

Now you are ready to congure squid:

```
./configure
```

Compile everything:

```
make
```

and nally, install:

```
make install
```

This will by default, install into */usr/local/squid*. If you wish to install somewhere else, see the *–prex* option for congure.

Now, don't forget to set EMXOPT before running squid each time. I recommend using the -Y and -N options.

3 Installing and Running Squid

3.1 How big of a system do I need to run Squid?

There are no hard-and-fast rules. The most important resource for Squid is physical memory. Your processor does not need to be ultra-fast. Your disk system will be the major bottleneck, so fast disks are important for high-volume caches. Do not use IDE disks if you can help it.

In late 1998, if you are buying a new machine for a cache, I would recommend the following conguration:

300 MHz Pentium II CPU

512 MB RAM

Five 9 GB UW-SCSI disks

Your system disk, and logle disk can probably be IDE without losing any cache performance.

Also, see *Squid Sizing for Intel Platforms* <http://wwwcache.ja.net/servers/squids.html> by Martin Hamilton This is a very nice page summarizing system congurations people are using for large Squid caches.

3.2 How do I install Squid?

After 2, you can install it with this simple command:

```
% make install
```

If you have enabled the 7.6 then you will also want to type

```
% su
# make install-pinger
```

After installing, you will want to edit and customize the *squid.conf* le. By default, this le is located at */usr/local/squid/etc/squid.conf*.

Also, a QUICKSTART guide has been included with the source distribution. Please see the directory where you unpacked the source archive.

3.3 What does the *squid.conf* le do?

The *squid.conf* le denes the conguration for *squid*. the conguration includes (but not limited to) HTTP port number, the ICP request port number, incoming and outgoing requests, information about rewall access, and various timeout information.

3.4 Do you have a *squid.conf* example?

Yes, after you `make install`, a sample *squid.conf* le will exist in the "etc" directory under the Squid installation directory.

The sample *squid.conf* le contains comments explaining each option.

3.5 How do I start Squid?

First you need to make your Squid conguration. The Squid conguration can be found in /usr/local/squid/etc/squid.conf and by default includes documentation on all directives.

In the Suqid distribution there is a small QUICKSTART guide indicating which directives you need to look closer at and why. At a absolute minimum you need to change the http_access conguration to allow access from your clients.

To verify your conguration le you can use the -k parse option

```
% /usr/local/squid/sbin/squid -k parse
```

If this outputs any errors then these are syntax errors or other fatal miscongurations and needs to be corrected before you continue. If it is silent and immediately gives back the command promt then your squid.conf is syntactically correct and could be understood by Squid.

After you've nished editing the conguration le, you can start Squid for the rst time. The procedure depends a little bit on which version you are using.

First, you must create the swap directories. Do this by running Squid with the -z option:

```
% /usr/local/squid/sbin/squid -z
```

NOTE: If you run Squid as root then you may need to rst create /usr/local/squid/var/logs and your cache_dir directories and assign ownership of these to the cache_eective _user congured in your squid.conf.

Once the creation of the cache directories completes, you can start Squid and try it out. Probably the best thing to do is run it from your terminal and watch the debugging output. Use this command:

```
% /usr/local/squid/sbin/squid -NCd1
```

If everything is working okay, you will see the line:

```
Ready to serve requests.
```

If you want to run squid in the background, as a daemon process, just leave o all options:

```
% /usr/local/squid/sbin/squid
```

NOTE: depending on which http_port you select you may need to start squid as root (http_port <1024).

NOTE: In Squid-2.4 and earlier Squid was installed in bin by default, not sbin.

3.6 How do I start Squid automatically when the system boots?

Squid-2 has a restart feature built in. This greatly simplies starting Squid and means that you don't need to use *RunCache* or *inittab*. At the minimum, you only need to enter the pathname to the Squid executable. For example:

```
/usr/local/squid/sbin/squid
```

Squid will automatically background itself and then spawn a child process. In your *syslog* messages le, you should see something like this:

```
Sep 23 23:55:58 kitty squid[14616]: Squid Parent: child process 14617 started
```

That means that process ID 14563 is the parent process which monitors the child process (pid 14617). The child process is the one that does all of the work. The parent process just waits for the child process to exit. If the child process exits unexpectedly, the parent will automatically start another child process. In that case, *syslog* shows:

```
Sep 23 23:56:02 kitty squid[14616]: Squid Parent: child process 14617 exited with status 1
Sep 23 23:56:05 kitty squid[14616]: Squid Parent: child process 14619 started
```

If there is some problem, and Squid can not start, the parent process will give up after a while. Your *syslog* will show:

```
Sep 23 23:56:12 kitty squid[14616]: Exiting due to repeated, frequent failures
```

When this happens you should check your *syslog* messages and *cache.log* le for error messages.

When you look at a process (*ps* command) listing, you'll see two squid processes:

```
24353  ??  Ss     0:00.00 /usr/local/squid/bin/squid
24354  ??  R      0:03.39 (squid) (squid)
```

The rst is the parent process, and the child process is the one called "(squid)". Note that if you accidentally kill the parent process, the child process will not notice.

If you want to run Squid from your termainal and prevent it from backgrounding and spawning a child process, use the *-N* command line option.

```
/usr/local/squid/bin/squid -N
```

3.6.1 From inittab

On systems which have an */etc/inittab* le (Digital Unix, Solaris, IRIX, HP-UX, Linux), you can add a line like this:

```
sq:3:respawn:/usr/local/squid/sbin/squid.sh < /dev/null >> /tmp/squid.log 2>&1
```

We recommend using a *squid.sh* shell script, but you could instead call Squid directly with the -N option and other options you may require. A sameple *squid.sh* script is shown below:

```
#!/bin/sh
C=/usr/local/squid
PATH=/usr/bin:$C/bin
TZ=PST8PDT
export PATH TZ

# User to notify on restarts
notify="root"

# Squid command line options
opts=""

cd $C
umask 022
sleep 10
while [ -f /var/run/nosquid ]; do
        sleep 1
done
/usr/bin/tail -20 $C/logs/cache.log \
        | Mail -s "Squid restart on `hostname` at `date`" $notify
exec bin/squid -N $opts
```

3.6.2 From rc.local

On BSD-ish systems, you will need to start Squid from the "rc" les, usually */etc/rc.local*. For example:

```
if [ -f /usr/local/squid/sbin/squid ]; then
        echo -n ' Squid'
        /usr/local/squid/sbin/squid
fi
```

3.6.3 From init.d

Squid ships with a init.d type startup script in contrib/squid.rc which works on most init.d type systems. Or you can write your own using any normal init.d script found in your system as template and add the start/stop fragments shown below.

Start:

```
/usr/local/squid/sbin/squid
```

Stop:

```
/usr/local/squid/sbin/squid -k shutdown
n=120
while /usr/local/squid/sbin/squid -k check && [ $n -gt 120 ]; do
    sleep 1
    echo -n .
    n=`expr $n - 1`
done
```

3.7 How do I tell if Squid is running?

You can use the *client* program:

```
% client http://www.netscape.com/ > test
```

There are other command-line HTTP client programs available as well. Two that you may nd useful are *wget* <ftp://gnjilux.cc.fer.hr/pub/unix/util/wget/> and *echoping* <ftp://ftp.internatif.org/pub/unix/echoping/>.

Another way is to use Squid itself to see if it can signal a running Squid process:

```
% squid -k check
```

And then check the shell's exit status variable.

Also, check the log les, most importantly the *access.log* and *cache.log* les.

3.8 *squid* command line options

These are the command line options for **Squid-2**:

-a

Specify an alternate port number for incoming HTTP requests. Useful for testing a conguration le on a non-standard port.

-d

Debugging level for "stderr" messages. If you use this option, then debugging messages up to the specied level will also be written to stderr.

-f

Specify an alternate *squid.conf* le instead of the pathname compiled into the executable.

-h

Prints the usage and help message.

-k recongure

Sends a *HUP* signal, which causes Squid to re-read its conguration les.

-k rotate

Sends an *USR1* signal, which causes Squid to rotate its log les. Note, if *logle_rotate* is set to zero, Squid still closes and re-opens all log les.

-k shutdown

Sends a *TERM* signal, which causes Squid to wait briey for current connections to nish and then exit. The amount of time to wait is specied with *shutdown_lifetime*.

-k interrupt

Sends an *INT* signal, which causes Squid to shutdown immediately, without waiting for current connections.

-k kill

Sends a *KILL* signal, which causes the Squid process to exit immediately, without closing any connections or log les. Use this only as a last resort.

-k debug

Sends an *USR2* signal, which causes Squid to generate full debugging messages until the next *USR2* signal is recieved. Obviously very useful for debugging problems.

-k check

Sends a "*ZERO*" signal to the Squid process. This simply checks whether or not the process is actually running.

-s

Send debugging (level 0 only) message to syslog.

-u

Specify an alternate port number for ICP messages. Useful for testing a conguration le on a non-standard port.

-v

Prints the Squid version.

-z

Creates disk swap directories. You must use this option when installing Squid for the rst time, or when you add or modify the *cache_dir* conguration.

-D

Do not make initial DNS tests. Normally, Squid looks up some well-known DNS hostnames to ensure that your DNS name resolution service is working properly.

-F

If the *swap.state* logs are clean, then the cache is rebuilt in the "foreground" before any requests are served. This will decrease the time required to rebuild the cache, but HTTP requests will not be satisied during this time.

-N

Do not automatically become a background daemon process.

-R

Do not set the SO_REUSEADDR option on sockets.

-V

Enable virtual host support for the httpd-accelerator mode. This is identical to writing *httpd_accel_host virtual* in the cong le.

-X

Enable full debugging while parsing the cong le.

-Y

Return ICP_OP_MISS_NOFETCH instead of ICP_OP_MISS while the *swap.state* le is being read. If your cache has mostly child caches which use ICP, this will allow your cache to rebuild faster.

3.9 How do I see how Squid works?

Check the *cache.log* le in your logs directory. It logs interesting (and boring) things as a part of its normal operation.

Install and use the 9.

4 Conguration issues

4.1 How do I join a cache hierarchy?

To place your cache in a hierarchy, use the `cache_host` directive in *squid.conf* to specify the parent and sibling nodes.

For example, the following *squid.conf* le on `childcache.example.com` congures its cache to retrieve data from one parent cache and two sibling caches:

```
#  squid.conf - On the host: childcache.example.com
#
#  Format is: hostname  type  http_port  udp_port
#
cache_host parentcache.example.com   parent  3128 3130
cache_host childcache2.example.com   sibling 3128 3130
cache_host childcache3.example.com   sibling 3128 3130
```

The `cache_host_domain` directive allows you to specify that certain caches siblings or parents for certain domains:

```
#  squid.conf - On the host: sv.cache.nlanr.net
#
#  Format is: hostname  type  http_port  udp_port
#

cache_host electraglide.geog.unsw.edu.au parent 3128 3130
cache_host cache1.nzgate.net.nz          parent 3128 3130
cache_host pb.cache.nlanr.net   parent 3128 3130
cache_host it.cache.nlanr.net   parent 3128 3130
cache_host sd.cache.nlanr.net   parent 3128 3130
cache_host uc.cache.nlanr.net   sibling 3128 3130
cache_host bo.cache.nlanr.net   sibling 3128 3130
cache_host_domain electraglide.geog.unsw.edu.au .au
cache_host_domain cache1.nzgate.net.nz   .au .aq .fj .nz
cache_host_domain pb.cache.nlanr.net     .uk .de .fr .no .se .it
cache_host_domain it.cache.nlanr.net     .uk .de .fr .no .se .it
cache_host_domain sd.cache.nlanr.net     .mx .za .mu .zm
```

The conguration above indicates that the cache will use `pb.cache.nlanr.net` and `it.cache.nlanr.net` for domains uk, de, fr, no, se and it, `sd.cache.nlanr.net` for domains mx, za, mu and zm, and `cache1.nzgate.net.nz` for domains au, aq, fj, and nz.

4.2 How do I join NLANR's cache hierarchy?

We have a simple set of *guidelines for joining* `<http://www.ircache.net/Cache/joining.html>` the NLANR cache hierarchy.

4.3 Why should I want to join NLANR's cache hierarchy?

The NLANR hierarchy can provide you with an initial source for parent or sibling caches. Joining the NLANR global cache system will frequently improve the performance of your caching service.

4.4 How do I register my cache with NLANR's registration service?

Just enable these options in your *squid.conf* and you'll be registered:

```
cache_announce 24
announce_to sd.cache.nlanr.net:3131
```

NOTE: announcing your cache **is not** the same thing as joining the NLANR cache hierarchy. You can join the NLANR cache hierarchy without registering, and you can register without joining the NLANR cache hierarchy.

4.5 How do I nd other caches close to me and arrange parent/child/sibling relationships with them?

Visit the NLANR cache *registration database* `<http://www.ircache.net/Cache/Tracker/>` to discover other caches near you. Keep in mind that just because a cache is registered in the database **does not** mean they are willing to be your parent/sibling/child. But it can't hurt to ask...

4.6 My cache registration is not appearing in the Tracker database.

Your site will not be listed if your cache IP address does not have a DNS PTR record. If we can't map the IP address back to a domain name, it will be listed as "Unknown."

The registration messages are sent with UDP. We may not be receiving your announcement message due to rewalls which block UDP, or dropped packets due to congestion.

4.7 What is the httpd-accelerator mode?

This entry has been moved to 20.1.

4.8 How do I congure Squid to work behind a rewall?

Note: The information here is current for version 2.2.

If you are behind a rewall then you can't make direct connections to the outside world, so you **must** use a parent cache. Squid doesn't use ICP queries for a request if it's behind a rewall or if there is only one parent.

You can use the `never_direct` access list in *squid.conf* to specify which requests must be forwarded to your parent cache outside the rewall, and the `always_direct` access list to specify which requests must not be forwarded. For example, if Squid must connect directly to all servers that end with *mydomain.com*, but must use the parent for all others, you would write:

```
acl INSIDE dstdomain .mydomain.com
always_direct allow INSIDE
never_direct allow all
```

You could also specify internal servers by IP address

```
acl INSIDE_IP dst 1.2.3.0/24
always_direct allow INSIDE_IP
never_direct allow all
```

Note, however that when you use IP addresses, Squid must perform a DNS lookup to convert URL hostnames to an address. Your internal DNS servers may not be able to lookup external domains.

If you use *never_direct* and you have multiple parent caches, then you probably will want to mark one of them as a default choice in case Squid can't decide which one to use. That is done with the *default* keyword on a *cache_peer* line. For example:

```
cache_peer xyz.mydomain.com parent 3128 0 default
```

4.9 How do I congure Squid forward all requests to another proxy?

Note: The information here is current for version 2.2.

First, you need to give Squid a parent cache. Second, you need to tell Squid it can not connect directly to origin servers. This is done with three conguration le lines:

```
cache_peer parentcache.foo.com parent 3128 0 no-query default
acl all src 0.0.0.0/0.0.0.0
never_direct allow all
```

Note, with this conguration, if the parent cache fails or becomes unreachable, then every request will result in an error message.

In case you want to be able to use direct connections when all the parents go down you should use a dierent approach:

```
cache_peer parentcache.foo.com parent 3128 0 no-query
prefer_direct off
```

The default behaviour of Squid in the absence of positive ICP, HTCP, etc replies is to connect to the origin server instead of using parents. The *prefer_direct o* directive tells Squid to try parents rst.

4.10 I have *dnsserver* processes that aren't being used, should I lower the number in *squid.conf*?

The *dnsserver* processes are used by *squid* because the `gethostbyname(3)` library routines used to convert web sites names to their internet addresses blocks until the function returns (i.e., the process that calls it has to wait for a reply). Since there is only one *squid* process, everyone who uses the cache would have to wait each time the routine was called. This is why the *dnsserver* is a separate process, so that these processes can block, without causing blocking in *squid*.

It's very important that there are enough *dnsserver* processes to cope with every access you will need, otherwise *squid* will stop occasionally. A good rule of thumb is to make sure you have at least the maximum number of dnsservers *squid* has **ever** needed on your system, and probably add two to be on the safe side. In other words, if you have only ever seen at most three *dnsserver* processes in use, make at least ve. Remember that a *dnsserver* is small and, if unused, will be swapped out.

4.11 My *dnsserver* average/median service time seems high, how can I reduce it?

First, nd out if you have enough *dnsserver* processes running by looking at the Cachemanager *dns* output. Ideally, you should see that the rst *dnsserver* handles a lot of requests, the second one less than the rst, etc. The last *dnsserver* should have serviced relatively few requests. If there is not an obvious decreasing trend, then you need to increase the number of *dns_children* in the conguration le. If the last *dnsserver* has zero requests, then you denately have enough.

Another factor which aects the *dnsserver* service time is the proximity of your DNS resolver. Normally we do not recommend running Squid and *named* on the same host. Instead you should try use a DNS resolver (*named*) on a dierent host, but on the same LAN. If your DNS trac must pass through one or more routers, this could be causing unnecessary delays.

4.12 How can I easily change the default HTTP port?

Before you run the congure script, simply set the *CACHE_HTTP_PORT* environment variable.

```
setenv CACHE_HTTP_PORT 8080
./configure
make
make install
```

4.13 Is it possible to control how big each *cache_dir* is?

With Squid-1.1 it is NOT possible. Each *cache_dir* is assumed to be the same size. The *cache_swap* setting denes the size of all *cache_dir*'s taken together. If you have N *cache_dir*'s then each one will hold *cache_swap* N Megabytes.

4.14 What *cache_dir* size should I use?

Most people have a disk partition dedicated to the Squid cache. You don't want to use the entire partition size. You have to leave some extra room. Currently, Squid is not very tolerant of running out of disk space.

Lets say you have a 9GB disk. Remember that disk manufacturers lie about the space available. A so-called 9GB disk usually results in about 8.5GB of raw, usable space. First, put a lesystem on it, and mount it. Then check the "available space" with your *df* program. Note that you lose some disk space to lesystem overheads, like superblocks, inodes, and directory entries. Also note that Unix normally keeps 10% free for itself. So with a 9GB disk, you're probably down to about 8GB after formatting.

Next, I suggest taking o another 10% or so for Squid overheads, and a "safe buer." Squid normally puts its *swap.state* les in each cache directory. These grow in size until you rotate the logs, or restart squid. Also note that Squid performs better when there is more free space. So if performance is important to you, then take o even more space. Typically, for a 9GB disk, I recommend a *cache_dir* setting of 6000 to 7500 Megabytes:

```
cache_dir ... 7000 16 256
```

Its better to start out conservative. After the cache becomes full, look at the disk usage. If you think there is plenty of unused space, then increase the *cache_dir* setting a little.

If you're getting "disk full" write errors, then you denately need to decrease your cache size.

4.15 I'm adding a new *cache_dir*. Will I lose my cache?

With Squid-1.1, yes, you will lose your cache. This is because version 1.1 uses a simplistic algorithm to distribute les between cache directories.

With Squid-2, you will not lose your existing cache. You can add and delete *cache_dir*'s without aecting any of the others.

4.16 Squid and *http-gw* from the TIS toolkit.

Several people on both the *fwtk-users* and the *squid-users* mailing asked about using Squid in combination with http-gw from the *TIS toolkit* <http://www.tis.com/>. The most elegant way in my opinion is to run an internal Squid caching proxyserver which handles client requests and let this server forward it's requests to the http-gw running on the rewall. Cache hits won't need to be handled by the rewall.

In this example Squid runs on the same server as the http-gw, Squid uses 8000 and http-gw uses 8080 (web). The local domain is *home.nl*.

4.16.1 Firewall conguration:

Either run http-gw as a daemon from the */etc/rc.d/rc.local* (Linux Slackware):

```
exec /usr/local/fwtk/http-gw -daemon 8080
```

or run it from inetd like this:

```
web stream      tcp      nowait.100  root /usr/local/fwtk/http-gw http-gw
```

I increased the watermark to 100 because a lot of people run into problems with the default value.

Make sure you have at least the following line in */usr/local/etc/netperm-table*:

```
http-gw:        hosts 127.0.0.1
```

You could add the IP-address of your own workstation to this rule and make sure the http-gw by itself works, like:

```
http-gw:                hosts 127.0.0.1 10.0.0.1
```

4.16.2 Squid conguration:

The following settings are important:

```
http_port       8000
icp_port        0

cache_host      localhost.home.nl parent 8080 0 default
acl HOME        dstdomain .home.nl
never_direct    deny HOME
```

This tells Squid to use the parent for all domains other than *home.nl*. Below, *access.log* entries show what happens if you do a reload on the Squid-homepage:

```
872739961.631   1566 10.0.0.21 ERR_CLIENT_ABORT/304 83 GET http://www.squid-cache.org/ - DEFAULT_PARENT
872739962.976   1266 10.0.0.21 TCP_CLIENT_REFRESH/304 88 GET http://www.nlanr.net/Images/cache_now.gif
872739963.007   1299 10.0.0.21 ERR_CLIENT_ABORT/304 83 GET http://www.squid-cache.org/Icons/squidnow.gi
872739963.061   1354 10.0.0.21 TCP_CLIENT_REFRESH/304 83 GET http://www.squid-cache.org/Icons/Squidlogo
```

http-gw entries in syslog:

```
Aug 28 02:46:00 memo http-gw[2052]: permit host=localhost/127.0.0.1 use of gateway (V2.0beta)
Aug 28 02:46:00 memo http-gw[2052]: log host=localhost/127.0.0.1 protocol=HTTP cmd=dir dest=www.squid-ca
Aug 28 02:46:01 memo http-gw[2052]: exit host=localhost/127.0.0.1 cmds=1 in=0 out=0 user=unauth duratio
Aug 28 02:46:01 memo http-gw[2053]: permit host=localhost/127.0.0.1 use of gateway (V2.0beta)
Aug 28 02:46:01 memo http-gw[2053]: log host=localhost/127.0.0.1 protocol=HTTP cmd=get dest=www.squid-ca
Aug 28 02:46:01 memo http-gw[2054]: permit host=localhost/127.0.0.1 use of gateway (V2.0beta)
Aug 28 02:46:01 memo http-gw[2054]: log host=localhost/127.0.0.1 protocol=HTTP cmd=get dest=www.squid-ca
Aug 28 02:46:01 memo http-gw[2055]: permit host=localhost/127.0.0.1 use of gateway (V2.0beta)
Aug 28 02:46:01 memo http-gw[2055]: log host=localhost/127.0.0.1 protocol=HTTP cmd=get dest=www.nlanr.ne
Aug 28 02:46:02 memo http-gw[2055]: exit host=localhost/127.0.0.1 cmds=1 in=0 out=0 user=unauth duratio
Aug 28 02:46:03 memo http-gw[2053]: exit host=localhost/127.0.0.1 cmds=1 in=0 out=0 user=unauth duratio
Aug 28 02:46:04 memo http-gw[2054]: exit host=localhost/127.0.0.1 cmds=1 in=0 out=0 user=unauth duratio
```

To summarize:

Advantages:

> http-gw allows you to selectively block ActiveX and Java, and it's primary design goal is security.
>
> The rewall doesn't need to run large applications like Squid.
>
> The internal Squid-server still gives you the benet of caching.

Disadvantages:

> The internal Squid proxyserver can't (and shouldn't) work with other parent or neighbor caches.
>
> Initial requests are slower because these go through http-gw, http-gw also does reverse lookups. Run a nameserver on the rewall or use an internal nameserver.

–Rodney van den Oever <mailto:RvdOever@baan.nl>

4.17 What is "HTTP_X_FORWARDED_FOR"? Why does squid provide it to WWW servers, and how can I stop it?

When a proxy-cache is used, a server does not see the connection coming from the originating client. Many people like to implement access controls based on the client address. To accommodate these people, Squid adds its own request header called "X-Forwarded-For" which looks like this:

```
X-Forwarded-For: 128.138.243.150, unknown, 192.52.106.30
```

Entries are always IP addresses, or the word *unknown* if the address could not be determined or if it has been disabled with the *forwarded_for* conguration option.

We must note that access controls based on this header are extremely weak and simple to fake. Anyone may hand-enter a request with any IP address whatsoever. This is perhaps the reason why client IP addresses have been omitted from the HTTP/1.1 specication.

Because of the weakness of this header support for access controls based on X-Forwarder-For is not yet available in any ocially released version of squid. However, unocial patches are available from the *follow_x* <http://devel.squid-cache.org/follow_xff/index.html> Squid development project and may be integrated into later versions of Squid once a suitable trust model have been developed.

4.18 Can Squid anonymize HTTP requests?

Yes it can, however the way of doing it has changed from earlier versions of squid. As of squid-2.2 a more customisable method has been introduced. Please follow the instructions for the version of squid that you are using. As a default, no anonymizing is done.

If you choose to use the anonymizer you might wish to investigate the forwarded_for option to prevent the client address being disclosed. Failure to turn o the forwarded _for option will reduce the eectiveness of the anonymizer. Finally if you lter the User-Agent header using the fake _user_agent option can prevent some user problems as some sites require the User-Agent header.

4.18.1 Squid 2.2

With the introduction of squid 2.2 the anonoymizer has become more customisable. It now allows specication of exactly which headers will be allowed to pass.

The new anonymizer uses the 'anonymize_headers' tag. It has two modes 'deny' all and allow the specied headers. The following example will simulate the old paranoid mode.

```
anonymize_headers allow Allow Authorization Cache-Control
anonymize_headers allow Content-Encoding Content-Length
anonymize_headers allow Content-Type Date Expires Host
anonymize_headers allow If-Modified-Since Last-Modified
anonymize_headers allow Location Pragma Accept Charset
anonymize_headers allow Accept-Encoding Accept-Language
anonymize_headers allow Content-Language Mime-Version
anonymize_headers allow Retry-After Title Connection
anonymize_headers allow Proxy-Connection
```

This will prevent any headers other than those listed from being passed by the proxy.

The second mode is 'allow' all and deny the specied headers. The example replicates the old standard mode.

```
anonymize_headers deny From Referer Server
anonymize_headers deny User-Agent WWW-Authenticate Link
```

It allows all headers to pass unless they are listed.

You can not mix allow and deny in a squid conguration it is either one or the other!

4.18.2 Squid 2.1 and Earlier

There are three modes: *none*, *standard*, and *paranoid*. The mode is set with the *http_anonymizer* conguration option.

With no anonymizing (the default), Squid forwards all request headers as received from the client, to the origin server (subject to the regular rules of HTTP).

In the *standard* mode, Squid lters out the following specic request headers:

From:

Referer:

Server:

User-Agent:

WWW-Authenticate:

Link:

In the *paranoid* mode, Squid allows only the following specic request headers:

Allow:

Authorization:

Cache-Control:

Content-Encoding:

Content-Length:

Content-Type:

Date:

Expires:

Host:

If-Modied-Since:

Last-Modied:

Location:

Pragma:

Accept:

Accept-Charset:

Accept-Encoding:

Accept-Language:

Content-Language:

Mime-Version:

Retry-After:

Title:

Connection:

Proxy-Connection:

References: *Anonymous WWW* `<http://www.iks-jena.de/mitarb/lutz/anon/web.en.html>`

4.19 Can I make Squid go direct for some sites?

Sure, just use the *always_direct* access list.

For example, if you want Squid to connect directly to *hotmail.com* servers, you can use these lines in your cong le:

```
acl hotmail dstdomain .hotmail.com
always_direct allow hotmail
```

4.20 Can I make Squid proxy only, without caching anything?

Sure, there are few things you can do.

You can use the *no_cache* access list to make Squid never cache any response:

```
acl all src 0/0
no_cache deny all
```

With Squid-2.4 and later you can use the "null" storage module:

```
cache_dir null /tmp
```

Note: the directory (e.g., */tmp*) must exist so that squid can chdir to it, unless you also use the *coredump_dir* option.

To congure Squid for the "null" storage module, specify it on the *congure* command line:

```
./configure --enable-storeio=ufs,null ...
```

4.21 Can I prevent users from downloading large les?

You can set the global *reply_body_max_size* parameter. This option controls the largest HTTP message body that will be sent to a cache client for one request.

If the HTTP response coming from the server has a `Content-length` header, then Squid compares the content-length value to the *reply_body_max_size* value. If the content-length is larger, the server connection is closed and the user receives an error message from Squid.

Some responses don't have `Content-length` headers. In this case, Squid counts how many bytes are written to the client. Once the limit is reached, the client's connection is simply closed.

Note that "creative" user-agents will still be able to download really large les through the cache using HTTP/1.1 range requests.

5 Communication between browsers and Squid

Most web browsers available today support proxying and are easily congured to use a Squid server as a proxy. Some browsers support advanced features such as lists of domains or URL patterns that shouldn't be fetched through the proxy, or JavaScript automatic proxy conguration.

5.1 Netscape manual conguration

Select **Network Preferences** from the **Options** menu. On the **Proxies** page, click the radio button next to **Manual Proxy Conguration** and then click on the **View** button. For each protocol that your Squid server supports (by default, HTTP, FTP, and gopher) enter the Squid server's hostname or IP address and put the HTTP port number for the Squid server (by default, 3128) in the **Port** column. For any protocols that your Squid does not support, leave the elds blank.

Here is a *screen shot* `</Doc/FAQ/navigator.jpg>` of the Netscape Navigator manual proxy conguration screen.

5.2 Netscape automatic conguration

Netscape Navigator's proxy conguration can be automated with JavaScript (for Navigator versions 2.0 or higher). Select **Network Preferences** from the **Options** menu. On the **Proxies** page, click the radio button next to **Automatic Proxy Conguration** and then ll in the URL for your JavaScript proxy conguration le in the text box. The box is too small, but the text will scroll to the right as you go.

Here is a *screen shot* `</Doc/FAQ/navigator-auto.jpg>` of the Netscape Navigator automatic proxy conguration screen.

You may also wish to consult Netscape's documentation for the Navigator *JavaScript proxy conguration* `<http://home.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html>`

Here is a sample auto conguration JavaScript from Oskar Pearson:

```
//We (www.is.co.za) run a central cache for our customers that they
//access through a firewall - thus if they want to connect to their intranet
//system (or anything in their domain at all) they have to connect
//directly - hence all the "fiddling" to see if they are trying to connect
//to their local domain.

//Replace each occurrence of company.com with your domain name
//and if you have some kind of intranet system, make sure
//that you put it's name in place of "internal" below.

//We also assume that your cache is called "cache.company.com", and
//that it runs on port 8080. Change it down at the bottom.

//(C) Oskar Pearson and the Internet Solution (http://www.is.co.za)

    function FindProxyForURL(url, host)
        {
            //If they have only specified a hostname, go directly.
            if (isPlainHostName(host))
                    return "DIRECT";

            //These connect directly if the machine they are trying to
            //connect to starts with "intranet" - ie http://intranet
            //Connect  directly if it is intranet.*
            //If you have another machine that you want them to
            //access directly, replace "internal*" with that
            //machine's name
```

```
    if (shExpMatch( host, "intranet*")||
                    shExpMatch(host, "internal*"))
        return "DIRECT";

    //Connect directly to our domains (NB for Important News)
    if (dnsDomainIs( host,"company.com")||
    //If you have another domain that you wish to connect to
    //directly, put it in here
                    dnsDomainIs(host,"sistercompany.com"))
        return "DIRECT";

    //So the error message "no such host" will appear through the
    //normal Netscape box - less support queries :)
    if (!isResolvable(host))
            return "DIRECT";

    //We only cache http, ftp and gopher
    if (url.substring(0, 5) == "http:" ||
                    url.substring(0, 4) == "ftp:"||
                    url.substring(0, 7) == "gopher:")

    //Change the ":8080" to the port that your cache
    //runs on, and "cache.company.com" to the machine that
    //you run the cache on
            return "PROXY cache.company.com:8080; DIRECT";

    //We don't cache WAIS
    if (url.substring(0, 5) == "wais:")
            return "DIRECT";

    else
            return "DIRECT";
}
```

5.3 Lynx and Mosaic conguration

For Mosaic and Lynx, you can set environment variables before starting the application. For example (assuming csh or tcsh):

```
% setenv http_proxy http://mycache.example.com:3128/
% setenv gopher_proxy http://mycache.example.com:3128/
% setenv ftp_proxy http://mycache.example.com:3128/
```

For Lynx you can also edit the *lynx.cfg* le to congure proxy usage. This has the added benet of causing all Lynx users on a system to access the proxy without making environment variable changes for each user. For example:

```
http_proxy:http://mycache.example.com:3128/
ftp_proxy:http://mycache.example.com:3128/
gopher_proxy:http://mycache.example.com:3128/
```

5.4 Redundant Proxy Auto-Conguration

There's one nasty side-eect to using auto-proxy scripts: if you start the web browser it will try and load the auto-proxy-script.

If your script isn't available either because the web server hosting the script is down or your workstation can't reach the web server (e.g. because you're working o-line with your notebook and just want to read a previously saved HTML-le) you'll get dierent errors depending on the browser you use.

The Netscape browser will just return an error after a timeout (after that it tries to nd the site 'www.proxy.com' if the script you use is called 'proxy.pac').

The Microsoft Internet Explorer on the other hand won't even start, no window displays, only after about 1 minute it'll display a window asking you to go on with/without proxy conguration.

The point is that your workstations always need to locate the proxy-script. I created some extra redundancy by hosting the script on two web servers (actually Apache web servers on the proxy servers themselves) and adding the following records to my primary nameserver:

```
        proxy   CNAME           proxy1
                CNAME           proxy2
```

The clients just refer to 'http://proxy/proxy.pac'. This script looks like this:

```
function FindProxyForURL(url,host)
{
        // Hostname without domainname or host within our own domain?
        // Try them directly:
        // http://www.domain.com actually lives before the firewall, so
        // make an exception:
        if ((isPlainHostName(host)||dnsDomainIs( host,".domain.com")) &&
                !localHostOrDomainIs(host, "www.domain.com"))
                return "DIRECT";

        // First try proxy1 then proxy2. One server mostly caches '.com'
        // to make sure both servers are not
        // caching the same data in the normal situation. The other
        // server caches the other domains normally.
        // If one of 'm is down the client will try the other server.
        else if (shExpMatch(host, "*.com"))
                return "PROXY proxy1.domain.com:8080; PROXY proxy2.domain.com:8081; DIRECT";
        return "PROXY proxy2.domain.com:8081; PROXY proxy1.domain.com:8080; DIRECT";
}
```

I made sure every client domain has the appropriate 'proxy' entry. The clients are automatically congured with two nameservers using DHCP.

–*Rodney van den Oever* <mailto:RvdOever@baan.nl>

5.5 Proxy Auto-Conguration with URL Hashing

The *Sharp Super Proxy Script page* <http://naragw.sharp.co.jp/sps/> contains a lot of good information about hash-based proxy auto-conguration scripts. With these you can distribute the load between a number of caching proxies.

5.6 Microsoft Internet Explorer conguration

Select **Options** from the **View** menu. Click on the **Connection** tab. Tick the **Connect through Proxy Server** option and hit the **Proxy Settings** button. For each protocol that your Squid server supports (by default, HTTP, FTP, and gopher) enter the Squid server's hostname or IP address and put the HTTP port number for the Squid server (by default, 3128) in the **Port** column. For any protocols that your Squid does not support, leave the elds blank.

Here is a *screen shot* `</Doc/FAQ/msie.jpg>` of the Internet Explorer proxy conguration screen.

Microsoft is also starting to support Netscape-style JavaScript automated proxy conguration. As of now, only MSIE version 3.0a for Windows 3.1 and Windows NT 3.51 supports this feature (i.e., as of version 3.01 build 1225 for Windows 95 and NT 4.0, the feature was not included).

If you have a version of MSIE that does have this feature, elect **Options** from the **View** menu. Click on the **Advanced** tab. In the lower left-hand corner, click on the **Automatic Conguration** button. Fill in the URL for your JavaScript le in the dialog box it presents you. Then exit MSIE and restart it for the changes to take eect. MSIE will reload the JavaScript le every time it starts.

5.7 Netmanage Internet Chameleon WebSurfer conguration

Netmanage WebSurfer supports manual proxy conguration and exclusion lists for hosts or domains that should not be fetched via proxy (this information is current as of WebSurfer 5.0). Select **Preferences** from the **Settings** menu. Click on the **Proxies** tab. Select the **Use Proxy** options for HTTP, FTP, and gopher. For each protocol that enter the Squid server's hostname or IP address and put the HTTP port number for the Squid server (by default, 3128) in the **Port** boxes. For any protocols that your Squid does not support, leave the elds blank.

Take a look at this *screen shot* `</Doc/FAQ/netmanage.jpg>` if the instructions confused you.

On the same conguration window, you'll nd a button to bring up the exclusion list dialog box, which will let you enter some hosts or domains that you don't want fetched via proxy. It should be self-explanatory, but you might look at this *screen shot* `</Doc/FAQ/netmanage-exclusion.jpg>` just for fun anyway.

5.8 Opera 2.12 proxy conguration

Select *Proxy Servers...* from the *Preferences* menu. Check each protocol that your Squid server supports (by default, HTTP, FTP, and Gopher) and enter the Squid server's address as hostname:port (e.g. mycache.example.com:3128 or 123.45.67.89:3128). Click on *Okay* to accept the setup.

Notes:

> Opera 2.12 doesn't support gopher on its own, but requires a proxy; therefore Squid's gopher proxying can extend the utility of your Opera immensely.
>
> Unfortunately, Opera 2.12 chokes on some HTTP requests, for example *abuse.net* `<http://spam.abuse.net/spam/>`. At the moment I think it has something to do with cookies. If you have trouble with a site, try disabling the HTTP proxying by unchecking that protocol in the *Preferences|Proxy Servers...* dialogue. Opera will remember the address, so reenabling is easy.
>
> –*Hume Smith* `<mailto:hclsmith@tallships.istar.ca>`

5.9 How do I tell Squid to use a specic username for FTP urls?

Insert your username in the host part of the URL, for example:

```
ftp://joecool@ftp.foo.org/
```

Squid should then prompt you for your account password. Alternatively, you can specify both your username and password in the URL itself:

```
ftp://joecool:secret@ftp.foo.org/
```

However, we certainly do not recommend this, as it could be very easy for someone to see or grab your password.

5.10 Conguring Browsers for WPAD

by *Mark Reynolds* <mailto:mark@rts.com.au>

You may like to start by reading the *Expired Internet-Draft* <http://www.web-cache.com/Writings/Internet-Drafts/draf that describes WPAD.

After reading the 8 steps below, if you don't understand any of the terms or methods mentioned, you probably shouldn't be doing this. Implementing wpad requires you to **fully** understand:

1. web server installations and modications.
2. squid proxy server (or others) installation etc.
3. Domain Name System maintenance etc.

Please don't bombard the squid list with web server or dns questions. See your system administrator, or do some more research on those topics.

This is not a recommendation for any product or version. As far as I know IE5 is the only browser out now implementing wpad. I think wpad is an excellent feature that will return several hours of life per month. Hopefully, all browser clients will implement it as well. But it will take years for all the older browsers to fade away though.

I have only focused on the domain name method, to the exclusion of the DHCP method. I think the dns method might be easier for most people. I don't currently, and may never, fully understand wpad and IE5, but this method worked for me. It **may** work for you.

But if you'd rather just have a go ...

1. Create a standard 5.2. The sample provided there is more than adequate to get you going. No doubt all the other load balancing and backup scripts will be ne also.
2. Store the resultant le in the document root directory of a handy web server as *wpad.dat* (Not *proxy.pac* as you may have previously done.)

   Andrei Ivanov <mailto:ira at racoon.riga.lv> notes that you should be able to use an HTTP redirect if you want to store the wpad.dat le somewhere else. You can probably even redirect *wpad.dat* to *proxy.pac*:

   ```
   Redirect /wpad.dat http://racoon.riga.lv/proxy.pac
   ```

3. If you do nothing more, a url like `http://www.your.domain.name/wpad.dat` should bring up the script text in your browser window.

4. Insert the following entry into your web server *mime.types* le. Maybe in addition to your pac le type, if you've done this before.

```
application/x-ns-proxy-autoconfig       dat
```

   And then restart your web server, for new mime type to work.

5. Assuming Internet Explorer 5, under *Tools*, *Internet Options*, *Connections*, *Settings* **or** *Lan Settings*, set **ONLY** *Use Automatic Conguration Script* to be the URL for where your new *wpad.dat* le can be found. i.e. `http://www.your.domain.name/wpad.dat` Test that that all works as per your script and network. There's no point continuing until this works ...

6. Create/install/implement a DNS record so that `wpad.your.domain.name` resolves to the host above where you have a functioning auto cong script running. You should now be able to use `http://wpad.your.domain.name/wpad.dat` as the Auto Cong Script location in step 5 above.

7. And nally, go back to the setup screen detailed in 5 above, and choose nothing but the *Automatically Detect Settings* option, turning everything else o. Best to restart IE5, as you normally do with any Microsoft product... And it should all work. Did for me anyway.

8. One nal question might be 'Which domain name does the client (IE5) use for the wpad... lookup?' It uses the hostname from the control panel setting. It starts the search by adding the hostname "WPAD" to current fully-qualied domain name. For instance, a client in a.b.Microsoft.com would search for a WPAD server at wpad.a.b.microsoft.com. If it could not locate one, it would remove the bottom-most domain and try again; for instance, it would try wpad.b.microsoft.com next. IE 5 would stop searching when it found a WPAD server or reached the third-level domain, wpad.microsoft.com.

Anybody using these steps to install and test, please feel free to make notes, corrections or additions for improvements, and post back to the squid list...

There are probably many more tricks and tips which hopefully will be detailed here in the future. Things like *wpad.dat* les being served from the proxy server themselves, maybe with a round robin dns setup for the WPAD host.

5.11 Conguring Browsers for WPAD with DHCP

You can also use DHCP to congure browsers for WPAD. This technique allows you to set any URL as the PAC URL. For ISC DHCPD, enter a line like this in your *dhcpd.conf* le:

```
option wpad code 252 = text;
option wpad "http://www.example.com/proxy.pac";
```

Replace the hostname with the name or address of your own server.

Ilja Pavkovic notes that the DHCP mode does not work reliably with every version of Internet Explorer. The DNS name method to nd wpad.dat is more reliable.

5.12 IE 5.0x crops trailing slashes from FTP URL's

by *Reuben Farrelly* `<mailto:reuben at reub dot net>`

There was a bug in the 5.0x releases of Internet Explorer in which IE cropped any trailing slash o an FTP URL. The URL showed up correctly in the browser's "Address:" eld, however squid logs show that the trailing slash was being taken o.

An example of where this impacted squid if you had a setup where squid would go direct for FTP directory listings but forward a request to a parent for FTP le transfers. This was useful if your upstream proxy was an older version of Squid or another vendors software which displayed directory listings with broken icons and you wanted your own local version of squid to generate proper FTP directory listings instead. The workaround for this is to add a double slash to any directory listing in which the slash was important, or else upgrade to IE 5.5. (Or use Netscape)

5.13 IE 6.0 SP1 fails when using basic authentication

When using basic authentication with Internet Explorer 6 SP1, you may encounter issues when you rst launch Internet Explorer. The problem will show itself when you rst authenticate, you will receive a "Page Cannot Be Displayed" error. However, if you click refresh, the page will be correctly displayed.

This only happens immediately after you authenticate.

This is not a Squid error or bug. Microsoft broke the Basic Authentication when they put out IE6 SP1.

There is a knowledgebase article (*KB 331906* <http://support.microsoft.com/default.aspx?id=kb;en-us;331906>) regarding this issue. The x is to call Microsoft, open an incident referencing this KB article and they will send you a "hot x". They do warn that this code is not "regression tested" but so far there have not been any reports of this breaking anything else. The problematic le is wininet.dll.

According to Joao Coutinho, this simple solution also corrects the problem:

Go to Tools/Internet

Go to Options/Advanced

UNSELECT "Show friendly HTTP error messages" under Browsing.

6 Squid Log Files

The logs are a valuable source of information about Squid workloads and performance. The logs record not only access information, but also system conguration errors and resource consumption (eg, memory, disk space). There are several log le maintained by Squid. Some have to be explicitely activated during compile time, others can safely be deactivated during run-time.

There are a few basic points common to all log les. The time stamps logged into the log les are usually UTC seconds unless stated otherwise. The initial time stamp usually contains a millisecond extension.

6.1 *squid.out*

If you run your Squid from the *RunCache* script, a le *squid.out* contains the Squid startup times, and also all fatal errors, e.g. as produced by an *assert()* failure. If you are not using *RunCache*, you will not see such a le.

6.2 *cache.log*

The *cache.log* le contains the debug and error messages that Squid generates. If you start your Squid using the default *RunCache* script, or start it with the *-s* command line option, a copy of certain messages will go into your syslog facilities. It is a matter of personal preferences to use a separate le for the squid log data.

From the area of automatic log le analysis, the *cache.log* le does not have much to oer. You will usually look into this le for automated error reports, when programming Squid, testing new features, or searching for reasons of a perceived misbehaviour, etc.

6.3 *useragent.log*

The user agent log le is only maintained, if

1. you congured the compile time *–enable-useragent-log* option, and
2. you pointed the *useragent_log* conguration option to a le.

From the user agent log le you are able to nd out about distributation of browsers of your clients. Using this option in conjunction with a loaded production squid might not be the best of all ideas.

6.4 *store.log*

The *store.log* le covers the objects currently kept on disk or removed ones. As a kind of transaction log it is ususally used for debugging purposes. A denitive statement, whether an object resides on your disks is only possible after analysing the *complete* log le. The release (deletion) of an object may be logged at a later time than the swap out (save to disk).

The *store.log* le may be of interest to log le analysis which looks into the objects on your disks and the time they spend there, or how many times a hot object was accessed. The latter may be covered by another log le, too. With knowledge of the *cache_dir* conguration option, this log le allows for a URL to lename mapping without recursing your cache disks. However, the Squid developers recommend to treat *store.log* primarily as a debug le, and so should you, unless you know what you are doing.

The print format for a store log entry (one line) consists of eleven space-separated columns, compare with the *storeLog()* function in le *src/store_log.c*:

```
"%9d.%03d %-7s %08X %4d %9d %9d %9d %s %d/%d %s %s\n"
```

time

The timestamp when the line was logged in UTC with a millisecond fraction.

action

The action the object was sumitted to, compare with *src/store_log.c*:

CREATE Seems to be unused.

RELEASE The object was removed from the cache (see also 6.4).

SWAPOUT The object was saved to disk.

SWAPIN The object existed on disk and was read into memory.

le number

The le number for the object storage le. Please note that the path to this le is calculated according to your *cache_dir* conguration.

A le number of *FFFFFFFF* denominates "memory only" objects. Any action code for such a le number refers to an object which existed only in memory, not on disk. For instance, if a *RELEASE* code was logged with le number *FFFFFFFF*, the object existed only in memory, and was released from memory.

status

The HTTP reply status code.

datehdr

The value of the HTTP "Date: " reply header.

lastmod

The value of the HTTP "Last-Modied: " reply header.

expires

The value of the HTTP "Expires: " reply header.

type

The HTTP "Content-Type" major value, or "unknown" if it cannot be determined.

sizes

This column consists of two slash separated elds:

1. The advertised content length from the HTTP "Content-Length: " reply header.
2. The size actually read.

If the advertised (or expected) length is missing, it will be set to zero. If the advertised length is not zero, but not equal to the real length, the object will be realeased from the cache.

method

The request method for the object, e.g. *GET*.

key

The key to the object, usually the URL.

The timestamp format for the columns 6.4 to 6.4 are all expressed in UTC seconds. The actual values are parsed from the HTTP reply headers. An unparsable header is represented by a value of -1, and a missing header is represented by a value of -2.

The column 6.4 usually contains just the URL of the object. Some objects though will never become public. Thus the key is said to include a unique integer number and the request method in addition to the URL.

6.5 *hierarchy.log*

This logle exists for Squid-1.0 only. The format is

```
[date] URL peerstatus peerhost
```

6.6 *access.log*

Most log le analysis program are based on the entries in *access.log*. Currently, there are two le formats possible for the log le, depending on your conguration for the *emulate_httpd_log* option. By default, Squid will log in its native log le format. If the above option is enabled, Squid will log in the common log le format as dened by the CERN web daemon.

The common log le format contains other information than the native log le, and less. The native format contains more information for the admin interested in cache evaluation.

6.6.1 *The common log le format*

The *Common Logle Format* <http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format> is used by numerous HTTP servers. This format consists of the following seven elds:

```
remotehost rfc931 authuser [date] "method URL" status bytes
```

It is parsable by a variety of tools. The common format contains dierent information than the native log le format. The HTTP version is logged, which is not logged in native log le format.

6.6.2 *The native log le format*

The native format is dierent for dierent major versions of Squid. For Squid-1.0 it is:

```
time elapsed remotehost code/status/peerstatus bytes method URL
```

For Squid-1.1, the information from the *hierarchy.log* was moved into *access.log*. The format is:

```
time elapsed remotehost code/status bytes method URL rfc931 peerstatus/peerhost type
```

For Squid-2 the columns stay the same, though the content within may change a little.

The native log le format logs more and dierent information than the common log le format: the request duration, some timeout information, the next upstream server address, and the content type.

There exist tools, which convert one le format into the other. Please mind that even though the log formats share most information, both formats contain information which is not part of the other format, and thus this part of the information is lost when converting. Especially converting back and forth is not possible without loss.

squid2common.pl is a conversion utility, which converts any of the squid log le formats into the old CERN proxy style output. There exist tools to analyse, evaluate and graph results from that format.

6.6.3 *access.log native format in detail*

It is recommended though to use Squid's native log format due to its greater amount of information made available for later analysis. The print format line for native *access.log* entries looks like this:

```
"%9d.%03d %6d %s %s/%03d %d %s %s %s %s%s/%s %s"
```

Therefore, an *access.log* entry usually consists of (at least) 10 columns separated by one ore more spaces:

time

A Unix timestamp as UTC seconds with a millisecond resolution. You can convert Unix timestamps into something more human readable using this short perl script:

```
#! /usr/bin/perl -p
s/^\d+\.\d+/localtime $&/e;
```

duration

The elapsed time considers how many milliseconds the transaction busied the cache. It diers in interpretation between TCP and UDP:

For HTTP/1.0, this is basically the time between *accept()* and *close()*.

For persistent connections, this ought to be the time between scheduling the reply and nishing sending it.

For ICP, this is the time between scheduling a reply and actually sending it.

Please note that the entries are logged *after* the reply nished being sent, *not* during the lifetime of the transaction.

client address

The IP address of the requesting instance, the client IP address. The *client_netmask* conguration option can distort the clients for data protection reasons, but it makes analysis more dicult. Often it is better to use one of the log le anonymizers.

Also, the *log_fqdn* conguration option may log the fully qualied domain name of the client instead of the dotted quad. The use of that option is discouraged due to its performance impact.

result codes

This column is made up of two entries separated by a slash. This column encodes the transaction result:

1. The cache result of the request contains information on the kind of request, how it was satised, or in what way it failed. Please refer to section 6.7 for valid symbolic result codes.

   Several codes from older versions are no longer available, were renamed, or split. Especially the *ERR_* codes do not seem to appear in the log le any more. Also refer to section 6.7 for details on the codes no longer available in Squid-2.

   The NOVM versions and Squid-2 also rely on the Unix buer cache, thus you will see less *TCP_MEM_HIT*s than with a Squid-1. Basically, the NOVM feature relies on *read()* to obtain an object, but due to the kernel buer cache, no disk activity is needed. Only small objects (below 8KByte) are kept in Squid's part of main memory.

2. The status part contains the HTTP result codes with some Squid specic extensions. Squid uses a subset of the RFC dened error codes for HTTP. Refer to section 6.8 for details of the status codes recognized by a Squid-2.

bytes

The size is the amount of data delivered to the client. Mind that this does not constitute the net object size, as headers are also counted. Also, failed requests may deliver an error page, the size of which is also logged here.

request method

The request method to obtain an object. Please refer to section 6.9 for available methods. If you turned o *log_icp_queries* in your conguration, you will not see (and thus unable to analyse) ICP exchanges. The *PURGE* method is only available, if you have an ACL for "method purge" enabled in your conguration le.

URL

This column contains the URL requested. Please note that the log le may contain whitespaces for the URI. The default conguration for *uri_whitespace* denies whitespaces, though.

rfc931

The eigth column may contain the ident lookups for the requesting client. Since ident lookups have performance impact, the default conguration turns *ident_loookups* o. If turned o, or no ident information is available, a "-" will be logged.

hierarchy code

The hierarchy information consists of three items:

1. Any hierarchy tag may be prexed with *TIMEOUT_*, if the timeout occurs waiting for all ICP replies to return from the neighbours. The timeout is either dynamic, if the *icp_query_timeout* was not set, or the time congured there has run up.
2. A code that explains how the request was handled, e.g. by forwarding it to a peer, or going straight to the source. Refer to section 6.10 for details on hierarchy codes and removed hierarchy codes.
3. The IP address or hostname where the request (if a miss) was forwarded. For requests sent to origin servers, this is the origin server's IP address. For requests sent to a neighbor cache, this is the neighbor's hostname. NOTE: older versions of Squid would put the origin server hostname here.

type

The content type of the object as seen in the HTTP reply header. Please note that ICP exchanges usually don't have any content type, and thus are logged "-". Also, some weird replies have content types ":" or even empty ones.

There may be two more columns in the *access.log*, if the (debug) option *log_mime_headers* is enabled In this case, the HTTP request headers are logged between a "[" and a "]", and the HTTP reply headers are also logged between "[" and "]". All control characters like CR and LF are URL-escaped, but spaces are *not* escaped! Parsers should watch out for this.

6.7 Squid result codes

The **TCP_** codes refer to requests on the HTTP port (usually 3128). The **UDP_** codes refer to requests on the ICP port (usually 3130). If ICP logging was disabled using the *log_icp_queries* option, no ICP replies will be logged.

The following result codes were taken from a Squid-2, compare with the *log_tags* struct in *src/access_log.c*:

TCP_HIT

A valid copy of the requested object was in the cache.

TCP_MISS

The requested object was not in the cache.

TCP_REFRESH_HIT

The requested object was cached but *STALE*. The IMS query for the object resulted in "304 not modied".

TCP_REF_FAIL_HIT

The requested object was cached but *STALE*. The IMS query failed and the stale object was delivered.

TCP_REFRESH_MISS

The requested object was cached but *STALE*. The IMS query returned the new content.

TCP_CLIENT_REFRESH_MISS

The client issued a "no-cache" pragma, or some analogous cache control command along with the request. Thus, the cache has to refetch the object.

TCP_IMS_HIT

The client issued an IMS request for an object which was in the cache and fresh.

TCP_SWAPFAIL_MISS

The object was believed to be in the cache, but could not be accessed.

TCP_NEGATIVE_HIT

Request for a negatively cached object, e.g. "404 not found", for which the cache believes to know that it is inaccessible. Also refer to the explainations for *negative_ttl* in your *squid.conf* le.

TCP_MEM_HIT

A valid copy of the requested object was in the cache *and* it was in memory, thus avoiding disk accesses.

TCP_DENIED

Access was denied for this request.

TCP_OFFLINE_HIT

The requested object was retrieved from the cache during oine mode. The oine mode never validates any object, see *oine _mode* in *squid.conf* le.

UDP_HIT

A valid copy of the requested object was in the cache.

UDP_MISS

The requested object is not in this cache.

UDP_DENIED

Access was denied for this request.

UDP_INVALID

An invalid request was received.

UDP_MISS_NOFETCH

During "-Y" startup, or during frequent failures, a cache in hit only mode will return either UDP_HIT or this code. Neighbours will thus only fetch hits.

NONE

Seen with errors and cachemgr requests.

The following codes are no longer available in Squid-2:

ERR_*

Errors are now contained in the status code.

TCP_CLIENT_REFRESH

See: 6.7.

TCP_SWAPFAIL

See: 6.7.

TCP_IMS_MISS

Deleted, 6.7 used instead.

UDP_HIT_OBJ

Hit objects are no longer available.

UDP_RELOADING

See: 6.7.

6.8 HTTP status codes

These are taken from *RFC 2616* <ftp://ftp.isi.edu/in-notes/rfc2616.txt> and veried for Squid. Squid-2 uses almost all codes except 307 (Temporary Redirect), 416 (Request Range Not Satisable), and 417 (Expectation Failed). Extra codes include 0 for a result code being unavailable, and 600 to signal an invalid header, a proxy error. Also, some denitions were added as for *RFC 2518* <ftp://ftp.isi.edu/in-notes/rfc2518.txt> (WebDAV). Yes, there are really two entries for status code 424, compare with *http_status* in *src/enums.h*:

```
 000 Used mostly with UDP traffic.

 100 Continue
 101 Switching Protocols
*102 Processing

 200 OK
 201 Created
 202 Accepted
 203 Non-Authoritative Information
 204 No Content
 205 Reset Content
 206 Partial Content
*207 Multi Status

 300 Multiple Choices
 301 Moved Permanently
 302 Moved Temporarily
```

```
 303 See Other
 304 Not Modified
 305 Use Proxy
[307 Temporary Redirect]

 400 Bad Request
 401 Unauthorized
 402 Payment Required
 403 Forbidden
 404 Not Found
 405 Method Not Allowed
 406 Not Acceptable
 407 Proxy Authentication Required
 408 Request Timeout
 409 Conflict
 410 Gone
 411 Length Required
 412 Precondition Failed
 413 Request Entity Too Large
 414 Request URI Too Large
 415 Unsupported Media Type
[416 Request Range Not Satisfiable]
[417 Expectation Failed]
*424 Locked
*424 Failed Dependency
*433 Unprocessable Entity

 500 Internal Server Error
 501 Not Implemented
 502 Bad Gateway
 503 Service Unavailable
 504 Gateway Timeout
 505 HTTP Version Not Supported
*507 Insufficient Storage

 600 Squid header parsing error
```

6.9 Request methods

Squid recognizes several request methods as dened in *RFC 2616* <ftp://ftp.isi.edu/in-notes/rfc2616.txt>. Newer versions of Squid (2.2.STABLE5 and above) also recognize *RFC 2518* <ftp://ftp.isi.edu/in-notes/rfc2616.txt> "HTTP Extensions for Distributed Authoring – WEBDAV" extensions.

```
 method    defined    cachabil.  meaning
 --------- ---------- ---------- ----------------------------------------
 GET       HTTP/0.9   possibly   object retrieval and simple searches.
 HEAD      HTTP/1.0   possibly   metadata retrieval.
 POST      HTTP/1.0   CC or Exp. submit data (to a program).
 PUT       HTTP/1.1   never      upload data (e.g. to a file).
```

```
DELETE    HTTP/1.1   never      remove resource (e.g. file).
TRACE     HTTP/1.1   never      appl. layer trace of request route.
OPTIONS   HTTP/1.1   never      request available comm. options.
CONNECT   HTTP/1.1r3 never      tunnel SSL connection.

ICP_QUERY Squid      never      used for ICP based exchanges.
PURGE     Squid      never      remove object from cache.

PROPFIND  rfc2518    ?          retrieve properties of an object.
PROPATCH  rfc2518    ?          change properties of an object.
MKCOL     rfc2518    never      create a new collection.
COPY      rfc2518    never      create a duplicate of src in dst.
MOVE      rfc2518    never      atomically move src to dst.
LOCK      rfc2518    never      lock an object against modifications.
UNLOCK    rfc2518    never      unlock an object.
```

6.10 Hierarchy Codes

The following hierarchy codes are used with Squid-2:

NONE

For TCP HIT, TCP failures, cachemgr requests and all UDP requests, there is no hierarchy information.

DIRECT

The object was fetched from the origin server.

SIBLING_HIT

The object was fetched from a sibling cache which replied with UDP_HIT.

PARENT_HIT

The object was requested from a parent cache which replied with UDP_HIT.

DEFAULT_PARENT

No ICP queries were sent. This parent was chosen because it was marked "default" in the cong le.

SINGLE_PARENT

The object was requested from the only parent appropriate for the given URL.

FIRST_UP_PARENT

The object was fetched from the rst parent in the list of parents.

NO_PARENT_DIRECT

The object was fetched from the origin server, because no parents existed for the given URL.

FIRST_PARENT_MISS

The object was fetched from the parent with the fastest (possibly weighted) round trip time.

CLOSEST_PARENT_MISS

This parent was chosen, because it included the the lowest RTT measurement to the origin server. See also the *closests-only* peer conguration option.

CLOSEST_PARENT

The parent selection was based on our own RTT measurements.

CLOSEST_DIRECT

Our own RTT measurements returned a shorter time than any parent.

NO_DIRECT_FAIL

The object could not be requested because of a rewall conguration, see also *never_direct* and related material, and no parents were available.

SOURCE_FASTEST

The origin site was chosen, because the source ping arrived fastest.

ROUNDROBIN_PARENT

No ICP replies were received from any parent. The parent was chosen, because it was marked for round robin in the cong le and had the lowest usage count.

CACHE_DIGEST_HIT

The peer was chosen, because the cache digest predicted a hit. This option was later replaced in order to distinguish between parents and siblings.

CD_PARENT_HIT

The parent was chosen, because the cache digest predicted a hit.

CD_SIBLING_HIT

The sibling was chosen, because the cache digest predicted a hit.

NO_CACHE_DIGEST_DIRECT

This output seems to be unused?

CARP

The peer was selected by CARP.

ANY_PARENT

part of *src/peer_select.c:hier_strings[]*.

INVALID CODE

part of *src/peer_select.c:hier_strings[]*.

Almost any of these may be preceded by 'TIMEOUT_' if the two-second (default) timeout occurs waiting for all ICP replies to arrive from neighbors, see also the *icp_query_timeout* conguration option.

The following hierarchy codes were removed from Squid-2:

```
code                 meaning
-------------------  ------------------------------------------------
PARENT_UDP_HIT_OBJ   hit objects are not longer available.
SIBLING_UDP_HIT_OBJ  hit objects are not longer available.
SSL_PARENT_MISS      SSL can now be handled by squid.
FIREWALL_IP_DIRECT   No special logging for hosts inside the firewall.
LOCAL_IP_DIRECT      No special logging for local networks.
```

6.11 *cache/log* (Squid-1.x)

This le has a rather unfortunate name. It also is often called the *swap log*. It is a record of every cache object written to disk. It is read when Squid starts up to "reload" the cache. If you remove this le when squid is NOT running, you will eectively wipe out your cache contents. If you remove this le while squid IS running, you can easily recreate it. The safest way is to simply shutdown the running process:

```
% squid -k shutdown
```

This will disrupt service, but at least you will have your swap log back. Alternatively, you can tell squid to rotate its log les. This also causes a clean swap log to be written.

```
% squid -k rotate
```

For Squid-1.1, there are six elds:

1. **leno** : The swap le number holding the object data. This is mapped to a pathname on your lesystem.
2. **timestamp**: This is the time when the object was last veried to be current. The time is a hexadecimal representation of Unix time.
3. **expires**: This is the value of the Expires header in the HTTP reply. If an Expires header was not present, this will be -2 or fe. If the Expires header was present, but invalid (unparsable), this will be -1 or .
4. **lastmod**: Value of the HTTP reply Last-Modied header. If missing it will be -2, if invalid it will be -1.
5. **size**: Size of the object, including headers.
6. **url**: The URL naming this object.

6.12 *swap.state* (Squid-2.x)

In Squid-2, the swap log le is now called *swap.state*. This is a binary le that includes MD5 checksums, and *StoreEntry* elds. Please see the *Programmers Guide* `<../Prog-Guide/>` for information on the contents and format of that le.

If you remove *swap.state* while Squid is running, simply send Squid the signal to rotate its log les:

```
% squid -k rotate
```

Alternatively, you can tell Squid to shutdown and it will rewrite this le before it exits.

If you remove the *swap.state* while Squid is not running, you will not lose your entire cache. In this case, Squid will scan all of the cache directories and read each swap le to rebuild the cache. This can take a very long time, so you'll have to be patient.

By default the *swap.state* le is stored in the top-level of each *cache_dir*. You can move the logs to a dierent location with the *cache_swap_log* option.

6.13 Which log les can I delete safely?

You should never delete *access.log*, *store.log*, *cache.log*, or *swap.state* while Squid is running. With Unix, you can delete a le when a process has the le opened. However, the lesystem space is not reclaimed until the process closes the le.

If you accidentally delete *swap.state* while Squid is running, you can recover it by following the instructions in the previous questions. If you delete the others while Squid is running, you can not recover them.

The correct way to maintain your log les is with Squid's "rotate" feature. You should rotate your log les at least once per day. The current log les are closed and then renamed with numeric extensions (.0, .1, etc). If you want to, you can write your own scripts to archive or remove the old log les. If not, Squid will only keep up to *logle _rotate* versions of each log le. The logle rotation procedure also writes a clean *swap.state* le, but it does not leave numbered versions of the old les.

If you set *logle _rotate* to 0, Squid simply closes and then re-opens the logs. This allows third-party logle management systems, such as *newsyslog*, to maintain the log les.

To rotate Squid's logs, simple use this command:

```
squid -k rotate
```

For example, use this cron entry to rotate the logs at midnight:

```
0 0 * * * /usr/local/squid/bin/squid -k rotate
```

6.14 How can I disable Squid's log les?

To disable *access.log*:

```
cache_access_log /dev/null
```

To disable *store.log*:

```
cache_store_log none
```

It is a bad idea to disable the *cache.log* because this le contains many important status and debugging messages. However, if you really want to, you can: To disable *access.log*:

```
cache_log /dev/null
```

6.15 My log les get very big!

You need to *rotate* your log les with a cron job. For example:

```
0 0 * * * /usr/local/squid/bin/squid -k rotate
```

6.16 I want to use another tool to maintain the log les.

If you set *logle _rotate* to 0, Squid simply closes and then re-opens the logs. This allows third-party logle management systems, such as *newsyslog*, to maintain the log les.

6.17 Managing log les

The preferred log le for analysis is the *access.log* le in native format. For long term evaluations, the log le should be obtained at regular intervals. Squid oers an easy to use API for rotating log les, in order that they may be moved (or removed) without disturbing the cache operations in progress. The procedures were described above.

Depending on the disk space allocated for log le storage, it is recommended to set up a cron job which rotates the log les every 24, 12, or 8 hour. You will need to set your *logle _rotate* to a suciently large number. During a time of some idleness, you can safely transfer the log les to your analysis host in one burst.

Before transport, the log les can be compressed during o-peak time. On the analysis host, the log le are concatinated into one le, so one le for 24 hours is the yield. Also note that with *log_icp_queries* enabled, you might have around 1 GB of uncompressed log information per day and busy cache. Look into you cache manager info page to make an educated guess on the size of your log les.

The EU project *DESIRE* `<http://www.desire.org/>` developed some *some basic rules* `<http://www.uninett.no/prosjekt/desire/arneberg/statistics.html>` to obey when handling and processing log les:

> Respect the privacy of your clients when publishing results.
>
> Keep logs unavailable unless anonymized. Most countries have laws on privacy protection, and some even on how long you are legally allowed to keep certain kinds of information.
>
> Rotate and process log les at least once a day. Even if you don't process the log les, they will grow quite large, see section 6.15. If you rely on processing the log les, reserve a large enough partition solely for log les.
>
> Keep the size in mind when processing. It might take longer to process log les than to generate them!
>
> Limit yourself to the numbers you are interested in. There is data beyond your dreams available in your log le, some quite obvious, others by combination of dierent views. Here are some examples for gures to watch:
>
> - The hosts using your cache.
> - The elapsed time for HTTP requests - this is the latency the user sees. Usually, you will want to make a distinction for HITs and MISSes and overall times. Also, medians are preferred over averages.
> - The requests handled per interval (e.g. second, minute or hour).

6.18 Why do I get ERR_NO_CLIENTS_BIG_OBJ messages so often?

This message means that the requested object was in "Delete Behind" mode and the user aborted the transfer. An object will go into "Delete Behind" mode if

> It is larger than *maximum_object_size*
>
> It is being fetched from a neighbor which has the *proxy-only* option set.

6.19 What does ERR_LIFETIME_EXP mean?

This means that a timeout occurred while the object was being transferred. Most likely the retrieval of this object was very slow (or it stalled before nishing) and the user aborted the request. However, depending on your settings for *quick_abort*, Squid may have continued to try retrieving the object. Squid imposes a maximum amount of time on all open sockets, so after some amount of time the stalled request was aborted and logged win an ERR_LIFETIME_EXP message.

6.20 Retrieving "lost" les from the cache

I've been asked to retrieve an object which was accidentally destroyed at the source for recovery. So, how do I gure out where the things are so I can copy them out and strip o the headers?

The following method applies only to the Squid-1.1 versions:

Use *grep* to nd the named object (Url) in the 6.11 le. The rst eld in this le is an integer *le number* .

Then, nd the le *leno-to-pathname.pl* from the "scripts" directory of the Squid source distribution. The usage is

```
perl fileno-to-pathname.pl [-c squid.conf]
```

le numbers are read on stdin, and pathnames are printed on stdout.

6.21 Can I use *store.log* to gure out if a response was cachable?

Sort of. You can use *store.log* to nd out if a particular response was *cached*.

Cached responses are logged with the SWAPOUT tag. Uncached responses are logged with the RELEASE tag.

However, your analysis must also consider that when a cached response is removed from the cache (for example due to cache replacement) it is also logged in *store.log* with the RELEASE tag. To dierentiate these two, you can look at the lenumber (3rd) eld. When an uncachable response is released, the lenumber is FFFFFFFF (-1). Any other lenumber indicates a cached response was released.

7 Operational issues

7.1 How do I see system level Squid statistics?

The Squid distribution includes a CGI utility called *cachemgr.cgi* which can be used to view squid statistics with a web browser. This document has a section devoted to *cachemgr.cgi* usage which you should consult for more information.

7.2 How can I nd the biggest objects in my cache?

```
sort -r -n +4 -5 access.log | awk '{print $5, $7}' | head -25
```

7.3 I want to restart Squid with a clean cache

Note: The information here is current for version 2.2.

First of all, you must stop Squid of course. You can use the command:

```
% squid -k shutdown
```

The fastest way to restart with an entirely clean cache is to over write the *swap.state* les for each *cache_dir* in your cong le. Note, you can not just remove the *swap.state* le, or truncate it to zero size. Instead, you should put just one byte of garbage there. For example:

```
% echo "" > /cache1/swap.state
```

Repeat that for every *cache_dir*, then restart Squid. Be sure to leave the *swap.state* le with the same owner and permissions that it had before!

Another way, which takes longer, is to have squid recreate all the *cache_dir* directories. But rst you must move the existing directories out of the way. For example, you can try this:

```
% cd /cache1
% mkdir JUNK
% mv ?? swap.state* JUNK
% rm -rf JUNK &
```

Repeat this for your other *cache_dir*'s, then tell Squid to create new directories:

```
% squid -z
```

7.4 How can I proxy/cache Real Audio?

by *Rodney van den Oever* <mailto:roever@nse.simac.nl>, and *James R Grinter* <mailto:jrg@blodwen.demon.co.uk>

Point the RealPlayer at your Squid server's HTTP port (e.g. 3128).

Using the Preferences->Transport tab, select *Use specied transports* and with the *Specied Transports* button, select use *HTTP Only*.

The RealPlayer (and RealPlayer Plus) manual states:

```
Use HTTP Only
        Select this option if you are behind a firewall and cannot
        receive data through TCP.  All data will be streamed through
        HTTP.

        Note:  You may not be able to receive some content if you select
        this option.
```

Again, from the documentation:

```
RealPlayer 4.0 identifies itself to the firewall when making a
request for content to a RealServer.  The following string is
attached to any URL that the Player requests using HTTP GET:

        /SmpDsBhgRl

Thus, to identify an HTTP GET request from the RealPlayer, look
for:

        http://[^/]+/SmpDsBhgRl

The Player can also be identified by the mime type in a POST to
the RealServer.  The RealPlayer POST has the following mime
type:

        "application/x-pncmd"
```

Note that the rst request is a POST, and the second has a '?' in the URL, so standard Squid congurations would treat it as non-cachable. It also looks rather "magic."

HTTP is an alternative delivery mechanism introduced with version 3 players, and it allows a reasonable approximation to "streaming" data - that is playing it as you receive it.

It isn't available in the general case: only if someone has made the realaudio le available via an HTTP server, or they're using a version 4 server, they've switched it on, and you're using a version 4 client. If someone has made the le available via their HTTP server, then it'll be cachable. Otherwise, it won't be (as far as we can tell.)

The more common RealAudio link connects via their own *pnm:* method and is transferred using their proprietary protocol (via TCP or UDP) and not using HTTP. It can't be cached nor proxied by Squid, and requires something such as the simple proxy that Progressive Networks themselves have made available, if you're in a rewall/no direct route situation. Their product does not cache (and I don't know of any software available that does.)

Some confusion arises because there is also a conguration option to use an HTTP proxy (such as Squid) with the Realaudio/RealVideo players. This is because the players can fetch the "`.ram`" le that contains the *pnm:* reference for the audio/video stream. They fetch that .ram le from an HTTP server, using HTTP.

7.5 How can I purge an object from my cache?

Squid does not allow you to purge objects unless it is congured with access controls in *squid.conf*. First you must add something like

```
acl PURGE method PURGE
acl localhost src 127.0.0.1
http_access allow PURGE localhost
http_access deny PURGE
```

The above only allows purge requests which come from the local host and denies all other purge requests.

To purge an object, you can use the *client* program:

```
client -m PURGE http://www.miscreant.com/
```

If the purge was successful, you will see a "200 OK" response:

```
HTTP/1.0 200 OK
Date: Thu, 17 Jul 1997 16:03:32 GMT
Server: Squid/1.1.14
```

If the object was not found in the cache, you will see a "404 Not Found" response:

```
HTTP/1.0 404 Not Found
Date: Thu, 17 Jul 1997 16:03:22 GMT
Server: Squid/1.1.14
```

7.6 Using ICMP to Measure the Network

As of version 1.1.9, Squid is able to utilize ICMP Round-Trip-Time (RTT) measurements to select the optimal location to forward a cache miss. Previously, cache misses would be forwarded to the parent cache which returned the rst ICP reply message. These were logged with FIRST _PARENT_MISS in the access.log le. Now we can select the parent which is closest (RTT-wise) to the origin server.

7.6.1 Supporting ICMP in your Squid cache

It is more important that your parent caches enable the ICMP features. If you are acting as a parent, then you may want to enable ICMP on your cache. Also, if your cache makes RTT measurements, it will fetch objects directly if your cache is closer than any of the parents.

If you want your Squid cache to measure RTT's to origin servers, Squid must be compiled with the USE_ICMP option. This is easily accomplished by uncommenting "-DUSE_ICMP=1" in *src/Makele* and/or *src/Makele.in* .

An external program called *pinger* is responsible for sending and receiving ICMP packets. It must run with root privileges. After Squid has been compiled, the pinger program must be installed separately. A special Makele target will install *pinger* with appropriate permissions.

```
% make install
% su
# make install-pinger
```

There are three conguration le options for tuning the measurement database on your cache. *netdb_low* and *netdb_high* specify high and low water marks for keeping the database to a certain size (e.g. just like with the IP cache). The *netdb_ttl* option species the minimum rate for pinging a site. If *netdb_ttl* is set to 300 seconds (5 minutes) then an ICMP packet will not be sent to the same site more than once every ve minutes. Note that a site is only pinged when an HTTP request for the site is received.

Another option, *minimum_direct_hops* can be used to try nding servers which are close to your cache. If the measured hop count to the origin server is less than or equal to *minimum_direct_hops*, the request will be forwarded directly to the origin server.

7.6.2 Utilizing your parents database

Your parent caches can be asked to include the RTT measurements in their ICP replies. To do this, you must enable *query_icmp* in your cong le:

```
query_icmp on
```

This causes a ag to be set in your outgoing ICP queries.

If your parent caches return ICMP RTT measurements then the eighth column of your access.log will have lines similar to:

```
CLOSEST_PARENT_MISS/it.cache.nlanr.net
```

In this case, it means that *it.cache.nlanr.net* returned the lowest RTT to the origin server. If your cache measured a lower RTT than any of the parents, the request will be logged with

```
CLOSEST_DIRECT/www.sample.com
```

7.6.3 Inspecting the database

The measurement database can be viewed from the cachemgr by selecting "Network Probe Database." Hostnames are aggregated into /24 networks. All measurements made are averaged over time. Measurements are made to specic hosts, taken from the URLs of HTTP requests. The recv and sent elds are the number of ICMP packets sent and received. At this time they are only informational.

A typical database entry looks something like this:

```
Network          recv/sent     RTT  Hops Hostnames
192.41.10.0        20/  21    82.3   6.0 www.jisedu.org www.dozo.com
    bo.cache.nlanr.net         42.0   7.0
    uc.cache.nlanr.net         48.0  10.0
    pb.cache.nlanr.net         55.0  10.0
    it.cache.nlanr.net        185.0  13.0
```

This means we have sent 21 pings to both www.jisedu.org and www.dozo.com. The average RTT is 82.3 milliseconds. The next four lines show the measured values from our parent caches. Since *bo.cache.nlanr.net* has the lowest RTT, it would be selected as the location to forward a request for a www.jisedu.org or www.dozo.com URL.

7.7 Why are so few requests logged as TCP_IMS_MISS?

When Squid receives an *If-Modied-Since* request, it will not forward the request unless the object needs to be refreshed according to the *refresh_pattern* rules. If the request does need to be refreshed, then it will be logged as TCP_REFRESH_HIT or TCP_REFRESH_MISS.

If the request is not forwarded, Squid replies to the IMS request according to the object in its cache. If the modication times are the same, then Squid returns TCP _IMS_HIT. If the modication times are dierent, then Squid returns TCP_IMS_MISS. In most cases, the cached object will not have changed, so the result is TCP_IMS_HIT. Squid will only return TCP_IMS_MISS if some other client causes a newer version of the object to be pulled into the cache.

7.8 How can I make Squid NOT cache some servers or URLs?

In Squid-2, you use the *no_cache* option to specify uncachable requests. For example, this makes all responses from origin servers in the 10.0.1.0/24 network uncachable:

```
acl Local dst 10.0.1.0/24
no_cache deny Local
```

This example makes all URL's with '.html' uncachable:

```
acl HTML url_regex .html$
no_cache deny HTML
```

This example makes a specic URL uncachable:

```
acl XYZZY url_regex ^http://www.i.suck.com/foo.html$
no_cache deny XYZZY
```

This example caches nothing between the hours of 8AM to 11AM:

```
acl Morning time 08:00-11:00
no_cache deny Morning
```

In Squid-1.1, whether or not an object gets cached is controlled by the *cache_stoplist*, and *cache_stoplist_pattern* options. So, you may add:

```
cache_stoplist my.domain.com
```

Specifying uncachable objects by IP address is harder. The *1.1 patch page* <../1.1/patches.html> includes a patch called *no-cache-local.patch* which changes the behaviour of the *local_ip* and *local_domain* so that matching requests are NOT CACHED, in addition to being fetched directly.

7.9 How can I delete and recreate a cache directory?

Deleting an existing cache directory is not too dicult. Unfortunately, you can't simply change squid.conf and then recongure. You can't stop using a *cache_dir* while Squid is running. Also note that Squid requires at least one *cache_dir* to run.

1. Edit your *squid.conf* le and comment out, or delete the *cache_dir* line for the cache directory that you want to remove.
2. If you don't have any *cache_dir* lines in your squid.conf, then Squid was using the default. You'll need to add a new *cache_dir* line because Squid will continue to use the default otherwise. You can add a small, temporary directory, fo example:

   ```
   /usr/local/squid/cachetmp ....
   ```

   If you add a new *cache_dir* you have to run *squid -z* to initialize that directory.
3. Remeber that you can not delete a cache directory from a running Squid process; you can not simply recongure squid. You must shutdown Squid:

   ```
   squid -k shutdown
   ```
4. Once Squid exits, you may immediately start it up again. Since you deleted the old *cache_dir* from squid.conf, Squid won't try to access that directory. If you use the RunCache script, Squid should start up again automatically.

5. Now Squid is no longer using the cache directory that you removed from the cong le. You can verify this by checking "Store Directory" information with the cache manager. From the command line, type:

```
client mgr:storedir
```

6. Now that Squid is not using the cache directory, you can *rm -rf* it, format the disk, build a new lesystem, or whatever.

The procedure is similar to recreate the directory.

1. Edit *squid.conf* and add a new *cache_dir* line.
2. Initialize the new directory by running

```
% squid -z
```

   NOTE: it is safe to run this even if Squid is already running. *squid -z* will harmlessly try to create all of the subdirectories that already exist.

3. Recongure Squid

```
squid -k reconfigure
```

   Unlike deleting, you can add new cache directories while Squid is already running.

7.10 Why can't I run Squid as root?

by Dave J Woolley

If someone were to discover a buer overrun bug in Squid and it runs as a user other than root, they can only corrupt the les writeable to that user, but if it runs a root, they can take over the whole machine. This applies to all programs that don't absolutely need root status, not just squid.

7.11 Can you tell me a good way to upgrade Squid with minimal downtime?

Here is a technique that was described by *Radu Greab* <mailto:radu@netsoft.ro>.

Start a second Squid server on an unused HTTP port (say 4128). This instance of Squid probably doesn't need a large disk cache. When this second server has nished reloading the disk store, swap the *http_port* values in the two *squid.conf* les. Set the original Squid to use port 5128, and the second one to use 3128. Next, run "squid -k recongure" for both Squids. New requests will go to the second Squid, now on port 3128 and the rst Squid will nish handling its current requests. After a few minutes, it should be safe to fully shut down the rst Squid and upgrade it. Later you can simply repeat this process in reverse.

7.12 Can Squid listen on more than one HTTP port?

Note: The information here is current for version 2.3.

Yes, you can specify multiple *http_port* lines in your *squid.conf* le. Squid attempts to bind() to each port that you specify. Sometimes Squid may not be able to bind to a port, either because of permissions or because the port is already in use. If Squid can bind to at least one port, then it will continue running. If it can not bind to any of the ports, then Squid stops.

With version 2.3 and later you can specify IP addresses and port numbers together (see the squid.conf comments).

7.13 Can I make origin servers see the client's IP address when going through Squid?

Normally you cannot. Most TCP/IP stacks do not allow applications to create sockets with the local endpoint assigned to a foreign IP address. However, some folks have some *patches to Linux* `<http://www.balabit.hu/en/downloads/tproxy/>` that allow exactly that.

In this situation, you must ensure that all HTTP packets destined for the client IP addresses are routed to the Squid box. If the packets take another path, the real clients will send TCP resets to the origin servers, thereby breaking the connections.

8 Memory

8.1 Why does Squid use so much memory!?

Squid uses a lot of memory for performance reasons. It takes much, much longer to read something from disk than it does to read directly from memory.

A small amount of metadata for each cached object is kept in memory. This is the *StoreEntry* data structure. For *Squid-2* this is 56-bytes on "small" pointer architectures (Intel, Sparc, MIPS, etc) and 88-bytes on "large" pointer architectures (Alpha). In addition, There is a 16-byte cache key (MD5 checksum) associated with each *StoreEntry*. This means there are 72 or 104 bytes of metadata in memory for every object in your cache. A cache with 1,000,000 objects therefore requires 72 MB of memory for *metadata only*. In practice it requires much more than that.

Squid-1.1 also uses a lot of memory to store in-transit objects. This version stores incoming objects only in memory, until the transfer is complete. At that point it decides whether or not to store the object on disk. This means that when users download large les, your memory usage will increase signicantly. The squid.conf parameter *maximum_object_size* determines how much memory an in-transit object can consume before we mark it as uncachable. When an object is marked uncachable, there is no need to keep all of the object in memory, so the memory is freed for the part of the object which has already been written to the client. In other words, lowering *maximum_object_size* also lowers Squid-1.1 memory usage.

Other uses of memory by Squid include:

Disk buers for reading and writing

Network I/O buers

IP Cache contents

FQDN Cache contents

Netdb ICMP measurement database

Per-request state information, including full request and reply headers

Miscellaneous statistics collection.

"Hot objects" which are kept entirely in memory.

8.2 How can I tell how much memory my Squid process is using?

One way is to simply look at *ps* output on your system. For BSD-ish systems, you probably want to use the *-u* option and look at the *VSZ* and *RSS* elds:

```
wessels  236% ps -axuhm
USER       PID %CPU %MEM   VSZ  RSS  TT  STAT STARTED       TIME COMMAND
squid     9631  4.6 26.4 141204 137852  ??  S    10:13PM   78:22.80 squid -NCYs
```

For SYSV-ish, you probably want to use the *-l* option. When interpreting the *ps* output, be sure to check your *ps* manual page. It may not be obvious if the reported numbers are kbytes, or pages (usually 4 kb).

A nicer way to check the memory usage is with a program called *top*:

```
        last pid: 20128;  load averages:  0.06,  0.12,  0.11                    14:10:58
46 processes:  1 running, 45 sleeping
CPU states:     % user,     % nice,     % system,     % interrupt,     % idle
Mem: 187M Active, 1884K Inact, 45M Wired, 268M Cache, 8351K Buf, 1296K Free
Swap: 1024M Total, 256K Used, 1024M Free

  PID USERNAME PRI NICE SIZE    RES STATE    TIME   WCPU    CPU COMMAND
 9631 squid     2   0   138M   135M select  78:45  3.93%  3.93% squid
```

Finally, you can ask the Squid process to report its own memory usage. This is available on the Cache Manager *info* page. Your output may vary depending upon your operating system and Squid version, but it looks similar to this:

```
Resource usage for squid:
        Maximum Resident Size: 137892 KB
Memory usage for squid via mstats():
        Total space in arena:  140144 KB
        Total free:              8153 KB 6%
```

If your RSS (Resident Set Size) value is much lower than your process size, then your cache performance is most likely suering due to 9.24.

8.3 My Squid process grows without bounds.

You might just have your *cache_mem* parameter set too high. See the "8.9" entry below.

When a process continually grows in size, without levelling o or slowing down, it often indicates a memory leak. A memory leak is when some chunk of memory is used, but not free'd when it is done being used.

Memory leaks are a real problem for programs (like Squid) which do all of their processing within a single process. Historically, Squid has had real memory leak problems. But as the software has matured, we believe almost all of Squid's memory leaks have been eliminated, and new ones are least easy to identify.

Memory leaks may also be present in your system's libraries, such as *libc.a* or even *libmalloc.a*. If you experience the ever-growing process size phenomenon, we suggest you rst try an 8.10.

8.4 I set *cache_mem* to XX, but the process grows beyond that!

The *cache_mem* parameter **does NOT** specify the maximum size of the process. It only species how much memory to use for caching "hot" (very popular) replies. Squid's actual memory usage is depends very strongly on your cache size (disk space) and your incoming request load. Reducing *cache_mem* will usually also reduce the process size, but not necessarily, and there are other ways to reduce Squid's memory usage (see below).

See also 8.11.

8.5 How do I analyze memory usage from the cache manger output?

Note: This information is specic to Squid-1.1 versions

Look at your *cachemgr.cgi* `Cache Information` page. For example:

```
Memory usage for squid via mallinfo():
       Total space in arena:   94687 KB
       Ordinary blocks:        32019 KB 210034 blks
       Small blocks:           44364 KB 569500 blks
       Holding blocks:             0 KB   5695 blks
       Free Small blocks:       6650 KB
       Free Ordinary blocks:   11652 KB
       Total in use:           76384 KB 81%
       Total free:             18302 KB 19%

Meta Data:
StoreEntry               246043 x 64 bytes =  15377 KB
IPCacheEntry             971 x   88 bytes  =     83 KB
Hash link                2 x   24 bytes    =      0 KB
URL strings                                 =  11422 KB
Pool MemObject structures 514 x  144 bytes  =     72 KB (    70 free)
Pool for Request structur 516 x 4380 bytes  =   2207 KB (  2121 free)
Pool for in-memory object 6200 x 4096 bytes =  24800 KB ( 22888 free)
Pool for disk I/O         242 x 8192 bytes =   1936 KB (  1888 free)
Miscellaneous                               =   2600 KB
total Accounted                             =  58499 KB
```

First note that `mallinfo()` reports 94M in "arena." This is pretty close to what *top* says (97M).

Of that 94M, 81% (76M) is actually being used at the moment. The rest has been freed, or pre-allocated by `malloc(3)` and not yet used.

Of the 76M in use, we can account for 58.5M (76%). There are some calls to `malloc(3)` for which we can't account.

The `Meta Data` list gives the breakdown of where the accounted memory has gone. 45% has gone to `StoreEntry` and URL strings. Another 42% has gone to buering hold objects in VM while they are fetched and relayed to the clients (`Pool for in-memory object`).

The pool sizes are specied by *squid.conf* parameters. In version 1.0, these pools are somewhat broken: we keep a stack of unused pages instead of freeing the block. In the `Pool for in-memory object`, the unused stack size is 1/2 of `cache_mem`. The `Pool for disk I/O` is hardcoded at 200. For `MemObject` and `Request` it's 1/8 of your system's `FD_SETSIZE` value.

If you need to lower your process size, we recommend lowering the max object sizes in the 'http', 'ftp' and 'gopher' cong lines. You may also want to lower `cache_mem` to suit your needs. But if you `make cache_mem` too low, then some objects may not get saved to disk during high-load periods. Newer Squid versions allow you to set `memory_pools off` to disable the free memory pools.

8.6 The "Total memory accounted" value is less than the size of my Squid process.

We are not able to account for *all* memory that Squid uses. This would require excessive amounts of code to keep track of every last byte. We do our best to account for the major uses of memory.

Also, note that the *malloc* and *free* functions have their own overhead. Some additional memory is required to keep track of which chunks are in use, and which are free. Additionally, most operating systems do not allow processes to shrink in size. When a process gives up memory by calling *free*, the total process size does not shrink. So the process size really represents the maximum size your Squid process has reached.

8.7 xmalloc: Unable to allocate 4096 bytes!

by *Henrik Nordstrom* <mailto:hno@squid-cache.org>

Messages like "FATAL: xcalloc: Unable to allocate 4096 blocks of 1 bytes!" appear when Squid can't allocate more memory, and on most operating systems (inclusive BSD) there are only two possible reasons:

1. The machine is out of swap
2. The process' maximum data segment size has been reached

The rst case is detected using the normal swap monitoring tools available on the platform (*pstat* on SunOS, perhaps *pstat* is used on BSD as well).

To tell if it is the second case, rst rule out the rst case and then monitor the size of the Squid process. If it dies at a certain size with plenty of swap left then the max data segment size is reached without no doubts.

The data segment size can be limited by two factors:

1. Kernel imposed maximum, which no user can go above
2. The size set with ulimit, which the user can control.

When squid starts it sets data and le ulimit's to the hard level. If you manually tune ulimit before starting Squid make sure that you set the hard limit and not only the soft limit (the default operation of ulimit is to only change the soft limit). root is allowed to raise the soft limit above the hard limit.

This command prints the hard limits:

```
ulimit -aH
```

This command sets the data size to unlimited:

```
ulimit -HSd unlimited
```

8.7.1 BSD/OS

by *Arjan de Vet* <mailto:Arjan.deVet@adv.IAEhv.nl>

The default kernel limit on BSD/OS for datasize is 64MB (at least on 3.0 which I'm using).

Recompile a kernel with larger datasize settings:

```
maxusers        128
# Support for large inpcb hash tables, e.g. busy WEB servers.
options         INET_SERVER
# support for large routing tables, e.g. gated with full Internet routing:
options         "KMEMSIZE=\(16*1024*1024\)"
options         "DFLDSIZ=\(128*1024*1024\)"
options         "DFLSSIZ=\(8*1024*1024\)"
options         "SOMAXCONN=128"
options         "MAXDSIZ=\(256*1024*1024\)"
```

See */usr/share/doc/bsdi/cong.n* for more info.

In /etc/login.conf I have this:

```
default:\
        :path=/bin /usr/bin /usr/contrib/bin:\
        :datasize-cur=256M:\
        :openfiles-cur=1024:\
        :openfiles-max=1024:\
        :maxproc-cur=1024:\
        :stacksize-cur=64M:\
        :radius-challenge-styles=activ,crypto,skey,snk,token:\
        :tc=auth-bsdi-defaults:\
        :tc=auth-ftp-bsdi-defaults:

#
# Settings used by /etc/rc and root
# This must be set properly for daemons started as root by inetd as well.
# Be sure reset these values back to system defaults in the default class!
#
daemon:\
        :path=/bin /usr/bin /sbin /usr/sbin:\
        :widepasswords:\
        :tc=default:
#       :datasize-cur=128M:\
#       :openfiles-cur=256:\
#       :maxproc-cur=256:\
```

This should give enough space for a 256MB squid process.

8.7.2 FreeBSD (2.2.X)

by Duane Wessels

The procedure is almost identical to that for BSD/OS above. Increase the open ledescriptor limit in */sys/conf/param.c*:

```
int     maxfiles = 4096;
int     maxfilesperproc = 1024;
```

Increase the maximum and default data segment size in your kernel cong le, e.g. */sys/conf/i386/CONFIG*:

```
options         "MAXDSIZ=(512*1024*1024)"
options         "DFLDSIZ=(128*1024*1024)"
```

We also found it necessary to increase the number of mbuf clusters:

```
options         "NMBCLUSTERS=10240"
```

And, if you have more than 256 MB of physical memory, you probably have to disable BOUNCE_BUFFERS (whatever that is), so comment out this line:

```
#options        BOUNCE_BUFFERS          #include support for DMA bounce buffers
```

Also, update limits in */etc/login.conf*:

```
# Settings used by /etc/rc
#
daemon:\
        :coredumpsize=infinity:\
        :datasize=infinity:\
        :maxproc=256:\
        :maxproc-cur@:\
        :memoryuse-cur=64M:\
        :memorylocked-cur=64M:\
        :openfiles=4096:\
        :openfiles-cur@:\
        :stacksize=64M:\
        :tc=default:
```

And don't forget to run "cap_mkdb /etc/login.conf" after editing that le.

8.7.3 OSF, Digital Unix

by *Ong Beng Hui* <mailto:ongbh@zpoprp.zpo.dec.com>

To increase the data size for Digital UNIX, edit the le `/etc/sysconfigtab` and add the entry...

```
proc:
        per-proc-data-size=1073741824
```

Or, with csh, use the limit command, such as

```
> limit datasize 1024M
```

Editing `/etc/sysconfigtab` requires a reboot, but the limit command doesn't.

8.8 fork: (12) Cannot allocate memory

When Squid is recongured (SIGHUP) or the logs are rotated (SIGUSR1), some of the helper processes (dnsserver) must be killed and restarted. If your system does not have enough virtual memory, the Squid process may not be able to fork to start the new helper processes. This is due to the UNIX way of starting child processes using the fork() system call which temporary duplicates the whole Squid process, and when

rapidly starting many child processes such as on "squid -k rotate" the memory usage can temporarily grow to many times the normal memory usage due to several temporary copies of the whole process.

The best way to x this is to increase your virtual memory by adding swap space. Normally your system uses raw disk partitions for swap space, but most operating systems also support swapping on regular les (Digital Unix excepted). See your system manual pages for *swap*, *swapon*, and *mkle* . Alternatively you can use the sleep_after_fork directive to make Squid sleep a little while invoking helpers to allow the helper to start up before trying to start the next one. This can be helpful if you nd that Squid sometimes fail to restart all helpers on "squid -k recongure".

8.9 What can I do to reduce Squid's memory usage?

If your cache performance is suering because of memory limitations, you might consider buying more memory. But if that is not an option, There are a number of things to try:

Try a 8.10.

Reduce the *cache_mem* parameter in the cong le. This controls how many "hot" objects are kept in memory. Reducing this parameter will not signicantly aect performance, but you may recieve some warnings in *cache.log* if your cache is busy.

Turn the *memory_pools o* in the cong le. This causes Squid to give up unused memory by calling *free()* instead of holding on to the chunk for potential, future use.

Reduce the *cache_swap* parameter in your cong le. This will reduce the number of objects Squid keeps. Your overall hit ratio may go down a little, but your cache will perform signicantly better.

Reduce the *maximum_object_size* parameter (Squid-1.1 only). You won't be able to cache the larger objects, and your byte volume hit ratio may go down, but Squid will perform better overall.

If you are using Squid-1.1.x, try the "NOVM" version.

8.10 Using an alternate *malloc* library.

Many users have found improved performance and memory utilization when linking Squid with an external malloc library. We recommend either GNU malloc, or dlmalloc.

8.10.1 Using GNU malloc

To make Squid use GNU malloc follow these simple steps:

1. Download the GNU malloc source, available from one of *The GNU FTP Mirror sites* <http://www.gnu.org/order/ftp.html>.
2. Compile GNU malloc

```
% gzip -dc malloc.tar.gz | tar xf -
% cd malloc
% vi Makefile     # edit as needed
% make
```

3. Copy libmalloc.a to your system's library directory and be sure to name it *libgnumalloc.a*.

```
% su
# cp malloc.a /usr/lib/libgnumalloc.a
```

4. (Optional) Copy the GNU malloc.h to your system's include directory and be sure to name it *gnumalloc.h*. This step is not required, but if you do this, then Squid will be able to use the *mstat()* function to report memory usage statistics on the cachemgr info page.

```
# cp malloc.h /usr/include/gnumalloc.h
```

5. Recongure and recompile Squid

```
% make realclean
% ./configure ...
% make
% make install
```

Note, In later distributions, 'realclean' has been changed to 'distclean'. As the congure script runs, watch its output. You should nd that it locates libgnumalloc.a and optionally gnumalloc.h.

8.10.2 dlmalloc

dlmalloc <http://g.oswego.edu/dl/html/malloc.html> has been written by *Doug Lea* <mailto:dl@cs.oswego.edu>. According to Doug:

> This is not the fastest, most space-conserving, most portable, or most tunable malloc ever written. However it is among the fastest while also being among the most space-conserving, portable and tunable.

dlmalloc is included with the *Squid-2* source distribution. To use this library, you simply give an option to the *congure* script:

```
% ./configure --enable-dlmalloc ...
```

8.11 How much memory do I need in my Squid server?

As a rule of thumb on Squid uses approximately 10 MB of RAM per GB of the total of all cache_dirs (more on 64 bit servers such as Alpha), plus your cache_mem setting and about an additional 10-20MB. It is recommended to have at least twice this amount of physical RAM available on your Squid server. For a more detailed discussion on Squid's memory usage see the sections above.

The recommended extra RAM besides what is used by Squid is used by the operating system to improve disk I/O performance and by other applications or services running on the server. This will be true even of a server which runs Squid as the only tcp service, since there is a minimum level of memory needed for process management, logging, and other OS level routines.

If you have a low memory server, and a large disk, then you will not necessarily be able to use all the disk space, since as the cache lls the memory available will be insucient, forcing Squid to swap out memory and aecting performance. A very large cache _dir total and insucient physical RAM + Swap could cause Squid to stop functioning completely. The solution for larger caches is to get more physical RAM; allocating more to Squid via cache_mem will not help.

9 The Cache Manager

by *Jonathan Larmour* <mailto:JLarmour@origin-at.co.uk>

9.1 What is the cache manager?

The cache manager (*cachemgr.cgi*) is a CGI utility for displaying statistics about the *squid* process as it runs. The cache manager is a convenient way to manage the cache and view statistics without logging into the server.

9.2 How do you set it up?

That depends on which web server you're using. Below you will nd instructions for conguring the CERN and Apache servers to permit *cachemgr.cgi* usage.

EDITOR"S NOTE: readers are encouraged to submit instructions for conguration of cachemgr.cgi on other web server platforms, such as Netscape.

After you edit the server conguration les, you will probably need to either restart your web server or or send it a `SIGHUP` signal to tell it to re-read its conguration les.

When you're done conguring your web server, you'll connect to the cache manager with a web browser, using a URL such as:

```
http://www.example.com/Squid/cgi-bin/cachemgr.cgi/
```

9.3 Cache manager conguration for CERN httpd 3.0

First, you should ensure that only specied workstations can access the cache manager. That is done in your CERN *httpd.conf*, not in *squid.conf*.

```
Protection MGR-PROT {
        Mask    @(workstation.example.com)
}
```

Wildcards are acceptable, IP addresses are acceptable, and others can be added with a comma-separated list of IP addresses. There are many more ways of protection. Your server documentation has details.

You also need to add:

```
Protect         /Squid/*        MGR-PROT
Exec            /Squid/cgi-bin/*.cgi    /usr/local/squid/bin/*.cgi
```

This marks the script as executable to those in `MGR-PROT`.

9.4 Cache manager conguration for Apache

First, make sure the cgi-bin directory you're using is listed with a `ScriptAlias` in your Apache *httpd.conf* le like this:

```
ScriptAlias /Squid/cgi-bin/ /usr/local/squid/cgi-bin/
```

It's probably a **bad** idea to `ScriptAlias` the entire usr/local/squid/bin/ directory where all the Squid executables live.

Next, you should ensure that only specied workstations can access the cache manager. That is done in your Apache *httpd.conf*, not in *squid.conf*. At the bottom of *httpd.conf* le, insert:

```
<Location /Squid/cgi-bin/cachemgr.cgi>
order allow,deny
allow from workstation.example.com
</Location>
```

You can have more than one allow line, and you can allow domains or networks.

Alternately, *cachemgr.cgi* can be password-protected. You'd add the following to *httpd.conf*:

```
<Location /Squid/cgi-bin/cachemgr.cgi>
AuthUserFile /path/to/password/file
AuthGroupFile /dev/null
AuthName User/Password Required
AuthType Basic
require user cachemanager
</Location>
```

Consult the Apache documentation for information on using *htpasswd* to set a password for this "user."

9.5 Cache manager conguration for Roxen 2.0 and later

by Francesco "kinkie" Chemolli

Notice: this is *not* how things would get best done with Roxen, but this what you need to do go adhere to the example. Also, knowledge of basic Roxen conguration is required.

This is what's required to start up a fresh Virtual Server, only serving the cache manager. If you already have some Virtual Server you wish to use to host the Cache Manager, just add a new CGI support module to it.

Create a new virtual server, and set it to host http://www.example.com/. Add to it at least the following modules:

Content Types

CGI scripting support

In the *CGI scripting support* module, section *Settings*, change the following settings:

CGI-bin path: set to /Squid/cgi-bin/

Handle *.cgi: set to *no*

Run user scripts as owner: set to *no*

Search path: set to the directory containing the cachemgr.cgi le

In section *Security*, set *Patterns* to:

```
allow ip=1.2.3.4
```

where 1.2.3.4 is the IP address for workstation.example.com

Save the conguration, and you're done.

9.6 Cache manager ACLs in *squid.conf*

The default cache manager access conguration in *squid.conf* is:

```
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl all src 0.0.0.0/0.0.0.0
```

With the following rules:

```
http_access deny manager !localhost
http_access allow all
```

The rst ACL is the most important as the cache manager program interrogates squid using a special `cache_object` protocol. Try it yourself by doing:

```
telnet mycache.example.com 3128
GET cache_object://mycache.example.com/info HTTP/1.0
```

The default ACLs say that if the request is for a `cache_object`, and it isn't the local host, then deny access; otherwise allow access.

In fact, only allowing localhost access means that on the initial *cachemgr.cgi* form you can only specify the cache host as `localhost`. We recommend the following:

```
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl example src 123.123.123.123/255.255.255.255
acl all src 0.0.0.0/0.0.0.0
```

Where `123.123.123.123` is the IP address of your web server. Then modify the rules like this:

```
http_access allow manager localhost
http_access allow manager example
http_access deny manager
http_access allow all
```

If you're using *miss_access*, then don't forget to also add a *miss_access* rule for the cache manager:

```
miss_access allow manager
```

The default ACLs assume that your web server is on the same machine as *squid*. Remember that the connection from the cache manager program to squid originates at the web server, not the browser. So if your web server lives somewhere else, you should make sure that IP address of the web server that has *cachemgr.cgi* installed on it is in the `example` ACL above.

Always be sure to send a `SIGHUP` signal to *squid* any time you change the *squid.conf* le.

9.7 Why does it say I need a password and a URL?

If you "drop" the list box, and browse it, you will see that the password is only required to shutdown the cache, and the URL is required to refresh an object (i.e., retrieve it from its original source again) Otherwise these elds can be left blank: a password is not required to obtain access to the informational aspects of *cachemgr.cgi*.

9.8 I want to shutdown the cache remotely. What's the password?

See the `cachemgr_passwd` directive in *squid.conf*.

9.9 How do I make the cache host default to *my* cache?

When you run *congure* use the *–enable-cachemgr-hostname* option:

```
% ./configure --enable-cachemgr-hostname=‘hostname‘ ...
```

Note, if you do this after you already installed Squid before, you need to make sure *cachemgr.cgi* gets recompiled. For example:

```
% cd src
% rm cachemgr.o cachemgr.cgi
% make cachemgr.cgi
```

Then copy *cachemgr.cgi* to your HTTP server's *cgi-bin* directory.

9.10 What's the dierence between Squid TCP connections and Squid UDP connections?

Browsers and caches use TCP connections to retrieve web objects from web servers or caches. UDP connections are used when another cache using you as a sibling or parent wants to nd out if you have an object in your cache that it's looking for. The UDP connections are ICP queries.

9.11 It says the storage expiration will happen in 1970!

Don't worry. The default (and sensible) behavior of *squid* is to expire an object when it happens to overwrite it. It doesn't explicitly garbage collect (unless you tell it to in other ways).

9.12 What do the Meta Data entries mean?

StoreEntry

Entry describing an object in the cache.

IPCacheEntry

An entry in the DNS cache.

Hash link

Link in the cache hash table structure.

URL strings

The strings of the URLs themselves that map to an object number in the cache, allowing access to the StoreEntry.

Basically just like the `log` le in your cache directory:

1. `PoolMemObject structures`
2. Info about objects currently in memory, (eg, in the process of being transferred).
3. `Pool for Request structures`
4. Information about each request as it happens.
5. `Pool for in-memory object`
6. Space for object data as it is retrieved.

If *squid* is much smaller than this eld, run for cover! Something is very wrong, and you should probably restart *squid*.

9.13 In the utilization section, what is `Other`?

`Other` is a default category to track objects which don't fall into one of the dened categories.

9.14 In the utilization section, why is the `Transfer KB/sec` column always zero?

This column contains gross estimations of data transfer rates averaged over the entire time the cache has been running. These numbers are unreliable and mostly useless.

9.15 In the utilization section, what is the `Object Count`?

The number of objects of that type in the cache right now.

9.16 In the utilization section, what is the `Max/Current/Min KB`?

These refer to the size all the objects of this type have grown to/currently are/shrunk to.

9.17 What is the `I/O` section about?

These are histograms on the number of bytes read from the network per `read(2)` call. Somewhat useful for determining maximum buer sizes.

9.18 What is the `Objects` section for?

Warning: this will download to your browser a list of every URL in the cache and statistics about it. It can be very, very large. ***Sometimes it will be larger than the amount of available memory in your client!*** You probably don't need this information anyway.

9.19 What is the `VM Objects` section for?

`VM Objects` are the objects which are in Virtual Memory. These are objects which are currently being retrieved and those which were kept in memory for fast access (accelerator mode).

9.20 What does `AVG RTT` mean?

Average Round Trip Time. This is how long on average after an ICP ping is sent that a reply is received.

9.21 In the IP cache section, what's the dierence between a hit, a negative hit and a miss?

A HIT means that the document was found in the cache. A MISS, that it wasn't found in the cache. A negative hit means that it was found in the cache, but it doesn't exist.

9.22 What do the IP cache contents mean anyway?

The hostname is the name that was requested to be resolved.

For the `Flags` column:

`C` Means positively cached.

`N` Means negatively cached.

`P` Means the request is pending being dispatched.

`D` Means the request has been dispatched and we're waiting for an answer.

`L` Means it is a locked entry because it represents a parent or sibling.

The `TTL` column represents "Time To Live" (i.e., how long the cache entry is valid). (May be negative if the document has expired.)

The `N` column is the number of IP addresses from which the cache has documents.

The rest of the line lists all the IP addresses that have been associated with that IP cache entry.

9.23 What is the fqdncache and how is it dierent from the ipcache?

IPCache contains data for the Hostname to IP-Number mapping, and FQDNCache does it the other way round. For example:

IP Cache Contents:

```
Hostname                      Flags lstref    TTL  N [IP-Number]
gorn.cc.fh-lippe.de               C       0  21581 1 193.16.112.73
lagrange.uni-paderborn.de         C       6  21594 1 131.234.128.245
www.altavista.digital.com         C      10  21299 4 204.123.2.75  ...
2/ftp.symantec.com                DL   1583 -772855 0

Flags:  C --> Cached
        D --> Dispatched
```

```
        N --> Negative Cached
        L --> Locked
lstref: Time since last use
TTL:    Time-To-Live until information expires
N:      Count of addresses
```

FQDN Cache Contents:

```
IP-Number                    Flags    TTL N Hostname
130.149.17.15                    C -45570 1 andele.cs.tu-berlin.de
194.77.122.18                    C -58133 1 komet.teuto.de
206.155.117.51                   N -73747 0

Flags:  C --> Cached
        D --> Dispatched
        N --> Negative Cached
        L --> Locked
TTL:    Time-To-Live until information expires
N:      Count of names
```

9.24 What does "Page faults with physical i/o: 4897" mean?

This question was asked on the *squid-users* mailing list, to which there were three excellent replies.

by *Jonathan Larmour* <mailto:JLarmour@origin-at.co.uk>

You get a "page fault" when your OS tries to access something in memory which is actually swapped to disk. The term "page fault" while correct at the kernel and CPU level, is a bit deceptive to a user, as there's no actual error - this is a normal feature of operation.

Also, this doesn't necessarily mean your squid is swapping by that much. Most operating systems also implement paging for executables, so that only sections of the executable which are actually used are read from disk into memory. Also, whenever squid needs more memory, the fact that the memory was allocated will show up in the page faults.

However, if the number of faults is unusually high, and getting bigger, this could mean that squid is swapping. Another way to verify this is using a program called "vmstat" which is found on most UNIX platforms. If you run this as "vmstat 5" this will update a display every 5 seconds. This can tell you if the system as a whole is swapping a lot (see your local man page for vmstat for more information).

It is very bad for squid to swap, as every single request will be blocked until the requested data is swapped in. It is better to tweak the *cache_mem* and/or *memory_pools* setting in squid.conf, or switch to the NOVM versions of squid, than allow this to happen.

by *Peter Wemm* <mailto:peter@spinner.dialix.com.au>

There's two dierent operations at work, Paging and swapping. Paging is when individual pages are shued (either discarded or swapped to/from disk), while "swapping" *generally* means the entire process got sent to/from disk.

Needless to say, swapping a process is a pretty drastic event, and usually only reserved for when there's a memory crunch and paging out cannot free enough memory quickly enough. Also, there's some variation on how swapping is implemented in OS's. Some don't do it at all or do a hybrid of paging and swapping instead.

As you say, paging out doesn't necessarily involve disk IO, eg: text (code) pages are read-only and can simply be discarded if they are not used (and reloaded if/when needed). Data pages are also discarded if unmodied, and paged out if there's been any changes. Allocated memory (malloc) is always saved to disk since there's no executable le to recover the data from. mmap() memory is variable.. If it's backed from a le, it uses the same rules as the data segment of a le - ie: either discarded if unmodied or paged out.

There's also "demand zeroing" of pages as well that cause faults.. If you malloc memory and it calls brk()/sbrk() to allocate new pages, the chances are that you are allocated demand zero pages. Ie: the pages are not "really" attached to your process yet, but when you access them for the rst time, the page fault causes the page to be connected to the process address space and zeroed - this saves unnecessary zeroing of pages that are allocated but never used.

The "page faults with physical IO" comes from the OS via getrusage(). It's highly OS dependent on what it means. Generally, it means that the process accessed a page that was not present in memory (for whatever reason) and there was disk access to fetch it. Many OS's load executables by demand paging as well, so the act of starting squid implicitly causes page faults with disk IO - however, many (but not all) OS's use "read ahead" and "prefault" heuristics to streamline the loading. Some OS's maintain "intent queues" so that pages can be selected as pageout candidates ahead of time. When (say) squid touches a freshly allocated demand zero page and one is needed, the OS can page out one of the candidates on the spot, causing a 'fault with physical IO' with demand zeroing of allocated memory which doesn't happen on many other OS's. (The other OS's generally put the process to sleep while the pageout daemon nds a page for it).

The meaning of "swapping" varies. On FreeBSD for example, swapping out is implemented as unlocking upages, kernel stack, PTD etc for aggressive pageout with the process. The only thing left of the process in memory is the 'struct proc'. The FreeBSD paging system is highly adaptive and can resort to paging in a way that is equivalent to the traditional swapping style operation (ie: entire process). FreeBSD also tries stealing pages from active processes in order to make space for disk cache. I suspect this is why setting 'memory_pools o' on the non-NOVM squids on FreeBSD is reported to work better - the VM/buer system could be competing with squid to cache the same pages. It's a pity that squid cannot use mmap() to do le IO on the 4K chunks in it's memory pool (I can see that this is not a simple thing to do though, but that won't stop me wishing. :-).

by *John Line* `<mailto:webadm@info.cam.ac.uk>`

The comments so far have been about what paging/swapping gures mean in a "traditional" context, but it's worth bearing in mind that on some systems (Sun's Solaris 2, at least), the virtual memory and lesystem handling are unied and what a user process sees as reading or writing a le, the system simply sees as paging something in from disk or a page being updated so it needs to be paged out. (I suppose you could view it as similar to the operating system memory-mapping the les behind-the-scenes.)

The eect of this is that on Solaris 2, paging gures will also include le I/O. Or rather, the gures from vmstat certainly appear to include le I/O, and I presume (but can't quickly test) that gures such as those quoted by Squid will also include le I/O.

To conrm the above (which represents an impression from what I've read and observed, rather than 100% certain facts...), using an otherwise idle Sun Ultra 1 system system I just tried using cat (small, shouldn't need to page) to copy (a) one le to another, (b) a le to /dev/null, (c) /dev/zero to a le, and (d) /dev/zero to /dev/null (interrupting the last two with control-C after a while!), while watching with vmstat. 300-600 page-ins or page-outs per second when reading or writing a le (rather than a device), essentially zero in other cases (and when not cat-ing).

So ... beware assuming that all systems are similar and that paging gures represent *only* program code and data being shued to/from disk - they may also include the work in reading/writing all those les you were accessing...

9.24.1 Ok, so what is unusually high?

You'll probably want to compare the number of page faults to the number of HTTP requests. If this ratio is close to, or exceeding 1, then Squid is paging too much.

9.25 What does the IGNORED eld mean in the 'cache server list'?

This refers to ICP replies which Squid ignored, for one of these reasons:

The URL in the reply could not be found in the cache at all.

The URL in the reply was already being fetched. Probably this ICP reply arrived too late.

The URL in the reply did not have a MemObject associated with it. Either the request is already nished, or the user aborted before the ICP arrived.

The reply came from a multicast-responder, but the *cache_peer_access* conguration does not allow us to forward this request to that neighbor.

Source-Echo replies from known neighbors are ignored.

ICP_OP_DENIED replies are ignored after the rst 100.

10 Access Controls

10.1 Introduction

Squid's access control scheme is relatively comprehensive and dicult for some people to understand. There are two dierent components: *ACL elements*, and *access lists*. An access list consists of an *allow* or *deny* action followed by a number of ACL elements.

10.1.1 ACL elements

Note: The information here is current for version 2.4.

Squid knows about the following types of ACL elements:

src: source (client) IP addresses

dst: destination (server) IP addresses

myip: the local IP address of a client's connection

srcdomain: source (client) domain name

dstdomain: destination (server) domain name

srcdom_regex: source (client) regular expression pattern matching

dstdom_regex: destination (server) regular expression pattern matching

time: time of day, and day of week

url_regex: URL regular expression pattern matching

urlpath_regex: URL-path regular expression pattern matching, leaves out the protocol and hostname

port: destination (server) port number

myport: local port number that client connected to

proto: transfer protocol (http, ftp, etc)

method: HTTP request method (get, post, etc)

browser: regular expression pattern matching on the request's user-agent header

ident: string matching on the user's name

ident_regex: regular expression pattern matching on the user's name

src_as: source (client) Autonomous System number

dst_as: destination (server) Autonomous System number

proxy_auth: user authentication via external processes

proxy_auth_regex: user authentication via external processes

snmp_community: SNMP community string matching

maxconn: a limit on the maximum number of connections from a single client IP address

req_mime_type: regular expression pattern matching on the request content-type header

arp: Ethernet (MAC) address matching

Notes:

Not all of the ACL elements can be used with all types of access lists (described below). For example, *snmp_community* is only meaningful when used with *snmp_access*. The *src_as* and *dst_as* types are only used in *cache_peer_access* access lists.

The *arp* ACL requires the special congure option –enable-arp-acl. Furthermore, the ARP ACL code is not portable to all operating systems. It works on Linux, Solaris, and some *BSD variants.

The SNMP ACL element and access list require the –enable-snmp congure option.

Some ACL elements can cause processing delays. For example, use of *src_domain* and *srcdom_regex* require a reverse DNS lookup on the client's IP address. This lookup adds some delay to the request.

Each ACL element is assigned a unique *name*. A named ACL element consists of a *list of values*. When checking for a match, the multiple values use OR logic. In other words, an ACL element is *matched* when any one of its values is a match.

You can't give the same name to two dierent types of ACL elements. It will generate a syntax error.

You can put dierent values for the same ACL name on dierent lines. Squid combines them into one list.

10.1.2 Access Lists

There are a number of dierent access lists:

http_access: Allows HTTP clients (browsers) to access the HTTP port. This is the primary access control list.

icp_access: Allows neighbor caches to query your cache with ICP.

miss_access: Allows certain clients to forward cache misses through your cache.

no_cache: Denes responses that should not be cached.

redirector_access: Controls which requests are sent through the redirector pool.

ident_lookup_access: Controls which requests need an Ident lookup.

always_direct: Controls which requests should always be forwarded directly to origin servers.

never_direct: Controls which requests should never be forwarded directly to origin servers.

snmp_access: Controls SNMP client access to the cache.

broken_posts: Denes requests for which squid appends an extra CRLF after POST message bodies as required by some broken origin servers.

cache_peer_access: Controls which requests can be forwarded to a given neighbor (peer).

Notes:

An access list *rule* consists of an *allow* or *deny* keyword, followed by a list of ACL element names.

An access list consists of one or more access list rules.

Access list rules are checked in the order they are written. List searching terminates as soon as one of the rules is a match.

If a rule has multiple ACL elements, it uses AND logic. In other words, *all* ACL elements of the rule must be a match in order for the rule to be a match. This means that it is possible to write a rule that can never be matched. For example, a port number can never be equal to both 80 AND 8000 at the same time.

To summarise the acl logics can be described as:

```
http_access allow|deny acl AND acl AND ...
    OR
http_access allow|deny acl AND acl AND ...
    OR
...
```

If none of the rules are matched, then the default action is the *opposite* of the last rule in the list. Its a good idea to be explicit with the default action. The best way is to thse the *all* ACL. For example:

```
acl all src 0/0
http_access deny all
```

10.2 How do I allow my clients to use the cache?

Dene an ACL that corresponds to your client's IP addresses. For example:

```
acl myclients src 172.16.5.0/24
```

Next, allow those clients in the *http_access* list:

```
http_access allow myclients
```

10.3 how do I congure Squid not to cache a specic server?

```
acl someserver dstdomain .someserver.com
no_cache deny someserver
```

10.4 How do I implement an ACL ban list?

As an example, we will assume that you would like to prevent users from accessing cooking recipes.

One way to implement this would be to deny access to any URLs that contain the words "cooking" or "recipe." You would use these conguration lines:

```
acl Cooking1 url_regex cooking
acl Recipe1 url_regex recipe
http_access deny Cooking1
http_access deny Recipe1
http_access allow all
```

The *url_regex* means to search the entire URL for the regular expression you specify. Note that these regular expressions are case-sensitive, so a url containing "Cooking" would not be denied.

Another way is to deny access to specic servers which are known to hold recipes. For example:

```
acl Cooking2 dstdomain www.gourmet-chef.com
http_access deny Cooking2
http_access allow all
```

The *dstdomain* means to search the hostname in the URL for the string "www.gourmet-chef.com." Note that when IP addresses are used in URLs (instead of domain names), Squid-1.1 implements relaxed access controls. If the a domain name for the IP address has been saved in Squid's "FQDN cache," then Squid can compare the destination domain against the access controls. However, if the domain is not immediately available, Squid allows the request and makes a lookup for the IP address so that it may be available for future reqeusts.

10.5 How do I block specic users or groups from accessing my cache?

10.5.1 Ident

You can use *ident lookups* `<ftp://ftp.isi.edu/in-notes/rfc931.txt>` to allow specic users access to your cache. This requires that an *ident server* `<ftp://ftp.lysator.liu.se/pub/ident/servers>` process runs on the user's machine(s). In your *squid.conf* conguration le you would write something like this:

```
ident_lookup on
acl friends user kim lisa frank joe
http_access allow friends
http_access deny all
```

10.5.2 Proxy Authentication

Another option is to use proxy-authentication. In this scheme, you assign usernames and passwords to individuals. When they rst use the proxy they are asked to authenticate themselves by entering their username and password.

In Squid v2 this authentication is hanled via external processes. For information on how to congure this, please see 19.6.

10.6 Do you have a CGI program which lets users change their own proxy passwords?

Pedro L Orso <mailto:orso@brturbo.com> has adapted the Apache's *htpasswd* into a CGI program called *chpasswd.cgi* </htpasswd/chpasswd-cgi.tar.gz>.

10.7 Is there a way to do ident lookups only for a certain host and compare the result with a userlist in squid.conf?

Sort of.

If you use a *user* ACL in squid conf, then Squid will perform an *ident lookup* <ftp://ftp.isi.edu/in-notes/rfc931.txt> for every client request. In other words, Squid-1.1 will perform ident lookups for all requests or no requests. Dening a *user* ACL enables ident lookups, regardless of the *ident_lookup* setting.

However, even though ident lookups are performed for every request, Squid does not wait for the lookup to complete unless the ACL rules require it. Consider this conguration:

```
acl host1 src 10.0.0.1
acl host2 src 10.0.0.2
acl pals  user kim lisa frank joe
http_access allow host1
http_access allow host2 pals
```

Requests coming from 10.0.0.1 will be allowed immediately because there are no user requirements for that host. However, requests from 10.0.0.2 will be allowed only after the ident lookup completes, and if the username is in the set kim, lisa, frank, or joe.

10.8 Common Mistakes

10.8.1 And/Or logic

You've probably noticed (and been frustrated by) the fact that you cannot combine access controls with terms like "and" or "or." These operations are already built in to the access control scheme in a fundamental way which you must understand.

All elements of an *acl* entry are OR'ed together.

All elements of an *access* entry are AND'ed together. e.g. *http_access* and *icp_access*.

For example, the following access control conguration will never work:

```
acl ME src 10.0.0.1
acl YOU src 10.0.0.2
http_access allow ME YOU
```

In order for the request to be allowed, it must match the "ME" acl AND the "YOU" acl. This is impossible because any IP address could only match one or the other. This should instead be rewritten as:

```
acl ME src 10.0.0.1
acl YOU src 10.0.0.2
```

```
http_access allow ME
http_access allow YOU
```

Or, alternatively, this would also work:

```
acl US src 10.0.0.1 10.0.0.2
http_access allow US
```

10.8.2 allow/deny mixups

I have read through my squid.conf numerous times, spoken to my neighbors, read the FAQ and Squid Docs and cannot for the life of me work out why the following will not work.

I can successfully access cachemgr.cgi from our web server machine here, but I would like to use MRTG to monitor various aspects of our proxy. When I try to use 'client' or GET cache_object from the machine the proxy is running on, I always get access denied.

```
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl server    src 1.2.3.4/255.255.255.255
acl all src 0.0.0.0/0.0.0.0
acl ourhosts src 1.2.0.0/255.255.0.0

http_access deny manager !localhost !server
http_access allow ourhosts
http_access deny all
```

The intent here is to allow cache manager requests from the *localhost* and *server* addresses, and deny all others. This policy has been expressed here:

```
http_access deny manager !localhost !server
```

The problem here is that for allowable requests, this access rule is not matched. For example, if the source IP address is *localhost*, then "!localhost" is *false* and the access rule is not matched, so Squid continues checking the other rules. Cache manager requests from the *server* address work because *server* is a subset of *ourhosts* and the second access rule will match and allow the request. Also note that this means any cache manager request from *ourhosts* would be allowed.

To implement the desired policy correctly, the access rules should be rewritten as

```
http_access allow manager localhost
http_access allow manager server
http_access deny manager
http_access allow ourhosts
http_access deny all
```

If you're using *miss_access*, then don't forget to also add a *miss_access* rule for the cache manager:

```
miss_access allow manager
```

You may be concerned that the having ve access rules instead of three may have an impact on the cache performance. In our experience this is not the case. Squid is able to handle a moderate amount of access control checking without degrading overall performance. You may like to verify that for yourself, however.

10.8.3 Dierences between *src* and *srcdomain* ACL types.

For the *srcdomain* ACL type, Squid does a reverse lookup of the client's IP address and checks the result with the domains given on the *acl* line. With the *src* ACL type, Squid converts hostnames to IP addresses at startup and then only compares the client's IP address. The *src* ACL is preferred over *srcdomain* because it does not require address-to-name lookups for each request.

10.9 I set up my access controls, but they don't work! why?

If ACLs are giving you problems and you don't know why they aren't working, you can use this tip to debug them.

In *squid.conf* enable debugging for section 33 at level 2. For example:

```
debug_options ALL,1 33,2
```

Then restart or recongure squid.

From now on, your *cache.log* should contain a line for every request that explains if it was allowed, or denied, and which ACL was the last one that it matched.

If this does not give you sucient information to nail down the problem you can also enable detailed debug information on ACL processing

```
debug_options ALL,1 33,2 28,9
```

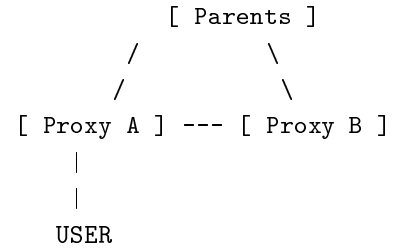
Then restart or recongure squid as above.

From now on, your *cache.log* should contain detailed traces of all access list processing. Be warned that this can be quite some lines per request.

See also 11.20

10.10 Proxy-authentication and neighbor caches

The problem...



Proxy A sends and ICP query to Proxy B about an object, Proxy B replies with an ICP_HIT. Proxy A forwards the HTTP request to Proxy B, but does not pass on the authentication details, therefore the HTTP GET from Proxy A fails.

Only ONE proxy cache in a chain is allowed to "use" the Proxy-Authentication request header. Once the header is used, it must not be passed on to other proxies.

Therefore, you must allow the neighbor caches to request from each other without proxy authentication. This is simply accomplished by listing the neighbor ACL's rst in the list of *http_access* lines. For example:

```
acl proxy-A src 10.0.0.1
acl proxy-B src 10.0.0.2
acl user_passwords proxy_auth /tmp/user_passwds

http_access allow proxy-A
http_access allow proxy-B
http_access allow user_passwords
http_access deny all
```

10.11 Is there an easy way of banning all Destination addresses except one?

```
acl GOOD dst 10.0.0.1
acl BAD dst 0.0.0.0/0.0.0.0
http_access allow GOOD
http_access deny BAD
```

10.12 Does anyone have a ban list of porn sites and such?

Jasons Staudenmayer <http://members.lycos.co.uk/njadmin>

Pedro Lineu Orso's List <http://web.onda.com.br/orso/>

Linux Center Hong Kong's List <http://www.hklc.com/squidblock/>

Snerpa, an ISP in Iceland operates a DNS-database of IP-addresses of blacklisted sites containing porn, violence, etc. which is utilized using a small perl-script redirector. Information on this on the *INlter* <http://www.snerpa.is/notendur/infilter/infilter-en.phtml> webpage.

The *SquidGuard* <http://www.squidguard.org/blacklist/> redirector folks provide a blacklist.

Bill Stearns maintains the *sa-blacklist* <http://www.stearns.org/sa-blacklist/> of known spammers. By blocking the spammer web sites in squid, users can no longer use up bandwidth downloading spam images and html. Even more importantly, they can no longer send out requests for things like scripts and gifs that have a unique identifer attached, showing that they opened the email and making their addresses more valuable to the spammer.

10.13 Squid doesn't match my subdomains

NOTE: Current Squid versions (as of Squid-2.4) will warn you when this kind of conguration is used. Also the conguration here uses the dstdomain syntax of Squid-2.1 or earlier.. (2.2 and later needs to have domains prexed by a dot)

There is a subtle problem with domain-name based access controls when a single ACL element has an entry that is a subdomain of another entry. For example, consider this list:

```
acl FOO dstdomain boulder.co.us vail.co.us co.us
```

In the rst place, the above list is simply wrong because the rst two (*boulder.co.us* and *vail.co.us*) are unnecessary. Any domain name that matches one of the rst two will also match the last one (*co.us*). Ok, but why does this happen?

The problem stems from the data structure used to index domain names in an access control list. Squid uses *Splay trees* for lists of domain names. As other tree-based data structures, the searching algorithm requires

a comparison function that returns -1, 0, or +1 for any pair of keys (domain names). This is similar to the way that *strcmp()* works.

The problem is that it is wrong to say that *co.us* is greater-than, equal-to, or less-than *boulder.co.us*.

For example, if you said that *co.us* is LESS than *f.co.us* , then the Splay tree searching algorithm might never discover *co.us* as a match for *kkk.co.us*.

similarly, if you said that *co.us* is GREATER than *f.co.us* , then the Splay tree searching algorithm might never discover *co.us* as a match for *bbb.co.us*.

The bottom line is that you can't have one entry that is a subdomain of another. Squid-2.2 will warn you if it detects this condition.

10.14 Why does Squid deny some port numbers?

It is dangerous to allow Squid to connect to certain port numbers. For example, it has been demonstrated that someone can use Squid as an SMTP (email) relay. As I'm sure you know, SMTP relays are one of the ways that spammers are able to ood our mailboxes. To prevent mail relaying, Squid denies requests when the URL port number is 25. Other ports should be blocked as well, as a precaution.

There are two ways to lter by port number: either allow specic ports, or deny specic ports. By default, Squid does the rst. This is the ACL entry that comes in the default *squid.conf*:

```
acl Safe_ports port 80 21 443 563 70 210 1025-65535
http_access deny !Safe_ports
```

The above conguration denies requests when the URL port number is not in the list. The list allows connections to the standard ports for HTTP, FTP, Gopher, SSL, WAIS, and all non-priveleged ports.

Another approach is to deny dangerous ports. The dangerous port list should look something like:

```
acl Dangerous_ports 7 9 19 22 23 25 53 109 110 119
http_access deny Dangerous_ports
```

...and probably many others.

Please consult the */etc/services* le on your system for a list of known ports and protocols.

10.15 Does Squid support the use of a database such as mySQL for storing the ACL list?

Note: The information here is current for version 2.2.

No, it does not.

10.16 How can I allow a single address to access a specic URL?

This example allows only the *special_client* to access the *special_url*. Any other client that tries to access the *special_url* is denied.

```
acl special_client src 10.1.2.3
acl special_url url_regex ^http://www.squid-cache.org/Doc/FAQ/$
http_access allow special_client special_url
http_access deny special_url
```

10.17 How can I allow some clients to use the cache at specic times?

Let's say you have two workstations that should only be allowed access to the Internet during working hours (8:30 - 17:30). You can use something like this:

```
acl FOO src 10.1.2.3 10.1.2.4
acl WORKING time MTWHF 08:30-17:30
http_access allow FOO WORKING
http_access deny FOO
```

10.18 How can I allow some users to use the cache at specic times?

```
acl USER1 proxy_auth Dick
acl USER2 proxy_auth Jane
acl DAY time 06:00-18:00
http_access allow USER1 DAY
http_access deny USER1
http_access allow USER2 !DAY
http_access deny USER2
```

10.19 Problems with IP ACL's that have complicated netmasks

Note: The information here is current for version 2.3.

The following ACL entry gives inconsistent or unexpected results:

```
acl restricted  src 10.0.0.128/255.0.0.128 10.85.0.0/16
```

The reason is that IP access lists are stored in "splay" tree data structures. These trees require the keys to be sortable. When you use a complicated, or non-standard, netmask (255.0.0.128), it confuses the function that compares two address/mask pairs.

The best way to x this problem is to use separate ACL names for each ACL value. For example, change the above to:

```
acl restricted1 src 10.0.0.128/255.0.0.128
acl restricted2 src 10.85.0.0/16
```

Then, of course, you'll have to rewrite your *http_access* lines as well.

10.20 Can I set up ACL's based on MAC address rather than IP?

Yes, for some operating systes. Squid calls these "ARP ACLs" and they are supported on Linux, Solaris, and probably BSD variants.

NOTE: Squid can only determine the MAC address for clients that are on the same subnet. If the client is on a dierent subnet, then Squid can not nd out its MAC address.

To use ARP (MAC) access controls, you rst need to compile in the optional code. Do this with the *–enable-arp-acl* congure option:

```
% ./configure --enable-arp-acl ...
% make clean
% make
```

If *src/acl.c* doesn't compile, then ARP ACLs are probably not supported on your system.

If everything compiles, then you can add some ARP ACL lines to your *squid.conf*:

```
acl M1 arp 01:02:03:04:05:06
acl M2 arp 11:12:13:14:15:16
http_access allow M1
http_access allow M2
http_access deny all
```

10.21 Debugging ACLs

See 10.9 and 11.20.

10.22 Can I limit the number of connections from a client?

Yes, use the *maxconn* ACL type in conjunction with *http_access deny*. For example:

```
acl losers src 1.2.3.0/24
acl 5CONN maxconn 5
http_access deny 5CONN losers
```

Given the above conguration, when a client whose source IP address is in the 1.2.3.0/24 subnet tries to establish 6 or more connections at once, Squid returns an error page. Unless you use the *deny_info* feature, the error message will just say "access denied."

The *maxconn* ACL requires the client_db feature. If you've disabled client_db (for example with *client_db o*) then *maxconn* ALCs will not work.

Note, the *maxconn* ACL type is kind of tricky because it uses less-than comparison. The ACL is a match when the number of established connections is *greater* than the value you specify. Because of that, you don't want to use the *maxconn* ACL with *http_access allow*.

Also note that you could use *maxconn* in conjunction with a user type (ident, proxy_auth), rather than an IP address type.

10.23 I'm trying to deny *foo.com*, but it's not working.

In Squid-2.3 we changed the way that Squid matches subdomains. There is a dierence between *.foo.com* and *foo.com*. The rst matches any domain in *foo.com*, while the latter matches only "foo.com" exactly. So if you want to deny *bar.foo.com*, you should write

```
acl yuck dstdomain .foo.com
http_access deny yuck
```

10.24 I want to customize, or make my own error messages.

You can customize the existing error messages as described in 19.10. You can also create new error messages and use these in conjunction with the *deny_info* option.

For example, lets say you want your users to see a special message when they request something that matches your pornography list. First, create a le named ERR _NO_PORNO in the */usr/local/squid/etc/errors* directory. That le might contain something like this:

```
<p>
Our company policy is to deny requests to known porno sites.  If you
feel you've received this message in error, please contact
the support staff (support@this.company.com, 555-1234).
```

Next, set up your access controls as follows:

```
acl porn url_regex "/usr/local/squid/etc/porno.txt"
deny_info ERR_NO_PORNO porn
http_access deny porn
(additional http_access lines ...)
```

10.25 I want to use local time zone in error messages

Squid by defaults uses GMT as timestamp in all geenrated error messages. This to allow the cache to participate in a hierarchy of caches in dierent timezones without risking confusion about what the time is.

To change the timestamp in Squid generated error messages you must change the Squid signature. See 19.10. The signature by defaults uses %T as timestamp, but if you like then you can use %t instead for a timestamp using local time zone.

11 Troubleshooting

11.1 Why am I getting "Proxy Access Denied?"

You may need to set up the *http_access* option to allow requests from your IP addresses. Please see 10 for information about that.

If *squid* is in httpd-accelerator mode, it will accept normal HTTP requests and forward them to a HTTP server, but it will not honor proxy requests. If you want your cache to also accept proxy-HTTP requests then you must enable this feature:

```
httpd_accel_with_proxy on
```

Alternately, you may have miscongured one of your ACLs. Check the *access.log* and *squid.conf* les for clues.

11.2 I can't get `local_domain` to work; *Squid* is caching the objects from my local servers.

The `local_domain` directive does not prevent local objects from being cached. It prevents the use of sibling caches when fetching local objects. If you want to prevent objects from being cached, use the `cache_stoplist` or `http_stop` conguration options (depending on your version).

11.3 I get `Connection Refused` when the cache tries to retrieve an object located on a sibling, even though the sibling thinks it delivered the object to my cache.

If the HTTP port number is wrong but the ICP port is correct you will send ICP queries correctly and the ICP replies will fool your cache into thinking the conguration is correct but large objects will fail since you don't have the correct HTTP port for the sibling in your *squid.conf* le. If your sibling changed their `http_port`, you could have this problem for some time before noticing.

11.4 Running out of ledescriptors

If you see the `Too many open files` error message, you are most likely running out of le descriptors. This may be due to running Squid on an operating system with a low ledescriptor limit. This limit is often congurable in the kernel or with other system tuning tools. There are two ways to run out of le descriptors: rst, you can hit the per-process limit on le descriptors. Second, you can hit the system limit on total le descriptors for all processes.

11.4.1 Linux

Henrik has a *How to get many ledescriptors on Linux 2.2.X* `<http://squid.sourceforge.net/hno/linux-lfd.html>` page.

You also might want to have a look at *lehandle patch* `<http://www.linux.org.za/oskar/patches/kernel/filehandle/>` by *Michael O'Reilly* `<mailto:michael@metal.iinet.net.au>`

If your kernel version is 2.2.x or greater, you can read and write the maximum number of le handles and/or inodes simply by accessing the special les:

```
/proc/sys/fs/file-max
/proc/sys/fs/inode-max
```

So, to increase your le descriptor limit:

```
echo 3072 > /proc/sys/fs/file-max
```

If your kernel version is between 2.0.35 and 2.1.x (?), you can read and write the maximum number of le handles and/or inodes simply by accessing the special les:

```
/proc/sys/kernel/file-max
/proc/sys/kernel/inode-max
```

While this does increase the current number of le descriptors, Squid's *congure* script probably won't gure out the new value unless you also update the include les, specically the value of *OPEN_MAX* in */usr/include/linux/limits.h*.

11.4.2 Solaris

Add the following to your */etc/system* le to increase your maximum le descriptors per process:

```
set rlim_fd_max = 4096
```

Next you should re-run the *congure* script in the top directory so that it nds the new value. If it does not nd the new limit, then you might try editing *include/autoconf.h* and setting `#define DEFAULT_FD_SETSIZE` by hand. Note that *include/autoconf.h* is created from *autoconf.h.in* every time you run congure. Thus, if you edit it by hand, you might lose your changes later on.

If you have a very old version of Squid (1.1.X), and you want to use more than 1024 descriptors, then you must edit *src/Makele* and enable `$(USE_POLL_OPT)`. Then recompile *squid*.

Jens-S. Voeckler `<mailto:voeckler at rvs dot uni-hannover dot de>` advises that you should NOT change the default soft limit (*rlim_fd_cur*) to anything larger than 256. It will break other programs, such as the license manager needed for the SUN workshop compiler. Jens-S. also says that it should be safe to raise the limit for the Squid process as high as 16,384 except that there may be problems duruing recongure or logrotate if all of the lower 256 ledescriptors are in use at the time or rotate/recongure.

11.4.3 FreeBSD

by *Torsten Sturm* `<mailto:torsten.sturm@axis.de>`

1. How do I check my maximum ledescriptors? Do `sysctl -a` and look for the value of `kern.maxfilesperproc`.
2. How do I increase them?

   ```
   sysctl -w kern.maxfiles=XXXX
   sysctl -w kern.maxfilesperproc=XXXX
   ```

   Warning: You probably want `maxfiles` > `maxfilesperproc` if you're going to be pushing the limit.
3. What is the upper limit? I don't think there is a formal upper limit inside the kernel. All the data structures are dynamically allocated. In practice there might be unintended metaphenomena (kernel spending too much time searching tables, for example).

11.4.4 General BSD

For most BSD-derived systems (SunOS, 4.4BSD, OpenBSD, FreeBSD, NetBSD, BSD/OS, 386BSD, Ultrix) you can also use the "brute force" method to increase these values in the kernel (requires a kernel rebuild):

1. How do I check my maximum ledescriptors? Do `pstat -T` and look for the `files` value, typically expressed as the ratio of `current`maximum/.
2. How do I increase them the easy way? One way is to increase the value of the `maxusers` variable in the kernel conguration le and build a new kernel. This method is quick and easy but also has the eect of increasing a wide variety of other variables that you may not need or want increased.
3. Is there a more precise method? Another way is to nd the *param.c* le in your kernel build area and change the arithmetic behind the relationship between `maxusers` and the maximum number of open les.

Here are a few examples which should lead you in the right direction:

1. SunOS Change the value of `nfile` in usr/kvm/sys/conf.common/param.c/tt> by altering this equation:

   ```
   int     nfile = 16 * (NPROC + 16 + MAXUSERS) / 10 + 64;
   ```

Where NPROC is dened by:

```
#define NPROC (10 + 16 * MAXUSERS)
```

2. FreeBSD (from the 2.1.6 kernel) Very similar to SunOS, edit */usr/src/sys/conf/param.c* and alter the relationship between maxusers and the maxfiles and maxfilesperproc variables:

```
int     maxfiles = NPROC*2;
int     maxfilesperproc = NPROC*2;
```

Where NPROC is dened by: #define NPROC (20 + 16 * MAXUSERS) The per-process limit can also be adjusted directly in the kernel conguration le with the following directive: options OPEN_MAX=128

3. BSD/OS (from the 2.1 kernel) Edit /usr/src/sys/conf/param.c and adjust the maxfiles math here:

```
int     maxfiles = 3 * (NPROC + MAXUSERS) + 80;
```

Where NPROC is dened by: #define NPROC (20 + 16 * MAXUSERS) You should also set the OPEN_MAX value in your kernel conguration le to change the per-process limit.

11.4.5 Recongure afterwards

NOTE: After you rebuild/recongure your kernel with more ledescriptors, you must then recompile Squid. Squid's congure script determines how many ledescriptors are available, so you must make sure the congure script runs again as well. For example:

```
cd squid-1.1.x
    make realclean
    ./configure --prefix=/usr/local/squid
    make
```

11.5 What are these strange lines about removing objects?

For example:

```
97/01/23 22:31:10| Removed 1 of 9 objects from bucket 3913
97/01/23 22:33:10| Removed 1 of 5 objects from bucket 4315
97/01/23 22:35:40| Removed 1 of 14 objects from bucket 6391
```

These log entries are normal, and do not indicate that *squid* has reached cache_swap_high.

Consult your cache information page in *cachemgr.cgi* for a line like this:

```
Storage LRU Expiration Age:     364.01 days
```

Objects which have not been used for that amount of time are removed as a part of the regular maintenance. You can set an upper limit on the LRU Expiration Age value with reference_age in the cong le.

11.6 Can I change a Windows NT FTP server to list directories in Unix format?

Why, yes you can! Select the following menus:

Start

Programs

Microsoft Internet Server (Common)

Internet Service Manager

This will bring up a box with icons for your various services. One of them should be a little ftp "folder." Double click on this.

You will then have to select the server (there should only be one) Select that and then choose "Properties" from the menu and choose the "directories" tab along the top.

There will be an option at the bottom saying "Directory listing style." Choose the "Unix" type, not the "MS-DOS" type.

–Oskar Pearson <oskar@is.co.za>

11.7 Why am I getting "Ignoring MISS from non-peer x.x.x.x?"

You are receiving ICP MISSes (via UDP) from a parent or sibling cache whose IP address your cache does not know about. This may happen in two situations.

1. If the peer is multihomed, it is sending packets out an interface which is not advertised in the DNS. Unfortunately, this is a conguration problem at the peer site. You can tell them to either add the IP address interface to their DNS, or use Squid's "udp_outgoing_address" option to force the replies out a specic interface. For example: *on your parent squid.conf:*

   ```
   udp_outgoing_address proxy.parent.com
   ```

   on your squid.conf:

   ```
   cache_host proxy.parent.com parent 3128 3130
   ```

2. You can also see this warning when sending ICP queries to multicast addresses. For security reasons, Squid requires your conguration to list all other caches listening on the multicast group address. If an unknown cache listens to that address and sends replies, your cache will log the warning message. To x this situation, either tell the unknown cache to stop listening on the multicast address, or if they are legitimate, add them to your conguration le.

11.8 DNS lookups for domain names with underscores (_) always fail.

The standards for naming hosts (*RFC 952* `<ftp://ftp.isi.edu/in-notes/rfc952.txt>`, *RFC 1101* `<ftp://ftp.isi.edu/in-notes/rfc1101.txt>`) do not allow underscores in domain names:

> A "name" (Net, Host, Gateway, or Domain name) is a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9), minus sign (-), and period (.).

The resolver library that ships with recent versions of BIND enforces this restriction, returning an error for any host with underscore in the hostname. The best solution is to complain to the hostmaster of the oending site, and ask them to rename their host.

See also the *comp.protocols.tcp-ip.domains FAQ* <http://www.intac.com/~cdp/cptd-faq/section4.html#underscore>.

Some people have noticed that *RFC 1033* <ftp://ftp.isi.edu/in-notes/rfc1033.txt> implies that underscores **are** allowed. However, this is an *informational* RFC with a poorly chosen example, and not a *standard* by any means.

11.9 Why does Squid say: "Illegal character in hostname; underscores are not allowed?'

See the above question. The underscore character is not valid for hostnames.

Some DNS resolvers allow the underscore, so yes, the hostname might work ne when you don't use Squid.

To make Squid allow underscores in hostnames, re-run the *congure* script with this option:

```
% ./configure --enable-underscores ...
```

and then recompile:

```
% make clean
% make
```

11.10 Why am I getting access denied from a sibling cache?

The answer to this is somewhat complicated, so please hold on. *NOTE:* most of this text is taken from *ICP and the Squid Web Cache* <http://www.life-gone-hazy.com/writings/icp-squid.ps.gz>.

An ICP query does not include any parent or sibling designation, so the receiver really has no indication of how the peer cache is congured to use it. This issue becomes important when a cache is willing to serve cache hits to anyone, but only handle cache misses for its paying users or customers. In other words, whether or not to allow the request depends on if the result is a hit or a miss. To accomplish this, Squid acquired the `miss_access` feature in October of 1996.

The necessity of "miss access" makes life a little bit complicated, and not only because it was awkward to implement. Miss access means that the ICP query reply must be an extremely accurate prediction of the result of a subsequent HTTP request. Ascertaining this result is actually very hard, if not impossible to do, since the ICP request cannot convey the full HTTP request. Additionally, there are more types of HTTP request results than there are for ICP. The ICP query reply will either be a hit or miss. However, the HTTP request might result in a "`304 Not Modified`" reply sent from the origin server. Such a reply is not strictly a hit since the peer needed to forward a conditional request to the source. At the same time, its not strictly a miss either since the local object data is still valid, and the Not-Modied reply is quite small.

One serious problem for cache hierarchies is mismatched freshness parameters. Consider a cache C using "strict" freshness parameters so its users get maximally current data. C has a sibling S with less strict freshness parameters. When an object is requested at C, C might nd that S already has the object via an ICP query and ICP HIT response. C then retrieves the object from S.

In an HTTP/1.0 world, C (and C's client) will receive an object that was never subject to its local freshness rules. Neither HTTP/1.0 nor ICP provides any way to ask only for objects less than a certain age. If the retrieved object is stale by Cs rules, it will be removed from Cs cache, but it will subsequently be fetched

from S so long as it remains fresh there. This conguration miscoupling problem is a signicant deterrent to establishing both parent and sibling relationships.

HTTP/1.1 provides numerous request headers to specify freshness requirements, which actually introduces a dierent problem for cache hierarchies: ICP still does not include any age information, neither in query nor reply. So S may return an ICP HIT if its copy of the object is fresh by its conguration parameters, but the subsequent HTTP request may result in a cache miss due to any `Cache-control:` headers originated by C or by C's client. Situations now emerge where the ICP reply no longer matches the HTTP request result.

In the end, the fundamental problem is that the ICP query does not provide enough information to accurately predict whether the HTTP request will be a hit or miss. In fact, the current ICP Internet Draft is very vague on this subject. What does ICP HIT really mean? Does it mean "I know a little about that URL and have some copy of the object?" Or does it mean "I have a valid copy of that object and you are allowed to get it from me?"

So, what can be done about this problem? We really need to change ICP so that freshness parameters are included. Until that happens, the members of a cache hierarchy have only two options to totally eliminate the "access denied" messages from sibling caches:

1. Make sure all members have the same `refresh_rules` parameters.
2. Do not use `miss_access` at all. Promise your sibling cache administrator that *your* cache is properly congured and that you will not abuse their generosity. The sibling cache administrator can check his log les to make sure you are keeping your word.

If neither of these is realistic, then the sibling relationship should not exist.

11.11 Cannot bind socket FD NN to *:8080 (125) Address already in use

This means that another processes is already listening on port 8080 (or whatever you're using). It could mean that you have a Squid process already running, or it could be from another program. To verify, use the *netstat* command:

```
netstat -naf inet | grep LISTEN
```

That will show all sockets in the LISTEN state. You might also try

```
netstat -naf inet | grep 8080
```

If you nd that some process has bound to your port, but you're not sure which process it is, you might be able to use the excellent *lsof* `<ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/>` program. It will show you which processes own every open le descriptor on your system.

11.12 icpDetectClientClose: ERROR xxx.xxx.xxx.xxx: (32) Broken pipe

This means that the client socket was closed by the client before Squid was nished sending data to it. Squid detects this by trying to `read(2)` some data from the socket. If the `read(2)` call fails, then Squid konws the socket has been closed. Normally the `read(2)` call returns *ECONNRESET: Connection reset by peer* and these are NOT logged. Any other error messages (such as *EPIPE: Broken pipe* are logged to *cache.log*. See the "intro" of section 2 of your Unix manual for a list of all error codes.

11.13 icpDetectClientClose: FD 135, 255 unexpected bytes

These are caused by misbehaving Web clients attempting to use persistent connections. Squid-1.1 does not support persistent connections.

11.14 Does Squid work with NTLM Authentication?

Version 2.5 </Versions/v2/2.5/> will support Microsoft NTLM authentication. However, there are some limits on our support: We cannot proxy connections to a origin server that use NTLM authentication, but we can act as a web accelerator or proxy server and authenticate the client connection using NTLM.

We support NT4, Samba, and Windows 2000 Domain Controllers. For more information get squid 2.5 and run *./congure –help* .

Why we cannot proxy NTLM even though we can use it. Quoting from summary at the end of the browser authentication section in *this article* <http://support.microsoft.com/support/kb/articles/Q198/1/16.ASP>:

> In summary, Basic authentication does not require an implicit end-to-end state, and can therefore be used through a proxy server. Windows NT Challenge/Response authentication requires implicit end-to-end state and will not work through a proxy server.

Squid transparently passes the NTLM request and response headers between clients and servers. NTLM relies on a single end-end connection (possibly with men-in-the-middle, but a single connection every step of the way. This implies that for NTLM authentication to work at all with proxy caches, the proxy would need to tightly link the client-proxy and proxy-server links, as well as understand the state of the link at any one time. NTLM through a CONNECT might work, but we as far as we know that hasn't been implemented by anyone, and it would prevent the pages being cached - removing the value of the proxy.

NTLM authentication is carried entirely inside the HTTP protocol, but is dierent from Basic authentication in many ways.

1. It is dependent on a stateful end-to-end connection which collides with RFC 2616 for proxy-servers to disjoin the client-proxy and proxy-server connections.
2. It is only taking place once per connection, not per request. Once the connection is authenticated then all future requests on the same connection inherities the authentication. The connection must be reestablished to set up other authentication or re-identify the user.

The reasons why it is not implemented in Netscape is probably:

> It is very specic for the Windows platform
>
> It is not dened in any RFC or even internet draft.
>
> The protocol has several shortcomings, where the most apparent one is that it cannot be proxied.
>
> There exists an open internet standard which does mostly the same but without the shortcomings or platform dependencies: *digest authentication* <ftp://ftp.isi.edu/in-notes/rfc2617.txt>.

11.15 The *default* parent option isn't working!

This message was received at *squid-bugs*:

If you have only one parent, congured as:

```
cache_host xxxx parent 3128 3130 no-query default
```

nothing is sent to the parent; neither UDP packets, nor TCP connections.

Simply adding *default* to a parent does not force all requests to be sent to that parent. The term *default* is perhaps a poor choice of words. A *default* parent is only used as a **last resort**. If the cache is able to make direct connections, direct will be preferred over default. If you want to force all requests to your parent cache(s), use the *never_direct* option:

```
acl all src 0.0.0.0/0.0.0.0
never_direct allow all
```

11.16 "Hot Mail" complains about: Intrusion Logged. Access denied.

"Hot Mail" is proxy-unfriendly and requires all requests to come from the same IP address. You can x this by adding to your *squid.conf*:

```
hierarchy_stoplist hotmail.com
```

11.17 My Squid becomes very slow after it has been running for some time.

This is most likely because Squid is using more memory than it should be for your system. When the Squid process becomes large, it experiences a lot of paging. This will very rapidly degrade the performance of Squid. Memory usage is a complicated problem. There are a number of things to consider.

Then, examine the Cache Manager *Info* ouput and look at these two lines:

```
Number of HTTP requests received:  121104
Page faults with physical i/o:      16720
```

Note, if your system does not have the *getrusage()* function, then you will not see the page faults line.

Divide the number of page faults by the number of connections. In this case 16720/121104 = 0.14. Ideally this ratio should be in the 0.0 - 0.1 range. It may be acceptable to be in the 0.1 - 0.2 range. Above that, however, and you will most likely nd that Squid's performance is unacceptably slow.

If the ratio is too high, you will need to make some changes to 8.9.

See also 8.11.

11.18 WARNING: Failed to start 'dnsserver'

This could be a permission problem. Does the Squid userid have permission to execute the *dnsserver* program?

You might also try testing *dnsserver* from the command line:

```
> echo oceana.nlanr.net | ./dnsserver
```

Should produce something like:

```
$name oceana.nlanr.net
$h_name oceana.nlanr.net
$h_len 4
$ipcount 1
132.249.40.200
$aliascount 0
$ttl 82067
$end
```

11.19 Sending in Squid bug reports

Bug reports for Squid should be registered in our *bug database* <http://www.squid-cache.org/bugs/>. Any bug report must include

The Squid version

Your Operating System type and version

A clear description of the bug symptoms.

If your Squid crashes the report must include a 11.19.1 as described below

Please note that bug reports are only processed if they can be reproduced or identied in the current STABLE or development versions of Squid. If you are running an older version of Squid the rst response will be to ask you to upgrade unless the developer who looks at your bug report immediately can identify that the bug also exists in the current versions. It should also be noted that any patches provided by the Squid developer team will be to the current STABLE version even if you run an older version.

11.19.1 crashes and core dumps

There are two conditions under which squid will exit abnormally and generate a coredump. First, a SIGSEGV or SIGBUS signal will cause Squid to exit and dump core. Second, many functions include consistency checks. If one of those checks fail, Squid calls abort() to generate a core dump.

Many people report that Squid doesn't leave a coredump anywhere. This may be due to one of the following reasons:

Resource Limits. The shell has limits on the size of a coredump le. You may need to increase the limit.

sysctl options. On FreeBSD, you won't get a coredump from programs that call setuid() and/or setgid() (like Squid sometimes does) unless you enable this option:

```
# sysctl -w kern.sugid_coredump=1
```

No debugging symbols. The Squid binary must have debugging symbols in order to get a meaningful coredump.

Threads and Linux. On Linux, threaded applications do not generate core dumps. When you use the aufs cache_dir type, it uses threads and you can't get a coredump.

It did leave a coredump le, you just can't nd it.

Resource Limits: These limits can usually be changed in shell scripts. The command to change the resource limits is usually either *limit* or *limits*. Sometimes it is a shell-builtin function, and sometimes it is a regular program. Also note that you can set resource limits in the */etc/login.conf* le on FreeBSD and maybe other BSD systems.

To change the coredumpsize limit you might use a command like:

```
limit coredumpsize unlimited
```

or

```
limits coredump unlimited
```

Debugging Symbols: To see if your Squid binary has debugging symbols, use this command:

```
% nm /usr/local/squid/bin/squid | head
```

The binary has debugging symbols if you see gobbledegook like this:

```
0812abec B AS_tree_head
080a7540 D AclMatchedName
080a73fc D ActionTable
080908a4 r B_BYTES_STR
080908bc r B_GBYTES_STR
080908ac r B_KBYTES_STR
080908b4 r B_MBYTES_STR
080a7550 D Biggest_FD
08097c0c R CacheDigestHashFuncCount
08098f00 r CcAttrs
```

There are no debugging symbols if you see this instead:

```
/usr/local/squid/bin/squid: no symbols
```

Debugging symbols may have been removed by your *install* program. If you look at the squid binary from the source directory, then it might have the debugging symbols.

Coredump Location: The core dump le will be left in one of the following locations:

1. The *coredump_dir* directory, if you set that option.
2. The rst *cache_dir* directory if you have used the *cache_eective _user* option.
3. The current directory when Squid was started

Recent versions of Squid report their current directory after starting, so look there rst:

```
2000/03/14 00:12:36| Set Current Directory to /usr/local/squid/cache
```

If you cannot nd a core le, then either Squid does not have permission to write in its current directory, or perhaps your shell limits are preventing the core le from being written.

Often you can get a coredump if you run Squid from the command line like this (csh shells and clones):

```
% limit core un
% /usr/local/squid/bin/squid -NCd1
```

Once you have located the core dump le, use a debugger such as *dbx* or *gdb* to generate a stack trace:

```
tirana-wessels squid/src 270% gdb squid /T2/Cache/core
GDB is free software and you are welcome to distribute copies of it
 under certain conditions; type "show copying" to see the conditions.
There is absolutely no warranty for GDB; type "show warranty" for details.
GDB 4.15.1 (hppa1.0-hp-hpux10.10), Copyright 1995 Free Software Foundation, Inc...
Core was generated by `squid'.
Program terminated with signal 6, Aborted.

[...]

(gdb) where
#0  0xc01277a8 in _kill ()
#1  0xc00b2944 in _raise ()
#2  0xc007bb08 in abort ()
#3  0x53f5c in __eprintf (string=0x7b037048 "", expression=0x5f <Address 0x5f out of bounds>, line=8, f
#4  0x29828 in fd_open (fd=10918, type=3221514150, desc=0x95e4 "HTTP Request") at fd.c:71
#5  0x24f40 in comm_accept (fd=2063838200, peer=0x7b0390b0, me=0x6b) at comm.c:574
#6  0x23874 in httpAccept (sock=33, notused=0xc00467a6) at client_side.c:1691
#7  0x25510 in comm_select_incoming () at comm.c:784
#8  0x25954 in comm_select (sec=29) at comm.c:1052
#9  0x3b04c in main (argc=1073745368, argv=0x40000dd8) at main.c:671
```

If possible, you might keep the coredump le around for a day or two. It is often helpful if we can ask you to send additional debugger output, such as the contents of some variables. But please note that a core le is only useful if paired with the exact same binary as generated the corele. If you recompile Squid then any coredumps from previous versions will be useless unless you have saved the corresponding Squid binaries, and any attempts to analyze such coredumps will most certainly give misleading information about the cause to the crash.

If you CANNOT get Squid to leave a core le for you then one of the following approaches can be used

First alternative is to start Squid under the contol of GDB

```
% gdb /path/to/squid
handle SIGPIPE pass nostop noprint
run -DNYCd3
[wait for crash]
backtrace
quit
```

The drawback from the above is that it isn't really suitable to run on a production system as Squid then won't restart automatically if it crashes. The good news is that it is fully possible to automate the process above to automatically get the stack trace and then restart Squid. Here is a short automated script that should work:

```
#!/bin/sh
```

```
trap "rm -f $$.gdb" 0
cat <<EOF >$$.gdb
handle SIGPIPE pass nostop noprint
run -DNYCd3
backtrace
quit
EOF
while sleep 2; do
   gdb -x $$.gdb /path/to/squid 2>&1 | tee -a squid.out
done
```

Other options if the above cannot be done is to:

a) Build Squid with the –enable-stacktraces option, if support exists for your OS (exists for Linux glibc on Intel, and Solaris with some extra libraries which seems rather impossible to nd these days..)

b) Run Squid using the "catchsegv" tool. (Linux glibc Intel)

but these approaches does not by far provide as much details as using gdb.

11.20 Debugging Squid

If you believe you have found a non-fatal bug (such as incorrect HTTP processing) please send us a section of your cache.log with debugging to demonstrate the problem. The cache.log le can become very large, so alternatively, you may want to copy it to an FTP or HTTP server where we can download it.

It is very simple to enable full debugging on a running squid process. Simply use the *-k debug* command line option:

```
% ./squid -k debug
```

This causes every *debug()* statement in the source code to write a line in the *cache.log* le. You also use the same command to restore Squid to normal debugging level.

To enable selective debugging (e.g. for one source le only), you need to edit *squid.conf* and add to the *debug_options* line. Every Squid source le is assigned a dierent debugging *section*. The debugging section assignments can be found by looking at the top of individual source les, or by reading the le *doc/debug-levels.txt* (correctly renamed to *debug-sections.txt* for Squid-2). You also specify the debugging *level* to control the amount of debugging. Higher levels result in more debugging messages. For example, to enable full debugging of Access Control functions, you would use

```
debug_options ALL,1 28,9
```

Then you have to restart or recongure Squid.

Once you have the debugging captured to *cache.log*, take a look at it yourself and see if you can make sense of the behaviour which you see. If not, please feel free to send your debugging output to the *squid-users* or *squid-bugs* lists.

11.21 FATAL: ipcache_init: DNS name lookup tests failed

Squid normally tests your system's DNS conguration before it starts server requests. Squid tries to resolve some common DNS names, as dened in the *dns_testnames* conguration directive. If Squid cannot resolve these names, it could mean:

1. your DNS nameserver is unreachable or not running.
2. your */etc/resolv.conf* le may contain incorrect information.
3. your */etc/resolv.conf* le may have incorrect permissions, and may be unreadable by Squid.

To disable this feature, use the *-D* command line option.

Note, Squid does NOT use the *dnsservers* to test the DNS. The test is performed internally, before the *dnsservers* start.

11.22 FATAL: Failed to make swap directory /var/spool/cache: (13) Permission denied

Starting with version 1.1.15, we have required that you rst run

```
squid -z
```

to create the swap directories on your lesystem. If you have set the *cache_eective _user* option, then the Squid process takes on the given userid before making the directories. If the *cache_dir* directory (e.g. /var/spool/cache) does not exist, and the Squid userid does not have permission to create it, then you will get the "permission denied" error. This can be simply xed by manually creating the cache directory.

```
# mkdir /var/spool/cache
# chown <userid> <groupid> /var/spool/cache
# squid -z
```

Alternatively, if the directory already exists, then your operating system may be returning "Permission Denied" instead of "File Exists" on the mkdir() system call. This *patch* `<store.c-mkdir.patch>` by *Miquel van Smoorenburg* `<mailto:miquels@cistron.nl>` should x it.

11.23 FATAL: Cannot open HTTP Port

Either (1) the Squid userid does not have permission to bind to the port, or (2) some other process has bound itself to the port. Remember that root privileges are required to open port numbers less than 1024. If you see this message when using a high port number, or even when starting Squid as root, then the port has already been opened by another process. Maybe you are running in the HTTP Accelerator mode and there is already a HTTP server running on port 80? If you're really stuck, install the way cool *lsof* `<ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/>` utility to show you which process has your port in use.

11.24 FATAL: All redirectors have exited!

This is explained in the 15.6.

11.25 FATAL: le _map_allocate: Exceeded lemap limit

See the next question.

11.26 FATAL: You've run out of swap le numbers.

Note: The information here applies to version 2.2 and earlier.

Squid keeps an in-memory bitmap of disk les that are available for use, or are being used. The size of this bitmap is determined at run name, based on two things: the size of your cache, and the average (mean) cache object size.

The size of your cache is specied in squid.conf, on the *cache_dir* lines. The mean object size can also be specied in squid.conf, with the 'store _avg_object_size' directive. By default, Squid uses 13 Kbytes as the average size.

When allocating the bitmaps, Squid allocates this many bits:

```
2 * cache_size / store_avg_object_size
```

So, if you exactly specify the correct average object size, Squid should have 50% lemap bits free when the cache is full. You can see how many lemap bits are being used by looking at the 'storedir' cache manager page. It looks like this:

```
Store Directory #0: /usr/local/squid/cache
First level subdirectories: 4
Second level subdirectories: 4
Maximum Size: 1024000 KB
Current Size: 924837 KB
Percent Used: 90.32%
Filemap bits in use: 77308 of 157538 (49%)
Flags:
```

Now, if you see the "You've run out of swap le numbers" message, then it means one of two things:

1. You've found a Squid bug.
2. Your cache's average le size is much smaller than the 'store _avg_object_size' value.

To check the average le size of object currently in your cache, look at the cache manager 'info' page, and you will nd a line like:

```
Mean Object Size:       11.96 KB
```

To make the warning message go away, set 'store_avg_object_size' to that value (or lower) and then restart Squid.

11.27 I am using up over 95% of the lemap bits?!!

Note: The information here is current for version 2.3

Calm down, this is now normal. Squid now dynamically allocates lemap bits based on the number of objects in your cache. You won't run out of them, we promise.

11.28 FATAL: Cannot open /usr/local/squid/logs/access.log: (13) Permission denied

In Unix, things like *processes* and *les* have an *owner*. For Squid, the process owner and le owner should be the same. If they are not the same, you may get messages like "permission denied."

To nd out who owns a le, use the *ls -l* command:

```
% ls -l /usr/local/squid/logs/access.log
```

A process is normally owned by the user who starts it. However, Unix sometimes allows a process to change its owner. If you specied a value for the *eective_user* option in *squid.conf*, then that will be the process owner. The les must be owned by this same userid.

If all this is confusing, then you probably should not be running Squid until you learn some more about Unix. As a reference, I suggest *Learning the UNIX Operating System, 4th Edition* <http://www.oreilly.com/catalog/lunix4/>.

11.29 When using a username and password, I can not access some les.

If I try by way of a test, to access

```
ftp://username:password@ftpserver/somewhere/foo.tar.gz
```

I get

```
somewhere/foo.tar.gz: Not a directory.
```

Use this URL instead:

```
ftp://username:password@ftpserver/%2fsomewhere/foo.tar.gz
```

11.30 pingerOpen: icmp_sock: (13) Permission denied

This means your *pinger* program does not have root priveleges. You should either do this:

```
% su
# make install-pinger
```

or

```
# chown root /usr/local/squid/bin/pinger
# chmod 4755 /usr/local/squid/bin/pinger
```

11.31 What is a forwarding loop?

A forwarding loop is when a request passes through one proxy more than once. You can get a forwarding loop if

a cache forwards requests to itself. This might happen with interception caching (or server acceleration) congurations.

a pair or group of caches forward requests to each other. This can happen when Squid uses ICP, Cache Digests, or the ICMP RTT database to select a next-hop cache.

Forwarding loops are detected by examining the *Via* request header. Each cache which "touches" a request must add its hostname to the *Via* header. If a cache notices its own hostname in this header for an incoming request, it knows there is a forwarding loop somewhere.

NOTE: Squid may report a forwarding loop if a request goes through two caches that have the same *visible_hostname* value. If you want to have multiple machines with the same *visible_hostname* then you must give each machine a dierent *unique_hostname* so that forwarding loops are correctly detected.

When Squid detects a forwarding loop, it is logged to the *cache.log* le with the recieved *Via* header. From this header you can determine which cache (the last in the list) forwarded the request to you.

One way to reduce forwarding loops is to change a *parent* relationship to a *sibling* relationship.

Another way is to use *cache_peer_access* rules. For example:

```
# Our parent caches
cache_peer A.example.com parent 3128 3130
cache_peer B.example.com parent 3128 3130
cache_peer C.example.com parent 3128 3130

# An ACL list
acl PEERS src A.example.com
acl PEERS src B.example.com
acl PEERS src C.example.com

# Prevent forwarding loops
cache_peer_access A.example.com allow !PEERS
cache_peer_access B.example.com allow !PEERS
cache_peer_access C.example.com allow !PEERS
```

The above conguration instructs squid to NOT forward a request to parents A, B, or C when a request is received from any one of those caches.

11.32 accept failure: (71) Protocol error

This error message is seen mostly on Solaris systems. *Mark Kennedy* <mailto:mtk@ny.ubs.com> gives a great explanation:

> Error 71 [EPROTO] is an obscure way of reporting that clients made it onto your server's TCP incoming connection queue but the client tore down the connection before the server could accept it. I.e. your server ignored its clients for too long. We've seen this happen when we ran out of le descriptors. I guess it could also happen if something made squid block for a long time.

11.33 storeSwapInFileOpened: ... Size mismatch

Got these messages in my cache log - I guess it means that the index contents do not match the contents on disk.

```
1998/09/23 09:31:30| storeSwapInFileOpened: /var/cache/00/00/00000015: Size mismatch: 776(fstat) != 378
1998/09/23 09:31:31| storeSwapInFileOpened: /var/cache/00/00/00000017: Size mismatch: 2571(fstat) != 41
```

What does Squid do in this case?

NOTE, these messages are specic to Squid-2. These happen when Squid reads an object from disk for a cache hit. After it opens the le, Squid checks to see if the size is what it expects it should be. If the size doesn't match, the error is printed. In this case, Squid does not send the wrong object to the client. It will re-fetch the object from the source.

11.34 Why do I get *fwdDispatch: Cannot retrieve 'https://www.buy.com/corp/ordertracking.asp'*

These messages are caused by buggy clients, mostly Netscape Navigator. What happens is, Netscape sends an HTTPS/SSL request over a persistent HTTP connection. Normally, when Squid gets an SSL request, it looks like this:

```
CONNECT www.buy.com:443 HTTP/1.0
```

Then Squid opens a TCP connection to the destination host and port, and the *real* request is sent encrypted over this connection. Thats the whole point of SSL, that all of the information must be sent encrypted.

With this client bug, however, Squid receives a request like this:

```
GET https://www.buy.com/corp/ordertracking.asp HTTP/1.0
Accept: */*
User-agent: Netscape ...
...
```

Now, all of the headers, and the message body have been sent, *unencrypted* to Squid. There is no way for Squid to somehow turn this into an SSL request. The only thing we can do is return the error message.

Note, this browser bug does represent a security risk because the browser is sending sensitive information unencrypted over the network.

11.35 Squid can't access URLs like http://3626046468/ab2/cybercards/moreinfo.html

by Dave J Woolley (DJW at bts dot co dot uk)

These are illegal URLs, generally only used by illegal sites; typically the web site that supports a spammer and is expected to survive a few hours longer than the spamming account.

Their intention is to:

confuse content ltering rules on proxies, and possibly some browsers' idea of whether they are trusted sites on the local intranet;

confuse whois (?);

make people think they are not IP addresses and unknown domain names, in an attempt to stop them trying to locate and complain to the ISP.

Any browser or proxy that works with them should be considered a security risk.

RFC 1738 <http://www.ietf.org/rfc/rfc1738.txt> has this to say about the hostname part of a URL:

The fully qualied domain name of a network host, or its IP address as a set of four decimal digit groups separated by ".". Fully qualied domain names take the form as described in Section 3.5 of RFC 1034 [13] and Section 2.1 of RFC 1123 [5]: a sequence of domain labels separated by ".", each domain label starting and ending with an alphanumerical character and possibly also containing "-" characters. The rightmost domain label will never start with a digit, though, which syntactically distinguishes all domain names from the IP addresses.

11.36 I get a lot of "URI has whitespace" error messages in my cache log, what should I do?

Whitespace characters (space, tab, newline, carriage return) are not allowed in URI's and URL's. Unfortunately, a number of Web services generate URL's with whitespace. Of course your favorite browser silently accomodates these bad URL's. The servers (or people) that generate these URL's are in violation of Internet standards. The whitespace characters should be encoded.

If you want Squid to accept URL's with whitespace, you have to decide how to handle them. There are four choices that you can set with the *uri_whitespace* option:

1. DENY: The request is denied with an "Invalid Request" message. This is the default.
2. ALLOW: The request is allowed and the URL remains unchanged.
3. ENCODE: The whitespace characters are encoded according to *RFC 1738* `<http://www.ietf.org/rfc/rfc1738.txt>`. This can be considered a violation of the HTTP specication.
4. CHOP: The URL is chopped at the rst whitespace character and then processed normally. This also can be considered a violation of HTTP.

11.37 commBind: Cannot bind socket FD 5 to 127.0.0.1:0: (49) Can't assign requested address

This likely means that your system does not have a loopback network device, or that device is not properly congured. All Unix systems should have a network device named *lo0*, and it should be congured with the address 127.0.0.1. If not, you may get the above error message. To check your system, run:

```
% ifconfig lo0
```

The result should look something like:

```
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet 127.0.0.1 netmask 0xff000000
```

If you use FreeBSD, see 14.2.4.

11.38 Unknown cache_dir type '/var/squid/cache'

The format of the *cache_dir* option changed with version 2.3. It now takes a *type* argument. All you need to do is insert `ufs` in the line, like this:

```
cache_dir ufs /var/squid/cache ...
```

11.39 unrecognized: 'cache_dns_program /usr/local/squid/bin/dnsserver'

As of Squid 2.3, the default is to use internal DNS lookup code. The *cache_dns_program* and *dns_children* options are not known squid.conf directives in this case. Simply comment out these two options.

If you want to use external DNS lookups, with the *dnsserver* program, then add this to your congure command:

```
--disable-internal-dns
```

11.40 Is *dns_defnames* broken in 2.3.STABLE1 and STABLE2?

Sort of. As of Squid 2.3, the default is to use internal DNS lookup code. The *dns_defnames* option is only used with the external *dnsserver* processes. If you relied on *dns_defnames* before, you have three choices:

1. See if the *append_domain* option will work for you instead.
2. Congure squid with –disable-internal-dns to use the external dnsservers.
3. Enhance *src/dns_internal.c* to understand the `search` and `domain` lines from */etc/resolv.conf*.

11.41 What does *sslReadClient: FD 14: read failure: (104) Connection reset by peer* mean?

"Connection reset by peer" is an error code that Unix operating systems sometimes return for *read*, *write*, *connect*, and other system calls.

Connection reset means that the other host, the peer, sent us a RESET packet on a TCP connection. A host sends a RESET when it receives an unexpected packet for a nonexistent connection. For example, if one side sends data at the same time that the other side closes a connection, when the other side receives the data it may send a reset back.

The fact that these messages appear in Squid's log might indicate a problem, such as a broken origin server or parent cache. On the other hand, they might be "normal," especially since some applications are known to force connection resets rather than a proper close.

You probably don't need to worry about them, unless you receive a lot of user complaints relating to SSL sites.

Rick Jones `<mailto:raj at cup dot hp dot com>` notes that if the server is running a Microsoft TCP stack, clients receive RST segments whenever the listen queue overows. In other words, if the server is really busy, new connections receive the reset message. This is contrary to rational behaviour, but is unlikely to change.

11.42 What does *Connection refused* mean?

This is an error message, generated by your operating system, in response to a *connect()* system call. It happens when there is no server at the other end listening on the port number that we tried to connect to.

Its quite easy to generate this error on your own. Simply telnet to a random, high numbered port:

```
% telnet localhost 12345
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection refused
```

It happens because there is no server listening for connections on port 12345.

When you see this in response to a URL request, it probably means the origin server web site is temporarily down. It may also mean that your parent cache is down, if you have one.

11.43 squid: ERROR: no running copy

You may get this message when you run commands like `squid -krotate`.

This error message usually means that the *squid.pid* le is missing. Since the PID le is normally present when squid is running, the absence of the PID le usually means Squid is not running. If you accidentally delete the PID le, Squid will continue running, and you won't be able to send it any signals.

If you accidentally removed the PID le, there are two ways to get it back.

1. run `ps` and nd the Squid process id. You'll probably see two processes, like this:

   ```
   bender-wessels % ps ax | grep squid
   83617  ??  Ss     0:00.00 squid -s
   83619  ??  S      0:00.48 (squid) -s (squid)
   ```

   You want the second process id, 83619 in this case. Create the PID le and put the process id number there. For example:

   ```
   echo 83619 > /usr/local/squid/logs/squid.pid
   ```

2. Use the above technique to nd the Squid process id. Send the process a HUP signal, which is the same as `squid -kreconfigure`:

   ```
   kill -HUP 83619
   ```

   The recongure process creates a new PID le automatically.

11.44 FATAL: getgrnam failed to nd groupid for eective group 'nogroup'

You are probably starting Squid as root. Squid is trying to nd a group-id that doesn't have any special priveleges that it will run as. The default is *nogroup*, but this may not be dened on your system. You need to edit *squid.conf* and set *cache_eective _group* to the name of an unpriveledged group from */etc/group*. There is a good chance that *nobody* will work for you.

11.45 "Unsupported Request Method and Protocol" for *https* URLs.

Note: The information here is current for version 2.3.

This is correct. Squid does not know what to do with an *https* URL. To handle such a URL, Squid would need to speak the SSL protocol. Unfortunately, it does not (yet).

Normally, when you type an *https* URL into your browser, one of two things happens.

1. The browser opens an SSL connection directly to the origin server.
2. The browser tunnels the request through Squid with the *CONNECT* request method.

The *CONNECT* method is a way to tunnel any kind of connection through an HTTP proxy. The proxy doesn't understand or interpret the contents. It just passes bytes back and forth between the client and server. For the gory details on tunnelling and the CONNECT method, please see *RFC 2817* <ftp://ftp.isi.edu/in-notes/rfc2817.txt> and *Tunneling TCP based protocols through Web proxy servers* <http://www.web-cache.com/Writings/Internet-Drafts/draft-luotonen-web-proxy-tunneling-01.txt> (expired).

11.46 Squid uses 100% CPU

There may be many causes for this.

Andrew Doroshenko reports that removing */dev/null*, or mounting a lesystem with the *nodev* option, can cause Squid to use 100% of CPU. His suggested solution is to "touch /dev/null."

11.47 Webmin's *cachemgr.cgi* crashes the operating system

Mikael Andersson reports that clicking on Webmin's *cachemgr.cgi* link creates numerous instances of *cachemgr.cgi* that quickly consume all available memory and brings the system to its knees.

Joe Cooper reports this to be caused by SSL problems in some browsers (mainly Netscape 6.x/Mozilla) if your Webmin is SSL enabled. Try with another browser such as Netscape 4.x or Microsoft IE, or disable SSL encryption in Webmin.

11.48 Segment Violation at startup or upon rst request

Some versions of GCC (notably 2.95.1 through 2.95.4 at least) have bugs with compiler optimization. These GCC bugs may cause NULL pointer accesses in Squid, resulting in a "FATAL: Received Segment Violation...dying" message and a core dump.

You can work around these GCC bugs by disabling compiler optimization. The best way to do that is start with a clean source tree and set the CC options specically:

```
% cd squid-x.y
% make distclean
% setenv CFLAGS='-g -Wall'
% ./configure ...
```

To check that you did it right, you can search for AC_CFLAGS in *src/Makele* :

```
% grep AC_CFLAGS src/Makefile
AC_CFLAGS       = -g -Wall
```

Now when you recompile, GCC won't try to optimize anything:

```
% make
Making all in lib...
gcc -g -Wall -I../include -I../include -c rfc1123.c
...etc...
```

NOTE: some people worry that disabling compiler optimization will negatively impact Squid's performance. The impact should be negligible, unless your cache is really busy and already runs at a high CPU usage. For most people, the compiler optimization makes little or no dierence at all.

11.49 urlParse: Illegal character in hostname 'proxy.mydomain.com:8080proxy.mydomain.c

By Yomler of fnac.net

A combination of a bad conguration of Internet Explorer and any application which use the cydoor DLLs will produce the entry in the log. See *cydoor.com* <http://www.cydoor.com/> for a complete list.

The bad conguration of IE is the use of a active conguration script (proxy.pac) and an active or inactive, but lled proxy settings. IE will only use the proxy.pac. Cydoor aps will use both and will generate the errors.

Disabling the old proxy settings in IE is not enought, you should delete them completely and only use the proxy.pac for example.

11.50 Requests for international domain names does not work

By Henrik Nordstøm

Some people have asked why requests for domain names using national symbols as "supported" by the certain domain registrars does not work in Squid. This is because there as of yet is no standard on how to manage national characters in the current Internet protocols such as HTTP or DNS. The current Internet standards is very strict on what is an acceptable hostname and only accepts A-Z a-z 0-9 and - in Internet hostname labels. Anything outside this is outside the current Internet standards and will cause interoperability issues such as the problems seen with such names and Squid.

When there is a consensus in the DNS and HTTP standardization groups on how to handle international domain names Squid will be changed to support this if any changes to Squid will be required.

If you are interested in the progress of the standardization process for international domain names please see the IETF IDN working group's *dedicated page* <http://www.i-d-n.net/>.

11.51 Why do I sometimes get "Zero Sized Reply"?

This happens when Squid makes a TCP connection to an origin server, but for some reason, the connection is closed before Squid reads any data. Depending on various factors, Squid may be able to retry the request again. If you see the "Zero Sized Reply" error message, it means that Squid was unable to retry, or that all retry attempts also failed.

What causes a connection to close prematurely? It could be a number of things, including:

1. An overloaded origin server.
2. TCP implementation/interoperability bugs.
3. Race conditions with HTTP persistent connections.
4. Buggy or miscongured NAT boxes, rewalls, and load-balancers.
5. Denial of service attacks.

You may be able to use *tcpdump* to track down and observe the problem.

Some users believe the problem is caused by very large cookies. One user reports that his Zero Sized Reply problem went away when he told Internet Explorer to not accept third-party cookies.

Here are some things you can try to reduce the occurance of the Zero Sized Reply error:

1. Delete or rename your cookie le and congure your browser to prompt you before accepting any new cookies.
2. Disable HTTP persistent connections with the *server_persistent_connections* and *client_persistent_connections* directives.
3. Disable any advanced TCP features on the Squid system. Disable ECN on Linux with `echo 0 > /proc/sys/net/ipv4/tcp_ecn/`.

If this error causes serious problems for you, Squid developers would be happy to help you uncover the problem. However, we will require high-quality debugging information from you, such as *tcpdump* output, server IP addresses, operating system versions, and *access.log* entries with full HTTP headers.

If you want to make Squid give the Zero Sized error on demand, you can use the short C program below. Simply compile and start the program on a system that doesn't already have a server running on port 80. Then try to connect to this fake server through Squid:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <assert.h>

int
main(int a, char **b)
{
        struct sockaddr_in S;
        int s,t,x;
        s = socket(PF_INET, SOCK_STREAM, 0);
        assert(s > 0);
        memset(&S, '\0', sizeof(S));
        S.sin_family = AF_INET;
        S.sin_port = htons(80);
        x = bind(s, (struct sockaddr *) &S, sizeof(S));
        assert(x == 0);
        x = listen(s, 10);
        assert(x == 0);
        while (1) {
                struct sockaddr_in F;
                int fl = sizeof(F);
                t = accept(s, (struct sockaddr *) &F, &fl);
                fprintf(stderr, "accpeted FD %d from %s:%d\n",
                        t, inet_ntoa(F.sin_addr), (int)ntohs(F.sin_port));
                close(t);
                fprintf(stderr, "closed FD %d\n", t);
        }
        return 0;
}
```

12 How does Squid work?

12.1 What are cachable objects?

An Internet Object is a le, document or response to a query for an Internet service such as FTP, HTTP, or gopher. A client requests an Internet object from a caching proxy; if the object is not already cached, the proxy server fetches the object (either from the host specied in the URL or from a parent or sibling cache) and delivers it to the client.

12.2 What is the ICP protocol?

ICP is a protocol used for communication among squid caches. The ICP protocol is dened in two Internet RFC's. *RFC 2186* `<http://www.ircache.net/Cache/ICP/rfc2186.txt>` describes the protocol itself, while *RFC 2187* `<http://www.ircache.net/Cache/ICP/rfc2187.txt>` describes the application of ICP to hierarchical Web caching.

ICP is primarily used within a cache hierarchy to locate specic objects in sibling caches. If a squid cache does not have a requested document, it sends an ICP query to its siblings, and the siblings respond with ICP replies indicating a "HIT" or a "MISS." The cache then uses the replies to choose from which cache to resolve its own MISS.

ICP also supports multiplexed transmission of multiple object streams over a single TCP connection. ICP is currently implemented on top of UDP. Current versions of Squid also support ICP via multicast.

12.3 What is the *dnsserver*?

The *dnsserver* is a process forked by *squid* to resolve IP addresses from domain names. This is necessary because the `gethostbyname(3)` function blocks the calling process until the DNS query is completed.

Squid must use non-blocking I/O at all times, so DNS lookups are implemented external to the main process. The *dnsserver* processes do not cache DNS lookups, that is implemented inside the *squid* process.

12.4 What is the *ftpget* program for?

ftpget exists only in Squid 1.1 and Squid 1.0 versions.

The *ftpget* program is an FTP client used for retrieving les from FTP servers. Because the FTP protocol is complicated, it is easier to implement it separately from the main *squid* code.

12.5 FTP PUT's don't work!

FTP PUT should work with Squid-2.0 and later versions. If you are using Squid-1.1, then you need to upgrade before PUT will work.

12.6 What is a cache hierarchy? What are parents and siblings?

A cache hierarchy is a collection of caching proxy servers organized in a logical parent/child and sibling arrangement so that caches closest to Internet gateways (closest to the backbone transit entry-points) act as parents to caches at locations farther from the backbone. The parent caches resolve "misses" for their children. In other words, when a cache requests an object from its parent, and the parent does not have the object in its cache, the parent fetches the object, caches it, and delivers it to the child. This ensures that

the hierarchy achieves the maximum reduction in bandwidth utilization on the backbone transit links, helps reduce load on Internet information servers outside the network served by the hierarchy, and builds a rich cache on the parents so that the other child caches in the hierarchy will obtain better "hit" rates against their parents.

In addition to the parent-child relationships, squid supports the notion of siblings: caches at the same level in the hierarchy, provided to distribute cache server load. Each cache in the hierarchy independently decides whether to fetch the reference from the object's home site or from parent or sibling caches, using a a simple resolution protocol. Siblings will not fetch an object for another sibling to resolve a cache "miss."

12.7 What is the Squid cache resolution algorithm?

Send ICP queries to all appropriate siblings

Wait for all replies to arrive with a congurable timeout (the default is two seconds).

Begin fetching the object upon receipt of the rst HIT reply, or

Fetch the object from the rst parent which replied with MISS (subject to weighting values), or

Fetch the object from the source

The algorithm is somewhat more complicated when rewalls are involved.

The `single_parent_bypass` directive can be used to skip the ICP queries if the only appropriate sibling is a parent cache (i.e., if there's only one place you'd fetch the object from, why bother querying?)

12.8 What features are Squid developers currently working on?

There are several open issues for the caching project namely more automatic load balancing and (both congured and dynamic) selection of parents, routing, multicast cache-to-cache communication, and better recognition of URLs that are not worth caching.

For our other to-do list items, please see our "TODO" le in the recent source distributions.

Prospective developers should review the resources available at the *Squid developers corner* `<http://www.squid-cache.org/Devel/>`

12.9 Tell me more about Internet trac workloads

Workload can be characterized as the burden a client or group of clients imposes on a system. Understanding the nature of workloads is important to the managing system capacity.

If you are interested in Internet trac workloads then NLANR's *Network Analysis activities* `<http://www.nlanr.net/NA/>` is a good place to start.

12.10 What are the tradeos of caching with the NLANR cache system?

The NLANR root caches are at the NSF supercomputer centers (SCCs), which are interconnected via NSF's high speed backbone service (vBNS). So inter-cache communication between the NLANR root caches does not cross the Internet.

The benets of hierarchical caching (namely, reduced network bandwidth consumption, reduced access latency, and improved resiliency) come at a price. Caches higher in the hierarchy must eld the misses of their

descendents. If the equilibrium hit rate of a leaf cache is 50%, half of all leaf references have to be resolved through a second level cache rather than directly from the object's source. If this second level cache has most of the documents, it is usually still a win, but if higher level caches often don't have the document, or become overloaded, then they could actually increase access latency, rather than reduce it.

12.11 Where can I nd out more about rewalls?

Please see the *Firewalls FAQ* <http://www.faqs.org/faqs/firewalls-faq/> information site.

12.12 What is the "Storage LRU Expiration Age?"

For example:

```
Storage LRU Expiration Age:      4.31 days
```

The LRU expiration age is a dynamically-calculated value. Any objects which have not been accessed for this amount of time will be removed from the cache to make room for new, incoming objects. Another way of looking at this is that it would take your cache approximately this many days to go from empty to full at your current trac levels.

As your cache becomes more busy, the LRU age becomes lower so that more objects will be removed to make room for the new ones. Ideally, your cache will have an LRU age value in the range of at least 3 days. If the LRU age is lower than 3 days, then your cache is probably not big enough to handle the volume of requests it receives. By adding more disk space you could increase your cache hit ratio.

The conguration parameter *reference_age* places an upper limit on your cache's LRU expiration age.

12.13 What is "Failure Ratio at 1.01; Going into hit-only-mode for 5 minutes"?

Consider a pair of caches named A and B. It may be the case that A can reach B, and vice-versa, but B has poor reachability to the rest of the Internet. In this case, we would like B to recognize that it has poor reachability and somehow convey this fact to its neighbor caches.

Squid will track the ratio of failed-to-successful requests over short time periods. A failed request is one which is logged as ERR_DNS_FAIL, ERR_CONNECT_FAIL, or ERR_READ_ERROR. When the failed-to-successful ratio exceeds 1.0, then Squid will return ICP_MISS_NOFETCH instead of ICP_MISS to neighbors. Note, Squid will still return ICP_HIT for cache hits.

12.14 Does squid periodically re-read its conguration le?

No, you must send a HUP signal to have Squid re-read its conguration le, including access control lists. An easy way to do this is with the *-k* command line option:

```
squid -k reconfigure
```

12.15 How does *unlinkd* work?

unlinkd is an external process used for unlinking unused cache les. Performing the unlink operation in an external process opens up some race-condition problems for Squid. If we are not careful, the following sequence of events could occur:

1. An object with swap le number **S** is removed from the cache.
2. We want to unlink le **F** which corresponds to swap le number **S**, so we write pathname **F** to the *unlinkd* socket. We also mark **S** as available in the lemap.
3. We have a new object to swap out. It is allocated to the rst available le number, which happens to be **S**. Squid opens le **F** for writing.
4. The *unlinkd* process reads the request to unlink **F** and issues the actual unlink call.

So, the problem is, how can we guarantee that *unlinkd* will not remove a cache le that Squid has recently allocated to a new object? The approach we have taken is to have Squid keep a stack of unused (but not deleted!) swap le numbers. The stack size is hard-coded at 128 entries. We only give unlink requests to *unlinkd* when the unused le number stack is full. Thus, if we ever have to start unlinking les, we have a pool of 128 le numbers to choose from which we know will not be removed by *unlinkd*.

In terms of implementation, the only way to send unlink requests to the *unlinkd* process is via the *storePutUnusedFileno* function.

Unfortunately there are times when Squid can not use the *unlinkd* process but must call *unlink(2)* directly. One of these times is when the cache swap size is over the high water mark. If we push the released le numbers onto the unused le number stack, and the stack is not full, then no les will be deleted, and the actual disk usage will remain unchanged. So, when we exceed the high water mark, we must call *unlink(2)* directly.

12.16 What is an icon URL?

One of the most unpleasant things Squid must do is generate HTML pages of Gopher and FTP directory listings. For some strange reason, people like to have little *icons* next to each listing entry, denoting the type of object to which the link refers (image, text le, etc.).

In Squid 1.0 and 1.1, we used internal browser icons with names like *gopher-internal-image*. Unfortunately, these were not very portable. Not all browsers had internal icons, or even used the same names. Perhaps only Netscape and Mosaic used these names.

For Squid 2 we include a set of icons in the source distribution. These icon les are loaded by Squid as cached objects at runtime. Thus, every Squid cache now has its own icons to use in Gopher and FTP listings. Just like other objects available on the web, we refer to the icons with *Uniform Resource Locators* `<ftp://ftp.isi.edu/in-notes/rfc1738.txt>`, or *URLs*.

12.17 Can I make my regular FTP clients use a Squid cache?

Nope, its not possible. Squid only accepts HTTP requests. It speaks FTP on the *server-side*, but **not** on the *client-side*.

The very cool *wget* `<ftp://gnjilux.cc.fer.hr/pub/unix/util/wget/>` will download FTP URLs via Squid (and probably any other proxy cache).

12.18 Why is the select loop average time so high?

Is there any way to speed up the time spent dealing with select? Cachemgr shows:

```
Select loop called: 885025 times, 714.176 ms avg
```

This number is NOT how much time it takes to handle ledescriptor I/O. We simply count the number of times select was called, and divide the total process running time by the number of select calls.

This means, on average it takes your cache .714 seconds to check all the open le descriptors once. But this also includes time select() spends in a wait state when there is no I/O on any le descriptors. My relatively idle workstation cache has similar numbers:

```
Select loop called: 336782 times, 715.938 ms avg
```

But my busy caches have much lower times:

```
Select loop called: 16940436 times, 10.427 ms avg
Select loop called: 80524058 times, 10.030 ms avg
Select loop called: 10590369 times, 8.675 ms avg
Select loop called: 84319441 times, 9.578 ms avg
```

12.19 How does Squid deal with Cookies?

The presence of Cookies headers in **requests** does not aect whether or not an HTTP reply can be cached. Similarly, the presense of *Set-Cookie* headers in **replies** does not aect whether the reply can be cached.

The proper way to deal with *Set-Cookie* reply headers, according to *RFC 2109* <ftp://ftp.isi.edu/in-notes/rfc2109.txt> is to cache the whole object, *EXCEPT* the *Set-Cookie* header lines.

With Squid-1.1, we can not lter out specic HTTP headers, so Squid-1.1 does not cache any response which contains a *Set-Cookie* header.

With Squid-2, however, we can lter out specic HTTP headers. But instead of ltering them on the receiving-side, we lter them on the sending-side. Thus, Squid-2 does cache replies with *Set-Cookie* headers, but it lters out the *Set-Cookie* header itself for cache hits.

12.20 How does Squid decide when to refresh a cached object?

When checking the object freshness, we calculate these values:

OBJ_DATE is the time when the object was given out by the origin server. This is taken from the HTTP Date reply header.

OBJ_LASTMOD is the time when the object was last modied, given by the HTTP Last-Modied reply header.

OBJ_AGE is how much the object has aged *since* it was retrieved:

```
OBJ_AGE = NOW - OBJ_DATE
```

LM_AGE is how old the object was *when* it was retrieved:

```
LM_AGE = OBJ_DATE - OBJ_LASTMOD
```

LM_FACTOR is the ratio of *OBJ_AGE* to *LM_AGE*:

```
LM_FACTOR = OBJ_AGE / LM_AGE
```

CLIENT_MAX_AGE is the (optional) maximum object age the client will accept as taken from the HTTP/1.1 Cache-Control request header.

EXPIRES is the (optional) expiry time from the server reply headers.

These values are compared with the parameters of the *refresh_pattern* rules. The refresh parameters are:

URL regular expression

CONF_MIN: The time (in minutes) an object without an explicit expiry time should be considered fresh. The recommended value is 0, any higher values may cause dynamic applications to be erronously cached unless the application designer has taken the appropriate actions.

CONF_PERCENT: A percentage of the objects age (time since last modication age) an object without explicit exipry time will be considered fresh.

CONF_MAX: An upper limit on how long objects without an explicit expiry time will be considered fresh.

The URL regular expressions are checked in the order listed until a match is found. Then the algorithms below are applied for determining if an object is fresh or stale.

12.20.1 Squid-1.1 and Squid-1.NOVM algorithm

```
if (CLIENT_MAX_AGE)
    if (OBJ_AGE > CLIENT_MAX_AGE)
        return STALE
if (OBJ_AGE <= CONF_MIN)
    return FRESH
if (EXPIRES) {
    if (EXPIRES <= NOW)
        return STALE
    else
        return FRESH
}
if (OBJ_AGE > CONF_MAX)
    return STALE
if (LM_FACTOR < CONF_PERCENT)
    return FRESH
return STALE
```

Kolics Bertold <mailto:bertold@tohotom.vein.hu> has made an excellent *ow chart diagram* <http://www.squid-cache.org/Doc/FAQ/refresh-flowchart.gif> showing this process.

12.20.2 Squid-2 algorithm

For Squid-2 the refresh algorithm has been slightly modied to give the *EXPIRES* value a higher precedence, and the *CONF_MIN* value lower precedence:

```
if (EXPIRES) {
    if (EXPIRES <= NOW)
        return STALE
```

```
    else
        return FRESH
}
if (CLIENT_MAX_AGE)
    if (OBJ_AGE > CLIENT_MAX_AGE)
        return STALE
if (OBJ_AGE > CONF_MAX)
    return STALE
if (OBJ_DATE > OBJ_LASTMOD) {
    if (LM_FACTOR < CONF_PERCENT)
        return FRESH
    else
        return STALE
}
if (OBJ_AGE <= CONF_MIN)
    return FRESH
return STALE
```

12.21 What exactly is a *deferred read*?

The cachemanager I/O page lists *deferred reads* for various server-side protocols.

Sometimes reading on the server-side gets ahead of writing to the client-side. Especially if your cache is on a fast network and your clients are connected at modem speeds. Squid-1.1 will read up to 256k (per request) ahead before it starts to defer the server-side reads.

12.22 Why is my cache's inbound trac equal to the outbound trac?

I've been monitoring the trac on my cache's ethernet adapter an found a behavior I can't explain: the inbound trac is equal to the outbound trac. The dierences are negligible. The hit ratio reports 40%. Shouldn't the outbound be at least 40% greater than the inbound?

by *David J N Begley* <mailto:david@avarice.nepean.uws.edu.au>

I can't account for the exact behavior you're seeing, but I can oer this advice; whenever you start measuring raw Ethernet or IP trac on interfaces, you can forget about getting all the numbers to exactly match what Squid reports as the amount of trac it has sent/received.

Why?

Squid is an application - it counts whatever data is sent to, or received from, the lower-level networking functions; at each successively lower layer, additional trac is involved (such as header overhead, retransmits and fragmentation, unrelated broadcasts/trac, etc.). The additional trac is never seen by Squid and thus isn't counted - but if you run MRTG (or any SNMP/RMON measurement tool) against a specic interface, all this additional trac will "magically appear".

Also remember that an interface has no concept of upper-layer networking (so an Ethernet interface doesn't distinguish between IP trac that's entirely internal to your organization, and trac that's to/from the Internet); this means that when you start measuring an interface, you have to be aware of *what* you are measuring before you can start comparing numbers elsewhere.

It is possible (though by no means guaranteed) that you are seeing roughly equivalent input/output because you're measuring an interface that both retrieves data from the outside world (Internet), *and* serves it to

end users (internal clients). That wouldn't be the whole answer, but hopefully it gives you a few ideas to start applying to your own circumstance.

To interpret any statistic, you have to rst know what you are measuring; for example, an interface counts inbound and outbound bytes - that's it. The interface doesn't distinguish between inbound bytes from external Internet sites or from internal (to the organization) clients (making requests). If you want that, try looking at RMON2.

Also, if you're talking about a 40% hit rate in terms of object requests/counts then there's absolutely no reason why you should expect a 40% reduction in trac; after all, not every request/object is going to be the same size so you may be saving a lot in terms of requests but very little in terms of actual trac.

12.23 How come some objects do not get cached?

To determine whether a given object may be cached, Squid takes many things into consideration. The current algorithm (for Squid-2) goes something like this:

1. Responses with *Cache-Control: Private* are NOT cachable.
2. Responses with *Cache-Control: No-Cache* are NOT cachable.
3. Responses with *Cache-Control: No-Store* are NOT cachable.
4. Responses for requests with an *Authorization* header are cachable ONLY if the reponse includes *Cache-Control: Public*.
5. Responses with *Vary* headers are NOT cachable because Squid does not yet support Vary features.
6. The following HTTP status codes are cachable:

   200 OK

   203 Non-Authoritative Information

   300 Multiple Choices

   301 Moved Permanently

   410 Gone

   However, if Squid receives one of these responses from a neighbor cache, it will NOT be cached if ALL of the *Date*, *Last-Modied* , and *Expires* reply headers are missing. This prevents such objects from bouncing back-and-forth between siblings forever.
7. A 302 Moved Temporarily response is cachable ONLY if the response also includes an *Expires* header.
8. The following HTTP status codes are "negatively cached" for a short amount of time (congurable):

   204 No Content

   305 Use Proxy

   400 Bad Request

   403 Forbidden

   404 Not Found

   405 Method Not Allowed

   414 Request-URI Too Large

   500 Internal Server Error

501 Not Implemented

502 Bad Gateway

503 Service Unavailable

504 Gateway Time-out

9. All other HTTP status codes are NOT cachable, including:

206 Partial Content

303 See Other

304 Not Modied

401 Unauthorized

407 Proxy Authentication Required

12.24 What does *keep-alive ratio* mean?

The *keep-alive ratio* shows up in the *server_list* cache manager page for Squid 2.

This is a mechanism to try detecting neighbor caches which might not be able to deal with persistent connections. Every time we send a *proxy-connection: keep-alive* request header to a neighbor, we count how many times the neighbor sent us a *proxy-connection: keep-alive* reply header. Thus, the *keep-alive ratio* is the ratio of these two counters.

If the ratio stays above 0.5, then we continue to assume the neighbor properly implements persistent connections. Otherwise, we will stop sending the keep-alive request header to that neighbor.

12.25 How does Squid's cache replacement algorithm work?

Squid uses an LRU (least recently used) algorithm to replace old cache objects. This means objects which have not been accessed for the longest time are removed rst. In the source code, the StoreEntry- >lastref value is updated every time an object is accessed.

Objects are not necessarily removed "on-demand." Instead, a regularly scheduled event runs to periodically remove objects. Normally this event runs every second.

Squid keeps the cache disk usage between the low and high water marks. By default the low mark is 90%, and the high mark is 95% of the total congured cache size. When the disk usage is close to the low mark, the replacement is less aggressive (fewer objects removed). When the usage is close to the high mark, the replacement is more aggressive (more objects removed).

When selecting objects for removal, Squid examines some number of objects and determines which can be removed and which cannot. A number of factors determine whether or not any given object can be removed. If the object is currently being requested, or retrieved from an upstream site, it will not be removed. If the object is "negatively-cached" it will be removed. If the object has a private cache key, it will be removed (there would be no reason to keep it – because the key is private, it can never be "found" by subsequent requests). Finally, if the time since last access is greater than the LRU threshold, the object is removed.

The LRU threshold value is dynamically calculated based on the current cache size and the low and high marks. The LRU threshold scaled exponentially between the high and low water marks. When the store swap size is near the low water mark, the LRU threshold is large. When the store swap size is near the high water mark, the LRU threshold is small. The threshold automatically adjusts to the rate of incoming requests. In fact, when your cache size has stabilized, the LRU threshold represents how long it takes to ll (or fully replace) your cache at the current request rate. Typical values for the LRU threshold are 1 to 10 days.

Back to selecting objects for removal. Obviously it is not possible to check every object in the cache every time we need to remove some of them. We can only check a small subset each time. The way in which this is implemented is very dierent between Squid-1.1 and Squid-2.

12.25.1 Squid 1.1

The Squid cache storage is implemented as a hash table with some number of "hash buckets." Squid-1.1 scans one bucket at a time and sorts all the objects in the bucket by their LRU age. Objects with an LRU age over the threshold are removed. The scan rate is adjusted so that it takes approximately 24 hours to scan the entire cache. The store buckets are randomized so that we don't always scan the same buckets at the same time of the day.

This algorithm has some aws. Because we only scan one bucket, there are going to be better candidates for removal in some of the other 16,000 or so buckets. Also, the qsort() function might take a non-trivial amount of CPU time, depending on how many entries are in each bucket.

12.25.2 Squid 2

For Squid-2 we eliminated the need to use qsort() by indexing cached objects into an automatically sorted linked list. Every time an object is accessed, it gets moved to the top of the list. Over time, the least used objects migrate to the bottom of the list. When looking for objects to remove, we only need to check the last 100 or so objects in the list. Unfortunately this approach increases our memory usage because of the need to store three additional pointers per cache object. But for Squid-2 we're still ahead of the game because we also replaced plain-text cache keys with MD5 hashes.

12.26 What are private and public keys?

keys refers to the database keys which Squid uses to index cache objects. Every object in the cache–whether saved on disk or currently being downloaded–has a cache key. For Squid-1.0 and Squid-1.1 the cache key was basically the URL. Squid-2 uses MD5 checksums for cache keys.

The Squid cache uses the notions of *private* and *public* cache keys. An object can start out as being private, but may later be changed to public status. Private objects are associated with only a single client whereas a public object may be sent to multiple clients at the same time. In other words, public objects can be located by any cache client. Private keys can only be located by a single client–the one who requested it.

Objects are changed from private to public after all of the HTTP reply headers have been received and parsed. In some cases, the reply headers will indicate the object should not be made public. For example, if the *no-cache* Cache-Control directive is used.

12.27 What is FORW_VIA_DB for?

We use it to collect data for *Plankton* <http://www.ircache.net/Cache/Plankton/>.

12.28 Does Squid send packets to port 7 (echo)? If so, why?

It may. This is an old feature from the Harvest cache software. The cache would send ICP "SECHO" message to the echo ports of origin servers. If the SECHO message came back before any of the other ICP replies, then it meant the origin server was probably closer than any neighbor cache. In that case Harvest/Squid sent the request directly to the origin server.

With more attention focused on security, many administrators lter UDP packets to port 7. The Computer Emergency Response Team (CERT) once issued an advisory note (*CA-96.01: UDP Port Denial-of-Service Attack* <http://www.cert.org/advisories/CA-96.01.UDP_service_denial.html>) that says UDP echo and chargen services can be used for a denial of service attack. This made admins extremely nervous about any packets hitting port 7 on their systems, and they made complaints.

The *source_ping* feature has been disabled in Squid-2. If you're seeing packets to port 7 that are coming from a Squid cache (remote port 3130), then its probably a very old version of Squid.

12.29 What does "WARNING: Reply from unknown nameserver [a.b.c.d]" mean?

It means Squid sent a DNS query to one IP address, but the response came back from a dierent IP address. By default Squid checks that the addresses match. If not, Squid ignores the response.

There are a number of reasons why this would happen:

1. Your DNS name server just works this way, either becuase its been congured to, or because its stupid and doesn't know any better.
2. You have a weird broadcast address, like 0.0.0.0, in your */etc/resolv.conf* le.
3. Somebody is trying to send spoofed DNS responses to your cache.

If you recognize the IP address in the warning as one of your name server hosts, then its probably numbers (1) or (2).

You can make these warnings stop, and allow responses from "unknown" name servers by setting this conguration option:

```
ignore_unknown_nameservers off
```

12.30 How does Squid distribute cache les among the available directories?

Note: The information here is current for version 2.2.

See *storeDirMapAllocate()* in the source code.

When Squid wants to create a new disk le for storing an object, it rst selects which *cache_dir* the object will go into. This is done with the *storeDirSelectSwapDir()* function. If you have *N* cache directories, the function identies the *3N/4* (75%) of them with the most available space. These directories are then used, in order of having the most available space. When Squid has stored one URL to each of the *3N/4 cache_dir*'s, the process repeats and *storeDirSelectSwapDir()* nds a new set of *3N/4* cache directories with the most available space.

Once the *cache_dir* has been selected, the next step is to nd an available *swap le number* . This is accomplished by checking the *le map* , with the *le _map_allocate()* function. Essentially the swap le numbers are allocated sequentially. For example, if the last number allocated happens to be 1000, then the next one will be the rst number after 1000 that is not already being used.

12.31 Why do I see negative byte hit ratio?

Byte hit ratio is calculated a bit dierently than Request hit ratio. Squid counts the number of bytes read from the network on the server-side, and the number of bytes written to the client-side. The byte hit ratio is calculated as

```
(client_bytes - server_bytes) / client_bytes
```

If server_bytes is greater than client_bytes, you end up with a negative value.

The server_bytes may be greater than client_bytes for a number of reasons, including:

Cache Digests and other internally generated requests. Cache Digest messages are quite large. They are counted in the server_bytes, but since they are consumed internally, they do not count in client_bytes.

User-aborted requests. If your *quick_abort* setting allows it, Squid sometimes continues to fetch aborted requests from the server-side, without sending any data to the client-side.

Some range requests, in combination with Squid bugs, can consume more bandwidth on the server-side than on the client-side. In a range request, the client is asking for only some part of the object. Squid may decide to retrieve the whole object anyway, so that it can be used later on. This means downloading more from the server than sending to the client. You can aect this behavior with the *range_oset _limit* option.

12.32 What does "Disabling use of private keys" mean?

First you need to understand the 12.26.

When Squid sends ICP queries, it uses the ICP *reqnum* eld to hold the private key data. In other words, when Squid gets an ICP reply, it uses the *reqnum* value to build the private cache key for the pending object.

Some ICP implementations always set the *reqnum* eld to zero when they send a reply. Squid can not use private cache keys with such neighbor caches because Squid will not be able to locate cache keys for those ICP replies. Thus, if Squid detects a neighbor cache that sends zero reqnum's, it disables the use of private cache keys.

Not having private cache keys has some important privacy implications. Two users could receive one response that was meant for only one of the users. This response could contain personal, condential information. You will need to disable the "zero reqnum" neighbor if you want Squid to use private cache keys.

12.33 What is a half-closed ledescriptor?

TCP allows connections to be in a "half-closed" state. This is accomplished with the *shutdown(2)* system call. In Squid, this means that a client has closed its side of the connection for writing, but leaves it open for reading. Half-closed connections are tricky because Squid can't tell the dierence between a half-closed connection, and a fully closed one.

If Squid tries to read a connection, and *read()* returns 0, and Squid knows that the client doesn't have the whole response yet, Squid puts marks the ledescriptor as half-closed. Most likely the client has aborted the request and the connection is really closed. However, there is a slight chance that the client is using the *shutdown()* call, and that it can still read the response.

To disable half-closed connections, simply put this in squid.conf:

```
half_closed_clients off
```

Then, Squid will always close its side of the connection instead of marking it as half-closed.

12.34 What does –enable-heap-replacement do?

Squid has traditionally used an LRU replacement algorithm. As of *version 2.3* </Versions/v2/2.3/>, you can use some other replacement algorithms by using the *–enable-heap-replacement* congure option. Currently, the heap replacement code supports two additional algorithms: LFUDA, and GDS.

With Squid version 2.4 and later you should use this congure option:

```
./configure --enable-removal-policies=heap
```

Then, in *squid.conf*, you can select dierent policies with the *cache_replacement_policy* option. See the *squid.conf* comments for details.

The LFUDA and GDS replacement code was contributed by John Dilley and others from Hewlett-Packard. Their work is described in these papers:

1. *Enhancement and Validation of Squid's Cache Replacement Policy* <http://www.hpl.hp.com/techreports/1999/HPL-1999-69.html> (HP Tech Report).
2. *Enhancement and Validation of the Squid Cache Replacement Policy* <http://workshop.ircache.net/Papers/dilley-abstract.html> (WCW 1999 paper).

12.35 Why is actual lesystem space used greater than what Squid thinks?

If you compare *df* output and cachemgr *storedir* output, you will notice that actual disk usage is greater than what Squid reports. This may be due to a number of reasons:

Squid doesn't keep track of the size of the *swap.state* le, which normally resides on each *cache_dir*.

Directory entries and take up lesystem space.

Other applications might be using the same disk partition.

Your lesystem block size might be larger than what Squid thinks. When calculating total disk usage, Squid rounds le sizes up to a whole number of 1024 byte blocks. If your lesystem uses larger blocks, then some "wasted" space is not accounted.

12.36 How do *positive_dns_ttl* and *negative_dns_ttl* work?

positive_dns_ttl is how long Squid caches a successful DNS lookup. Similarly, *negative_dns_ttl* is how long Squid caches a failed DNS lookup.

positive_dns_ttl is not always used. It is NOT used in the following cases:

Squid-2.3 and later versions with internal DNS lookups. Internal lookups are the default for Squid-2.3 and later.

If you applied the "DNS TTL" 2.9 for BIND.

If you are using FreeBSD, then it already has the DNS TTL patch built in.

Let's say you have the following settings:

```
positive_dns_ttl 1 hours
negative_dns_ttl 1 minutes
```

When Squid looks up a name like *www.squid-cache.org*, it gets back an IP address like 204.144.128.89. The address is cached for the next hour. That means, when Squid needs to know the address for *www.squid-cache.org* again, it uses the cached answer for the next hour. After one hour, the cached information expires, and Squid makes a new query for the address of *www.squid-cache.org*.

If you have the DNS TTL patch, or are using internal lookups, then each hostname has its own TTL value, which was set by the domain name administrator. You can see these values in the 'ipcache' cache manager page. For example:

```
Hostname                      Flags lstref    TTL N
www.squid-cache.org               C   73043  12784  1( 0)  204.144.128.89-OK
www.ircache.net                   C   73812  10891  1( 0)   192.52.106.12-OK
polygraph.ircache.net             C  241768 -181261  1( 0)   192.52.106.12-OK
```

The TTL eld shows how how many seconds until the entry expires. Negative values mean the entry is already expired, and will be refreshed upon next use.

The *negative_dns_ttl* species how long to cache failed DNS lookups. When Squid fails to resolve a hostname, you can be pretty sure that it is a real failure, and you are not likely to get a successful answer within a short time period. Squid retries its lookups many times before declaring a lookup has failed. If you like, you can set *negative_dns_ttl* to zero.

12.37 What does *swapin MD5 mismatch* mean?

It means that Squid opened up a disk le to serve a cache hit, but it found that the stored object doesn't match what the user's request. Squid stores the MD5 digest of the URL at the start of each disk le. When the le is opened, Squid checks that the disk le MD5 matches the MD5 of the URL requested by the user. If they don't match, the warning is printed and Squid forwards the request to the origin server.

You do not need to worry about this warning. It means that Squid is recovering from a corrupted cache directory.

12.38 What does *failed to unpack swaple meta data* mean?

Each of Squid's disk cache les has a metadata section at the beginning. This header is used to store the URL MD5, some StoreEntry data, and more. When Squid opens a disk le for reading, it looks for the meta data header and unpacks it.

This warning means that Squid couln't unpack the meta data. This is non-fatal bug, from which Squid can recover. Perhaps the meta data was just missing, or perhaps the le got corrupted.

You do not need to worry about this warning. It means that Squid is double-checking that the disk le matches what Squid thinks should be there, and the check failed. Squid recorvers and generates a cache miss in this case.

12.39 Why doesn't Squid make *ident* lookups in interception mode?

Its a side-eect of the way interception proxying works.

When Squid is congured for interception proxying, the operating system pretends that it is the origin server. That means that the "local" socket address for intercepted TCP connections is really the origin server's IP address. If you run *netstat -n* on your interception proxy, you'll see a lot of foreign IP addresses in the *Local Address* column.

When Squid wants to make an ident query, it creates a new TCP socket and *binds* the local endpoint to the same IP address as the local end of the client's TCP connection. Since the local address isn't really local (its some far away origin server's IP address), the *bind()* system call fails. Squid handles this as a failed ident lookup.

So why bind in that way? If you know you are interception proxying, then why not bind the local endpoint to the host's (intranet) IP address? Why make the masses suer needlessly?

Because thats just how ident works. Please read *RFC 931* <ftp://ftp.isi.edu/in-notes/rfc931.txt>, in particular the RESTRICTIONS section.

12.40 dnsSubmit: queue overload, rejecting blah

This means that you are using external *dnsserver* processes for lookups, and all processes are busy, and Squid's pending queue is full. Each *dnsserver* program can only handle one request at a time. When all *dnsserver* processes are busy, Squid queues up requests, but only to a certain point.

To alleviate this condition, you need to either (1) increase the number of *dnsserver* processes by changing the value for *dns_children* in your cong le, or (2) switch to using Squid's internal DNS client code.

Note that in some versions, Squid limits *dns_children* to 32. To increase it beyond that value, you would have to edit the source code.

12.41 What are FTP passive connections?

by Colin Campbell

Ftp uses two data streams, one for passing commands around, the other for moving data. The command channel is handled by the ftpd listening on port 21.

The data channel varies depending on whether you ask for passive ftp or not. When you request data in a non-passive environment, you client tells the server "I am listening on <ip-address> <port>." The server then connects FROM port 20 to the ip address and port specied by your client. This requires your "security device" to permit any host outside from port 20 to any host inside on any port > 1023. Somewhat of a hole.

In passive mode, when you request a data transfer, the server tells the client "I am listening on <ip address> <port>." Your client then connects to the server on that IP and port and data ows.

13 Multicast

13.1 What is Multicast?

Multicast is essentially the ability to send one IP packet to multiple receivers. Multicast is often used for audio and video conferencing systems.

13.2 How do I know if my network has multicast?

One way is to ask someone who manages your network. If your network manager doesn't know, or looks at you funny, then you probably don't have it.

Another way is to use the *mtrace* program, which can be found on the *Xerox PARC FTP site* <ftp://parcftp.xerox.com/pub/net-research/ipmulti/>. Mtrace is similar to traceroute. It will tell you about the multicast path between your site and another. For example:

```
> mtrace mbone.ucar.edu
mtrace: WARNING: no multicast group specified, so no statistics printed
Mtrace from 128.117.64.29 to 192.172.226.25 via group 224.2.0.1
Querying full reverse path... * switching to hop-by-hop:
0  oceana-ether.nlanr.net (192.172.226.25)
-1  avidya-ether.nlanr.net (192.172.226.57)  DVMRP  thresh^ 1
-2  mbone.sdsc.edu (198.17.46.39)  DVMRP  thresh^ 1
-3  * nccosc-mbone.dren.net (138.18.5.224)  DVMRP  thresh^ 48
-4  * * FIXW-MBONE.NSN.NASA.GOV (192.203.230.243)  PIM/Special  thresh^ 64
-5  dec3800-2-fddi-0.SanFrancisco.mci.net (204.70.158.61)  DVMRP  thresh^ 64
-6  dec3800-2-fddi-0.Denver.mci.net (204.70.152.61)  DVMRP  thresh^ 1
-7  mbone.ucar.edu (192.52.106.7)  DVMRP  thresh^ 64
-8  mbone.ucar.edu (128.117.64.29)
Round trip time 196 ms; total ttl of 68 required.
```

13.3 Should I be using Multicast ICP?

Short answer: No, probably not.

Reasons why you SHOULD use Multicast:

1. It reduces the number of times Squid calls *sendto()* to put a UDP packet onto the network.
2. Its trendy and cool to use Multicast.

Reasons why you SHOULD NOT use Multicast:

1. Multicast tunnels/congurations/infrastructure are often unstable. You may lose multicast connectivity but still have unicast connectivity.
2. Multicast does not simplify your Squid conguration le. Every trusted neighbor cache must still be specied.
3. Multicast does not reduce the number of ICP replies being sent around. It does reduce the number of ICP queries sent, but not the number of replies.
4. Multicast exposes your cache to some privacy issues. There are no special emissions required to join a multicast group. Anyone may join your group and eavesdrop on ICP query messages. However, the scope of your multicast trac can be controlled such that it does not exceed certain boundaries.

We only recommend people to use Multicast ICP over network infrastructure which they have close control over. In other words, only use Multicast over your local area network, or maybe your wide area network if you are an ISP. We think it is probably a bad idea to use Multicast ICP over congested links or commodity backbones.

13.4 How do I congure Squid to send Multicast ICP queries?

To congure Squid to send ICP queries to a Multicast address, you need to create another neighbour cache entry specied as *multicast*. For example:

```
cache_host 224.9.9.9 multicast 3128 3130 ttl=64
```

224.9.9.9 is a sample multicast group address. *multicast* indicates that this is a special type of neighbour. The HTTP-port argument (3128) is ignored for multicast peers, but the ICP-port (3130) is very important. The nal argument, *ttl=64* species the multicast TTL value for queries sent to this address. It is probably a good idea to increment the minimum TTL by a few to provide a margin for error and changing conditions.

You must also specify which of your neighbours will respond to your multicast queries, since it would be a bad idea to implicitly trust any ICP reply from an unknown address. Note that ICP replies are sent back to *unicast* addresses; they are NOT multicast, so Squid has no indication whether a reply is from a regular query or a multicast query. To congure your multicast group neighbours, use the *cache_host* directive and the *multicast-responder* option:

```
cache_host cache1 sibling 3128 3130 multicast-responder
cache_host cache2 sibling 3128 3130 multicast-responder
```

Here all elds are relevant. The ICP port number (3130) must be the same as in the *cache_host* line dening the multicast peer above. The third eld must either be *parent* or *sibling* to indicate how Squid should treat replies. With the *multicast-responder* ag set for a peer, Squid will NOT send ICP queries to it directly (i.e. unicast).

13.5 How do I know what Multicast TTL to use?

The Multicast TTL (which is specied on the *cache_host* line of your multicast group) determines how "far" your ICP queries will go. In the Mbone, there is a certain TTL threshold dened for each network interface or tunnel. A multicast packet's TTL must be larger than the dened TTL for that packet to be forwarded across that link. For example, the *mrouted* manual page recommends:

```
32   for links that separate sites within an organization.
64   for links that separate communities or organizations, and are
     attached to the Internet MBONE.
128  for links that separate continents on the MBONE.
```

A good way to determine the TTL you need is to run *mtrace* as shown above and look at the last line. It will show you the minimum TTL required to reach the other host.

If you set you TTL too high, then your ICP messages may travel "too far" and will be subject to eavesdropping by others. If you're only using multicast on your LAN, as we suggest, then your TTL will be quite small, for example *ttl=4*.

13.6 How do I congure Squid to receive and respond to Multicast ICP?

You must tell Squid to join a multicast group address with the *mcast_groups* directive. For example:

```
mcast_groups  224.9.9.9
```

Of course, all members of your Multicast ICP group will need to use the exact same multicast group address.

NOTE: Choose a multicast group address with care! If two organizations happen to choose the same multicast address, then they may nd that their groups "overlap" at some point. This will be especially true if one of the querying caches uses a large TTL value. There are two ways to reduce the risk of group overlap:

1. Use a unique group address
2. Limit the scope of multicast messages with TTLs or administrative scoping.

Using a unique address is a good idea, but not without some potential problems. If you choose an address randomly, how do you know that someone else will not also randomly choose the same address? NLANR has been assigned a block of multicast addresses by the IANA for use in situations such as this. If you would like to be assigned one of these addresses, please *write to us* <mailto:nlanr-cache@nlanr.net>. However, note that NLANR or IANA have no authority to prevent anyone from using an address assigned to you.

Limiting the scope of your multicast messages is probably a better solution. They can be limited with the TTL value discussed above, or with some newer techniques known as administratively scoped addresses. Here you can congure well-dened boundaries for the trac to a specic address. The *Administratively Scoped IP Multicast RFC* <ftp://ftp.isi.edu/in-notes/rfc2365.txt> describes this.

14 System-Dependent Weirdnesses

14.1 Solaris

14.1.1 TCP incompatibility?

J.D. Bronson (jb at ktxg dot com) reported that his Solaris box could not talk to certain origin servers, such as *moneycentral.msn.com* <http://moneycentral.msn.com/> and *www.mbnanetaccess.com* <http://www.mbnanetaccess.com>. J.D. xed his problem by setting:

```
tcp_xmit_hiwat 49152
tcp_xmit_lowat 4096
tcp_recv_hiwat 49152
```

14.1.2 select()

select(3c) won't handle more than 1024 le descriptors. The *congure* script should enable *poll()* by default for Solaris. *poll()* allows you to use many more ledescriptors, probably 8192 or more.

For older Squid versions you can enable *poll()* manually by changing HAVE_POLL in *include/autoconf.h*, or by adding -DUSE_POLL=1 to the DEFINES in src/Makele.

14.1.3 malloc

libmalloc.a is leaky. Squid's congure does not use -lmalloc on Solaris.

14.1.4 DNS lookups and *nscd*

by *David J N Begley* <mailto:david@avarice.nepean.uws.edu.au>.

DNS lookups can be slow because of some mysterious thing called **ncsd**. You should edit */etc/nscd.conf* and make it say:

```
enable-cache            hosts           no
```

Apparently nscd serializes DNS queries thus slowing everything down when an application (such as Squid) hits the resolver hard. You may notice something similar if you run a log processor executing many DNS resolver queries - the resolver starts to slow.. right.. down.. . . .

According to *Andres Kroonmaa* <mailto:andre at online dot ee>, users of Solaris starting from version 2.6 and up should NOT completely disable *nscd* daemon. *nscd* should be running and caching passwd and group les, although it is suggested to disable hosts caching as it may interfere with DNS lookups.

Several library calls rely on available free FILE descriptors FD < 256. Systems running without nscd may fail on such calls if rst 256 les are all in use.

Since solaris 2.6 Sun has changed the way some system calls work and is using *nscd* daemon as a implementor of them. To communicate to *nscd* Solaris is using undocumented door calls. Basically *nscd* is used to reduce memory usage of user-space system libraries that use passwd and group les. Before 2.6 Solaris cached full passwd le in library memory on the rst use but as this was considered to use up too much ram on large multiuser systems Sun has decided to move implementation of these calls out of libraries and to a single dedicated daemon.

14.1.5 DNS lookups and */etc/nsswitch.conf*

by *Jason Armistead* <mailto:ARMISTEJ@oeca.otis.com>.

The */etc/nsswitch.conf* le determines the order of searches for lookups (amongst other things). You might only have it set up to allow NIS and HOSTS les to work. You denitely want the "hosts:" line to include the word *dns*, e.g.:

```
hosts:      nis dns [NOTFOUND=return] files
```

14.1.6 DNS lookups and NIS

by *Chris Tilbury* <mailto:cudch@csv.warwick.ac.uk>.

Our site cache is running on a Solaris 2.6 machine. We use NIS to distribute authentication and local hosts information around and in common with our multiuser systems, we run a slave NIS server on it to help the response of NIS queries.

We were seeing very high name-ip lookup times (avg ˜2sec) and ip->name lookup times (avg ˜8 sec), although there didn't seem to be that much of a problem with response times for valid sites until the cache was being placed under high load. Then, performance went down the toilet.

After some time, and a bit of detective work, we found the problem. On Solaris 2.6, if you have a local NIS server running (*ypserv*) and you have NIS in your */etc/nsswitch.conf* hosts entry, then check the ags it is being started with. The 2.6 ypstart script checks to see if there is a *resolv.conf* le present when it starts ypserv. If there is, then it starts it with the *-d* option.

This has the same eect as putting the *YP_INTERDOMAIN* key in the hosts table – namely, that failed NIS host lookups are tried against the DNS by the NIS server.

This is a **bad thing(tm)**! If NIS itself tries to resolve names using the DNS, then the requests are serialised through the NIS server, creating a bottleneck (This is the same basic problem that is seen with *nscd*). Thus, one failing or slow lookup can, if you have NIS before DNS in the service switch le (which is the most common setup), hold up every other lookup taking place.

If you're running in this kind of setup, then you will want to make sure that

1. ypserv doesn't start with the *-d* ag.
2. you don't have the *YP_INTERDOMAIN* key in the hosts table (nd the *B=-b* line in the yp Makele and change it to *B=*)

We changed these here, and saw our average lookup times drop by up to an order of magnitude (˜150msec for name-ip queries and ˜1.5sec for ip-name queries, the latter still so high, I suspect, because more of these fail and timeout since they are not made so often and the entries are frequently non-existent anyway).

14.1.7 Tuning

Solaris 2.x - tuning your TCP/IP stack and more <http://www.rvs.uni-hannover.de/people/voeckler/tune/EN/tune.ht
by *Jens-S. Vckler* <http://www.rvs.uni-hannover.de/people/voeckler/>

14.1.8 disk write error: (28) No space left on device

You might get this error even if your disk is not full, and is not out of inodes. Check your syslog logs (/var/adm/messages, normally) for messages like either of these:

```
NOTICE: realloccg /proxy/cache: file system full
NOTICE: alloc: /proxy/cache: file system full
```

In a nutshell, the UFS lesystem used by Solaris can't cope with the workload squid presents to it very well. The lesystem will end up becoming highly fragmented, until it reaches a point where there are insucient free blocks left to create les with, and only fragments available. At this point, you'll get this error and squid will revise its idea of how much space is actually available to it. You can do a "fsck -n raw_device" (no need to unmount, this checks in read only mode) to look at the fragmentation level of the lesystem. It will probably be quite high (>15%).

Sun suggest two solutions to this problem. One costs money, the other is free but may result in a loss of performance (although Sun do claim it shouldn't, given the already highly random nature of squid disk access).

The rst is to buy a copy of VxFS, the Veritas Filesystem. This is an extent-based lesystem and it's capable of having online defragmentation performed on mounted lesystems. This costs money, however (VxFS is not very cheap!)

The second is to change certain parameters of the UFS lesystem. Unmount your cache lesystems and use tunefs to change optimization to "space" and to reduce the "minfree" value to 3-5% (under Solaris 2.6 and higher, very large lesystems will almost certainly have a minfree of 2% already and you shouldn't increase this). You should be able to get fragmentation down to around 3% by doing this, with an accompanied increase in the amount of space available.

Thanks to *Chris Tilbury* <mailto:cudch@csv.warwick.ac.uk>.

14.1.9 Solaris X86 and IPFilter

by *Je Madison* <mailto:jeff@sisna.com>

Important update regarding Squid running on Solaris x86. I have been working for several months to resolve what appeared to be a memory leak in squid when running on Solaris x86 regardless of the malloc that was used. I have made 2 discoveries that anyone running Squid on this platform may be interested in.

Number 1: There is not a memory leak in Squid even though after the system runs for some amount of time, this varies depending on the load the system is under, Top reports that there is very little memory free. True to the claims of the Sun engineer I spoke to this statistic from Top is incorrect. The odd thing is that you do begin to see performance suer substantially as time goes on and the only way to correct the situation is to reboot the system. This leads me to discovery number 2.

Number 2: There is some type of resource problem, memory or other, with IPFilter on Solaris x86. I have not taken the time to investigate what the problem is because we no longer are using IPFilter. We have switched to a Alteon ACE 180 Gigabit switch which will do the trans-proxy for you. After moving the trans-proxy, redirection process out to the Alteon switch Squid has run for 3 days strait under a huge load with no problem what so ever. We currently have 2 boxes with 40 GB of cached objects on each box. This 40 GB was accumulated in the 3 days, from this you can see what type of load these boxes are under. Prior to this change we were never able to operate for more than 4 hours.

Because the problem appears to be with IPFilter I would guess that you would only run into this issue if you are trying to run Squid as a interception proxy using IPFilter. That makes sense. If there is anyone with information that would indicate my nding are incorrect I am willing to investigate further.

14.1.10 Changing the directory lookup cache size

by *Mike Batchelor* <mailto:mbatchelor@citysearch.com>

On Solaris, the kernel variable for the directory name lookup cache size is *ncsize*. In */etc/system*, you might want to try

```
set ncsize = 8192
```

or even higher. The kernel variable *ufs_inode* - which is the size of the inode cache itself - scales with *ncsize* in Solaris 2.5.1 and later. Previous versions of Solaris required both to be adjusted independently, but now, it is not recommended to adjust *ufs_inode* directly on 2.5.1 and later.

You can set *ncsize* quite high, but at some point - dependent on the application - a too-large *ncsize* will increase the latency of lookups.

Defaults are:

```
Solaris 2.5.1 : (max_nprocs + 16 + maxusers) + 64
Solaris 2.6/Solaris 7 : 4 * (max_nprocs + maxusers) + 320
```

14.1.11 The priority_paging algorithm

by *Mike Batchelor* <mailto:mbatchelor@citysearch.com>

Another new tuneable (actually a toggle) in Solaris 2.5.1, 2.6 or Solaris 7 is the *priority_paging* algorithm. This is actually a complete rewrite of the virtual memory system on Solaris. It will page out application data last, and lesystem pages rst, if you turn it on (set *priority_paging* = 1 in */etc/system*). As you may know, the Solaris buer cache grows to ll available pages, and under the old VM system, applications could get paged out to make way for the buer cache, which can lead to swap thrashing and degraded application performance. The new *priority_paging* helps keep application and shared library pages in memory, preventing the buer cache from paging them out, until memory gets REALLY short. Solaris 2.5.1 requires patch 103640-25 or higher and Solaris 2.6 requires 105181-10 or higher to get priority_paging. Solaris 7 needs no patch, but all versions have it turned o by default.

14.2 FreeBSD

14.2.1 T/TCP bugs

We have found that with FreeBSD-2.2.2-RELEASE, there some bugs with T/TCP. FreeBSD will try to use T/TCP if you've enabled the "TCP Extensions." To disable T/TCP, use *sysinstall* to disable TCP Extensions, or edit */etc/rc.conf* and set

```
        tcp_extensions="NO"             # Allow RFC1323 & RFC1544 extensions (or NO).
```

or add this to your /etc/rc les:

```
        sysctl -w net.inet.tcp.rfc1644=0
```

14.2.2 mbuf size

We noticed an odd thing with some of Squid's interprocess communication. Often, output from the *dnsserver* processes would NOT be read in one chunk. With full debugging, it looks like this:

```
1998/04/02 15:18:48| comm_select: FD 46 ready for reading
1998/04/02 15:18:48| ipcache_dnsHandleRead: Result from DNS ID 2 (100 bytes)
1998/04/02 15:18:48| ipcache_dnsHandleRead: Incomplete reply
....other processing occurs...
1998/04/02 15:18:48| comm_select: FD 46 ready for reading
1998/04/02 15:18:48| ipcache_dnsHandleRead: Result from DNS ID 2 (9 bytes)
1998/04/02 15:18:48| ipcache_parsebuffer: parsing:
$name www.karup.com
$h_name www.karup.inter.net
$h_len 4
$ipcount 2
38.15.68.128
38.15.67.128
$ttl 2348
$end
```

Interestingly, it is very common to get only 100 bytes on the rst read. When two read() calls are required, this adds additional latency to the overall request. On our caches running Digital Unix, the median *dnsserver* response time was measured at 0.01 seconds. On our FreeBSD cache, however, the median latency was 0.10 seconds.

Here is a simple patch to x the bug:

```
===================================================================
RCS file: /home/ncvs/src/sys/kern/uipc_socket.c,v
retrieving revision 1.40
retrieving revision 1.41
diff -p -u -r1.40 -r1.41
--- src/sys/kern/uipc_socket.c  1998/05/15 20:11:30     1.40
+++ /home/ncvs/src/sys/kern/uipc_socket.c       1998/07/06 19:27:14     1.41
@@ -31,7 +31,7 @@
  * SUCH DAMAGE.
  *
  *     @(#)uipc_socket.c       8.3 (Berkeley) 4/15/94
- *     $Id: FAQ.sgml,v 1.193 2003/08/26 16:50:13 wessels Exp $
+ *     $Id: FAQ.sgml,v 1.193 2003/08/26 16:50:13 wessels Exp $
  */

 #include <sys/param.h>
@@ -491,6 +491,7 @@ restart:
```

```
                                mlen = MCLBYTES;
                                len = min(min(mlen, resid), space);
                        } else {
+                               atomic = 1;
 nopages:
                                len = min(min(mlen, resid), space);
                                /*
```

Another technique which may help, but does not x the bug, is to increase the kernel's mbuf size. The default is 128 bytes. The MSIZE symbol is dened in */usr/include/machine/param.h*. However, to change it we added this line to our kernel conguration le:

```
options         MSIZE="256"
```

14.2.3 Dealing with NIS

/var/yp/Makele has the following section:

```
# The following line encodes the YP_INTERDOMAIN key into the hosts.byname
# and hosts.byaddr maps so that ypserv(8) will do DNS lookups to resolve
# hosts not in the current domain. Commenting this line out will disable
# the DNS lookups.
B=-b
```

You will want to comment out the *B=-b* line so that *ypserv* does not do DNS lookups.

14.2.4 FreeBSD 3.3: The lo0 (loop-back) device is not congured on startup

Squid requires a the loopback interface to be up and congured. If it is not, you will get errors such as 11.37.

From *FreeBSD 3.3 Errata Notes* <http://www.freebsd.org/releases/3.3R/errata.html>:

> Fix: Assuming that you experience this problem at all, edit */etc/rc.conf* and search for where the network_interfaces variable is set. In its value, change the word *auto* to *lo0* since the auto keyword doesn't bring the loop-back device up properly, for reasons yet to be adequately determined. Since your other interface(s) will already be set in the network_interfaces variable after initial installation, it's reasonable to simply s/auto/lo0/ in rc.conf and move on.

Thanks to *Robert Lister* <mailto:robl at lentil dot org>.

14.2.5 FreeBSD 3.x or newer: Speed up disk writes using Softupdates

by *Andre Albsmeier* <mailto:andre.albsmeier@mchp.siemens.de>

FreeBSD 3.x and newer support Softupdates. This is a mechanism to speed up disk writes as it is possible by mounting ufs volumes async. However, Softupdates does this in a way that a performance similar or better than async is achieved but without loosing security in a case of a system crash. For more detailed information and the copyright terms see */sys/contrib/softupdates/README* and */sys/ufs/s/README.softupdate* .

To build a system supporting softupdates, you have to build a kernel with `options SOFTUPDATES` set (see *LINT* for a commented out example). After rebooting with the new kernel, you can enable softupdates on a per lesystem base with the command:

```
$ tunefs -n /mountpoint
```

The lesystem in question MUST NOT be mounted at this time. After that, softupdates are permanently enabled and the lesystem can be mounted normally. To verify that the softupdates code is running, simply issue a mount command and an output similar to the following will appear:

```
$ mount
/dev/da2a on /usr/local/squid/cache (ufs, local, noatime, soft-updates, writes: sync 70 async 2
```

14.2.6 Internal DNS problems with jail environment

Some users report problems with running Squid in the jail environment. Specically, Squid logs messages like:

```
2001/10/12 02:08:49| comm_udp_sendto: FD 4, 192.168.1.3, port 53: (22) Invalid argument
2001/10/12 02:08:49| idnsSendQuery: FD 4: sendto: (22) Invalid argument
```

You can eliminate the problem by putting the jail's network interface address in the 'udp_outgoing_addr' conguration option in *squid.conf*.

14.3 OSF1/3.2

If you compile both libgnumalloc.a and Squid with *cc*, the *mstats()* function returns bogus values. However, if you compile libgnumalloc.a with *gcc*, and Squid with *cc*, the values are correct.

14.4 BSD/OS

14.4.1 gcc/yacc

Some people report 2.10.

14.4.2 process priority

I've noticed that my Squid process seems to stick at a nice value of four, and clicks back to that even after I renice it to a higher priority. However, looking through the Squid source, I can't nd any instance of a setpriority() call, or anything else that would seem to indicate Squid's adjusting its own priority.

by *Bill Bogstad* <mailto:bogstad@pobox.com>

BSD Unices traditionally have auto-niced non-root processes to 4 after they used alot (4 minutes???) of CPU time. My guess is that it's the BSD/OS not Squid that is doing this. I don't know ohand if there is a way to disable this on BSD/OS.

by *Arjan de Vet* <mailto:Arjan.deVet@adv.iae.nl>

You can get around this by starting Squid with nice-level -4 (or another negative value).

by *Bert Driehuis* <mailto:bert_driehuis at nl dot compuware dot com>

The autonice behavior is a leftover from the history of BSD as a university OS. It penalises CPU bound jobs by nicing them after using 600 CPU seconds. Adding

```
sysctl -w kern.autonicetime=0
```

to */etc/rc.local* will disable the behavior systemwide.

14.5 Linux

14.5.1 Cannot bind socket FD 5 to 127.0.0.1:0: (49) Can't assign requested address

Try a dierent version of Linux. We have received many reports of this "bug" from people running Linux 2.0.30. The *bind(2)* system call should NEVER give this error when binding to port 0.

14.5.2 FATAL: Don't run Squid as root, set 'cache_eective _user'!

Some users have reported that setting `cache_effective_user` to `nobody` under Linux does not work. However, it appears that using any `cache_effective_user` other than `nobody` will succeed. One solution is to create a user account for Squid and set `cache_effective_user` to that. Alternately you can change the UID for the `nobody` account from 65535 to 65534.

Another problem is that RedHat 5.0 Linux seems to have a broken *setresuid()* function. There are two ways to x this. Before running congure:

```
% setenv ac_cv_func_setresuid no
% ./configure ...
% make clean
% make install
```

Or after running congure, manually edit include/autoconf.h and change the HAVE _SETRESUID line to:

```
#define HAVE_SETRESUID 0
```

Also, some users report this error is due to a NIS conguration problem. By adding *compat* to the *passwd* and *group* lines of */etc/nsswitch.conf*, the problem goes away. (*Ambrose Li* <mailto:acli@ada.ddns.org>).

Russ Mellon <mailto:galifrey@crown.net> notes that these problems with *cache_eective _user* are xed in version 2.2.x of the Linux kernel.

14.5.3 Large ACL lists make Squid slow

The regular expression library which comes with Linux is known to be very slow. Some people report it entirely fails to work after long periods of time.

To x, use the GNUregex library included with the Squid source code. With Squid-2, use the *–enable-gnuregex* congure option.

14.5.4 gethostbyname() leaks memory in RedHat 6.0 with glibc 2.1.1.

by *Radu Greab* <mailto:radu at netsoft dot ro>

The gethostbyname() function leaks memory in RedHat 6.0 with glibc 2.1.1. The quick x is to delete nisplus service from hosts entry in */etc/nsswitch.conf*. In my tests dnsserver memory use remained stable after I made the above change.

See *RedHat bug id 3919* <http://developer.redhat.com/bugzilla/show_bug.cgi?id=3919>.

14.5.5 assertion failed: StatHist.c:91: 'statHistBin(H, max) == H->capacity - 1' on Alpha system.

by *Jamie Raymond* <mailto:jraymond@gnu.org>

Some early versions of Linux have a kernel bug that causes this. All that is needed is a recent kernel that doesn't have the mentioned bug.

14.5.6 tools.c:605: storage size of 'rl' isn't known

This is a bug with some versions of glibc. The glibc headers incorrectly depended on the contents of some kernel headers. Everything broke down when the kernel folks rearranged a bit in the kernel-specic header les.

We think this glibc bug is present in versions 2.1.1 (or 2.1.0) and earlier. There are two solutions:

1. Make sure /usr/include/linux and /usr/include/asm are from the kernel version glibc is build/congured for, not any other kernel version. Only compiling of loadable kernel modules outside of the kernel sources depends on having the current versions of these, and for such builds -I/usr/src/linux/include (or where ever the new kernel headers are located) can be used to resolve the matter.

2. Upgrade glibc to 2.1.2 or later. This is always a good idea anyway, provided a prebuilt upgrade package exists for the Linux distribution used.. Note: Do not attempt to manually build and install glibc from source unless you know exactly what you are doing, as this can easily render the system unuseable.

14.5.7 Can't connect to some sites through Squid

When using Squid, some sites may give erorrs such as "(111) Connection refused" or "(110) Connection timed out" although these sites work ne without going through Squid.

Some versions of linux implement *Explicit Congestion Notication* `<http://www.aciri.org/floyd/ecn.html>` (ECN) and this can cause some TCP connections to fail when contacting some sites with broken rewalls or broken TCP/IP implementations. A list of sites to be broken can be found at *ECN Hall of Shame* `<http://urchin.earth.li/ecn/>`.

To work around such broken sites you can disable ECN with the following command:

```
echo 0 > /proc/sys/net/ipv4/tcp_ecn
```

Found this on the FreeBSD mailing list:

> From: Robert Watson
> As Bill Fumerola has indicated, and I thought I'd follow up in with a bit more detail, the behavior you're seeing is the result of a bug in the FreeBSD IPFW code. FreeBSD did a direct comparison of the TCP header ag eld with an internal eld in the IPFW rule description structure. Unfortunately, at some point, someone decided to overload the IPFW rule description structure eld to add a ag representing "ESTABLISHED". They used a ag value that was previously unused by the TCP protocol (which doesn't make it safer, just less noticeable). Later, when that ag was allocated for ECN (Endpoint Congestion Notication) in TCP, and Linux began using ECN by default, the packets began to match ESTABLISHED rules regardless of the other TCP header ags. This bug was corrected on the RELENG_4 branch, and security advisory for the bug was released. This was, needless to say, a pretty serious bug, and good example of why you should be very careful to compare only the bits you really mean to, and should seperate packet state from protocol state in management structures, as well as make use of extensive testing to make sure rules actually have the eect you describe.

See also the *thread on the NANOG mailing list* <http://answerpointe.cctec.com/maillists/nanog/historical/0104/ms
RFC3168 "The Addition of Explicit Congestion Notication (ECN) to IP, PROPOSED STANDARD " <ftp://ftp.isi.edu/in-notes/rfc3168.txt> , *Sally Floyd's page on ECN and problems related to it* <http://www.aciri.org/floyd/ecn.html> or *ECN Hall of Shame* <http://urchin.earth.li/ecn/> for more information.

14.6 HP-UX

14.6.1 StatHist.c:74: failed assertion 'statHistBin(H, min) == 0'

This was a very mysterious and unexplainable bug with GCC on HP-UX. Certain functions, when specied as *static*, would cause math bugs. The compiler also failed to handle implied int-double conversions properly. These bugs should all be handled correctly in Squid version 2.2.

14.7 IRIX

14.7.1 *dnsserver* always returns 255.255.255.255

There is a problem with GCC (2.8.1 at least) on Irix 6 which causes it to always return the string 255.255.255.255 for _ANY_ address when calling inet_ntoa(). If this happens to you, compile Squid with the native C compiler instead of GCC.

14.8 SCO-UNIX

by *F.J. Bosscha* <mailto:f.j.bosscha@nhl.nl>

To make squid run comfortable on SCO-unix you need to do the following:

Increase the *NOFILES* paramater and the *NUMSP* parameter and compile squid with I had, although squid told in the cache.log le he had 3000 ledescriptors, problems with the messages that there were no ledescriptors more available. After I increase also the NUMSP value the problems were gone.

One thing left is the number of tcp-connections the system can handle. Default is 256, but I increase that as well because of the number of clients we have.

14.9 AIX

14.9.1 "shmat failed" errors with *diskd*

32-bit processes on AIX and later are restricted by default to a maximum of 11 shared memory segments. This restriction can be removed on AIX 4.2.1 and later by setting the environment variable `EXTSHM=ON` in the script or shell which starts squid.

14.9.2 Core dumps when squid process grows to 256MB

32-bit processes cannot use more than 256MB of stack and data in the default memory model. To force the loader to use large address space for squid, either:

set the `LDR_CNTRL` environment variable, eg `LDR_CNTRL="MAXDATA=0x80000000"`; or

link with `-bmaxdata:0x80000000`; or

patch the squid binary

See *IBM's documentation* <http://publibn.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixprggd/genprogc/lrg_prg_s on large program support for more information, including how to patch an already-compiled program.

15 Redirectors

15.1 What is a redirector?

Squid has the ability to rewrite requested URLs. Implemented as an external process (similar to a dnsserver), Squid can be congured to pass every incoming URL through a *redirector* process that returns either a new URL, or a blank line to indicate no change.

The *redirector* program is **NOT** a standard part of the Squid package. However, some examples are provided below, and in the "contrib/" directory of the source distribution. Since everyone has dierent needs, it is up to the individual administrators to write their own implementation.

15.2 Why use a redirector?

A redirector allows the administrator to control the locations to which his users goto. Using this in conjunction with interception proxies allows simple but eective porn control.

15.3 How does it work?

The redirector program must read URLs (one per line) on standard input, and write rewritten URLs or blank lines on standard output. Note that the redirector program can not use buered I/O. Squid writes additional information after the URL which a redirector can use to make a decision. The input line consists of four elds:

```
URL ip-address/fqdn ident method
```

15.4 Do you have any examples?

A simple very fast redirector called *SQUIRM* <http://squirm.foote.com.au/> is a good place to start, it uses the regex lib to allow pattern matching.

Also see *jesred* <http://ivs.cs.uni-magdeburg.de/%7eelkner/webtools/jesred/>.

The following Perl script may also be used as a template for writing your own redirector:

```
#!/usr/local/bin/perl
$|=1;
while (<>) {
        s@http://fromhost.com@http://tohost.org@;
        print;
}
```

15.5 Can I use the redirector to return HTTP redirect messages?

Normally, the *redirector* feature is used to rewrite requested URLs. Squid then transparently requests the new URL. However, in some situations, it may be desirable to return an HTTP "301" or "302" redirect message to the client. This is now possible with Squid version 1.1.19.

Simply modify your redirector program to prepend either "301:" or "302:" before the new URL. For example, the following script might be used to direct external clients to a secure Web server for internal documents:

```
#!/usr/local/bin/perl
$|=1;
        while (<>) {
                @X = split;
                $url = $X[0];
                if ($url =~ /^http:\/\/internal\.foo\.com/) {
                        $url =~ s/^http/https/;
                        $url =~ s/internal/secure/;
                        print "302:$url\n";
                } else {
                        print "$url\n";
                }
        }
```

Please see sections 10.3.2 and 10.3.3 of *RFC 2068* <ftp://ftp.isi.edu/in-notes/rfc2068.txt> for an explanation of the 301 and 302 HTTP reply codes.

15.6 FATAL: All redirectors have exited!

A redirector process must exit (stop running) only when its *stdin* is closed. If you see the "All redirectories have exited" message, it probably means your redirector program has a bug. Maybe it runs out of memory or has memory access errors. You may want to test your redirector program outside of squid with a big input list, taken from your *access.log* perhaps. Also, check for 11.19.1 les from the redirector program.

15.7 Redirector interface is broken re IDENT values

I added a redirctor consisting of

```
#! /bin/sh
/usr/bin/tee /tmp/squid.log
```

and many of the redirector requests don't have a username in the ident eld.

Squid does not delay a request to wait for an ident lookup, unless you use the ident ACLs. Thus, it is very likely that the ident was not available at the time of calling the redirector, but became available by the time the request is complete and logged to access.log.

If you want to block requests waiting for ident lookup, try something like this:

```
acl foo ident REQUIRED
http_access allow foo
```

16 Cache Digests

Cache Digest FAQs compiled by Niall Doherty <mailto:ndoherty@eei.ericsson.se>.

16.1 What is a Cache Digest?

A Cache Digest is a summary of the contents of an Internet Object Caching Server. It contains, in a compact (i.e. compressed) format, an indication of whether or not particular URLs are in the cache.

A "lossy" technique is used for compression, which means that very high compression factors can be achieved at the expense of not having 100% correct information.

16.2 How and why are they used?

Cache servers periodically exchange their digests with each other.

When a request for an object (URL) is received from a client a cache can use digests from its peers to nd out which of its peers (if any) have that object. The cache can then request the object from the closest peer (Squid uses the NetDB database to determine this).

Note that Squid will only make digest queries in those digests that are *enabled*. It will disable a peers digest IFF it cannot fetch a valid digest for that peer. It will enable that peers digest again when a valid one is fetched.

The checks in the digest are very fast and they eliminate the need for per-request queries to peers. Hence:

Latency is eliminated and client response time should be improved.

Network utilisation may be improved.

Note that the use of Cache Digests (for querying the cache contents of peers) and the generation of a Cache Digest (for retrieval by peers) are independent. So, it is possible for a cache to make a digest available for peers, and not use the functionality itself and vice versa.

16.3 What is the theory behind Cache Digests?

Cache Digests are based on Bloom Filters - they are a method for representing a set of keys with lookup capabilities; where lookup means "is the key in the lter or not?".

In building a cache digest:

A vector (1-dimensional array) of m bits is allocated, with all bits initially set to 0.

A number, k, of independent hash functions are chosen, h1, h2, ..., hk, with range { 1, ..., m } (i.e. a key hashed with any of these functions gives a value between 1 and m inclusive).

The set of n keys to be operated on are denoted by: A = { a1, a2, a3, ..., an }.

16.3.1 Adding a Key

To add a key the value of each hash function for that key is calculated. So, if the key was denoted by a, then h1(a), h2(a), ..., hk(a) are calculated.

The value of each hash function for that key represents an index into the array and the corresponding bits are set to 1. So, a digest with 6 hash functions would have 6 bits to be set to 1 for each key added.

Note that the addition of a number of *dierent* keys could cause one particular bit to be set to 1 multiple times.

16.3.2 Querying a Key

To query for the existence of a key the indices into the array are calculated from the hash functions as above.

If any of the corresponding bits in the array are 0 then the key is not present.

If all of the corresponding bits in the array are 1 then the key is *likely* to be present.

Note the term *likely*. It is possible that a *collision* in the digest can occur, whereby the digest incorrectly indicates a key is present. This is the price paid for the compact representation. While the probability of a collision can never be reduced to zero it can be controlled. Larger values for the ratio of the digest size to the number of entries added lower the probability. The number of hash functions chosen also inuence the probability.

16.3.3 Deleting a Key

To delete a key, it is not possible to simply set the associated bits to 0 since any one of those bits could have been set to 1 by the addition of a dierent key!

Therefore, to support deletions a counter is required for each bit position in the array. The procedures to follow would be:

When adding a key, set appropriate bits to 1 and increment the corresponding counters.

When deleting a key, decrement the appropriate counters (while > 0), and if a counter reaches 0 *then* the corresponding bit is set to 0.

16.4 How is the size of the Cache Digest in Squid determined?

Upon initialisation, the *capacity* is set to the number of objects that can be (are) stored in the cache. Note that there are upper and lower limits here.

An arbitrary constant, bits_per_entry (currently set to 5), is used to calculate the size of the array using the following formula:

```
number of bits in array = capacity * bits_per_entry + 7
```

The size of the digest, in bytes, is therefore:

```
digest size = int (number of bits in array / 8)
```

When a digest rebuild occurs, the change in the cache size (capacity) is measured. If the capacity has changed by a large enough amount (10%) then the digest array is freed and reallocated memory, otherwise the same digest is re-used.

16.5 What hash functions (and how many of them) does Squid use?

The protocol design allows for a variable number of hash functions (k). However, Squid employs a very ecient method using a xed number - four.

Rather than computing a number of independent hash functions over a URL Squid uses a 128-bit MD5 hash of the key (actually a combination of the URL and the HTTP retrieval method) and then splits this into four equal chunks.

Each chunk, modulo the digest size (m), is used as the value for one of the hash functions - i.e. an index into the bit array.

Note: As Squid retrieves objects and stores them in its cache on disk, it adds them to the in-RAM index using a lookup key which is an MD5 hash - the very one discussed above. This means that the values for the Cache Digest hash functions are already available and consequently the operations are extremely ecient!

Obviously, modifying the code to support a variable number of hash functions would prove a little more dicult and would most likely reduce eciency.

16.6 How are objects added to the Cache Digest in Squid?

Every object referenced in the index in RAM is checked to see if it is suitable for addition to the digest.

A number of objects are not suitable, e.g. those that are private, not cachable, negatively cached etc. and are skipped immediately.

A *freshness* test is next made in an attempt to guess if the object will expire soon, since if it does, it is not worthwhile adding it to the digest. The object is checked against the refresh patterns for staleness...

Since Squid stores references to objects in its index using the MD5 key discussed earlier there is no URL actually available for each object - which means that the pattern used will fall back to the default pattern, ".". This is an unfortunate state of aairs, but little can be done about it. A *cd_refresh_pattern* option will be added to the conguration le soon which will at least make the confusion a little clearer :-)

Note that it is best to be conservative with your refresh pattern for the Cache Digest, i.e. do *not* add objects if they might become stale soon. This will reduce the number of False Hits.

16.7 Does Squid support deletions in Cache Digests? What are dis/deltas?

Squid does not support deletions from the digest. Because of this the digest must, periodically, be rebuilt from scratch to erase stale bits and prevent digest pollution.

A more sophisticated option is to use *dis* or *deltas*. These would be created by building a new digest and comparing with the current/old one. They would essentially consist of aggregated deletions and additions since the *previous* digest.

Since less bandwidth should be required using these it would be possible to have more frequent updates (and hence, more accurate information).

Costs:

RAM - extra RAM needed to hold two digests while comparisons takes place.

CPU - probably a negligible amount.

16.8 When and how often is the local digest built?

The local digest is built:

when store_rebuild completes after startup (the cache contents have been indexed in RAM), and

periodically thereafter. Currently, it is rebuilt every hour (more data and experience is required before other periods, whether xed or dynamically varying, can "intelligently" be chosen). The good thing is that the local cache decides on the expiry time and peers must obey (see later).

While the [new] digest is being built in RAM the old version (stored on disk) is still valid, and will be returned to any peer requesting it. When the digest has completed building it is then swapped out to disk, overwriting the old version.

The rebuild is CPU intensive, but not overly so. Since Squid is programmed using an event-handling model, the approach taken is to split the digest building task into chunks (i.e. chunks of entries to add) and to register each chunk as an event. If CPU load is overly high, it is possible to extend the build period - as long as it is nished before the next rebuild is due!

It may prove more ecient to implement the digest building as a separate process/thread in the future...

16.9 How are Cache Digests transferred between peers?

Cache Digests are fetched from peers using the standard HTTP protocol (note that a *pull* rather than *push* technique is used).

After the rst access to a peer, a *peerDigestValidate* event is queued (this event decides if it is time to fetch a new version of a digest from a peer). The queuing delay depends on the number of peers already queued for validation - so that all digests from dierent peers are not fetched simultaneously.

A peer answering a request for its digest will specify an expiry time for that digest by using the HTTP *Expires* header. The requesting cache thus knows when it should request a fresh copy of that peers digest.

Note: requesting caches use an If-Modied-Since request in case the peer has not rebuilt its digest for some reason since the last time it was fetched.

16.10 How and where are Cache Digests stored?

16.10.1 Cache Digest built locally

Since the local digest is generated purely for the benet of its neighbours keeping it in RAM is not strictly required. However, it was decided to keep the local digest in RAM partly because of the following:

Approximately the same amount of memory will be (re-)allocated on every rebuild of the digest,

the memory requirements are probably quite small (when compared to other requirements of the cache server),

if ongoing updates of the digest are to be supported (e.g. additions/deletions) it will be necessary to perform these operations on a digest in RAM, and

if dis/deltas are to be supported the "old" digest would have to be swapped into RAM anyway for the comparisons.

When the digest is built in RAM, it is then swapped out to disk, where it is stored as a "normal" cache item - which is how peers request it.

16.10.2 Cache Digest fetched from peer

When a query from a client arrives, *fast lookups* are required to decide if a request should be made to a neighbour cache. It it therefore required to keep all peer digests in RAM.

Peer digests are also stored on disk for the following reasons:

Recovery - If stopped and restarted, peer digests can be reused from the local on-disk copy (they will soon be validated using an HTTP IMS request to the appropriate peers as discussed earlier), and

Sharing - peer digests are stored as normal objects in the cache. This allows them to be given to neighbour caches.

16.11 How are the Cache Digest statistics in the Cache Manager to be interpreted?

Cache Digest statistics can be seen from the Cache Manager or through the *client* utility. The following examples show how to use the *client* utility to request the list of possible operations from the localhost, local digest statistics from the localhost, refresh statistics from the localhost and local digest statistics from another cache, respectively.

```
./client mgr:menu

./client mgr:store_digest

./client mgr:refresh

./client -h peer mgr:store_digest
```

The available statistics provide a lot of useful debugging information. The refresh statistics include a section for Cache Digests which explains why items were added (or not) to the digest.

The following example shows local digest statistics for a 16GB cache in a corporate intranet environment (may be a useful reference for the discussion below).

```
store digest: size: 768000 bytes
         entries: count: 588327 capacity: 1228800 util: 48%
         deletion attempts: 0
         bits: per entry: 5 on: 1953311 capacity: 6144000 util: 32%
         bit-seq: count: 2664350 avg.len: 2.31
         added: 588327 rejected: 528703 ( 47.33 %) del-ed: 0
         collisions: on add: 0.23 % on rej: 0.23 %
```

entries:capacity is a measure of how many items "are likely" to be added to the digest. It represents the number of items that were in the local cache at the start of digest creation - however, upper and lower limits currently apply. This value is multiplied by *bits: per entry* (an arbitrary constant) to give *bits:capacity*, which is the size of the cache digest in bits. Dividing this by 8 will give *store digest: size* which is the size in bytes.

The number of items represented in the digest is given by *entries:count*. This should be equal to *added* minus *deletion attempts*.

Since (currently) no modications are made to the digest after the initial build (no additions are made and deletions are not supported) *deletion attempts* will always be 0 and *entries:count* should simply be equal to *added*.

entries:util is not really a signicant statistic. At most it gives a measure of how many of the items in the store were deemed suitable for entry into the cache compared to how many were "prepared" for.

rej shows how many objects were rejected. Objects will not be added for a number of reasons, the most common being refresh pattern settings. Remember that (currently) the default refresh pattern will be used for checking for entry here and also note that changing this pattern can signicantly aect the number of items added to the digest! Too relaxed and False Hits increase, too strict and False Misses increase. Remember also that at time of validation (on the peer) the "real" refresh pattern will be used - so it is wise to keep the default refresh pattern conservative.

bits: on indicates the number of bits in the digest that are set to 1. *bits: util* gives this gure as a percentage of the total number of bits in the digest. As we saw earlier, a gure of 50% represents the optimal trade-o. Values too high (say > 75%) would cause a larger number of collisions, and hence False Hits, while lower values mean the digest is under-utilised (using unnecessary RAM). Note that low values are normal for caches that are starting to ll up.

A bit sequence is an uninterrupted sequence of bits with the same value. *bit-seq: avg.len* gives some insight into the quality of the hash functions. Long values indicate problem, even if *bits:util* is 50% (> 3 = suspicious, > 10 = very suspicious).

16.12 What are False Hits and how should they be handled?

A False Hit occurs when a cache believes a peer has an object and asks the peer for it *but* the peer is not able to satisfy the request.

Expiring or stale objects on the peer are frequent causes of False Hits. At the time of the query actual refresh patterns are used on the peer and stale entries are marked for revalidation. However, revalidation is prohibited unless the peer is behaving as a parent, or *miss_access* is enabled. Thus, clients can receive error messages instead of revalidated objects!

The frequency of False Hits can be reduced but never eliminated completely, therefore there must be a robust way of handling them when they occur. The philosophy behind the design of Squid is to use lightweight techniques and optimise for the common case and robustly handle the unusual case (False Hits).

Squid will soon support the HTTP *only-if-cached* header. Requests for objects made to a peer will use this header and if the objects are not available, the peer can reply appropriately allowing Squid to recognise the situation. The following describes what Squid is aiming towards:

Cache Digests used to obtain good estimates of where a requested object is located in a Cache Hierarchy.

Persistent HTTP Connections between peers. There will be no TCP startup overhead and both latency and network load will be similar for ICP (i.e. fast).

HTTP False Hit Recognition using the *only-if-cached* HTTP header - allowing fall back to another peer or, if no other peers are available with the object, then going direct (or *through* a parent if behind a rewall).

16.13 How can Cache Digest related activity be traced/debugged?

16.13.1 Enabling Cache Digests

If you wish to use Cache Digests (available in Squid version 2) you need to add a *congure* option, so that the relevant code is compiled in:

```
./configure --enable-cache-digests ...
```

16.13.2 What do the access.log entries look like?

If a request is forwarded to a neighbour due a HIT in that neighbour's Cache Digest the hierarchy (9th) eld of the access.log le for the *local cache* will look like *CACHE_DIGEST_HIT/neighbour*. The Log Tag (4th eld) should obviously show a MISS.

On the peer cache the request should appear as a normal HTTP request from the rst cache.

16.13.3 What does a False Hit look like?

The easiest situation to analyse is when two caches (say A and B) are involved neither of which uses the other as a parent. In this case, a False Hit would show up as a CACHE_DIGEST_HIT on A and *NOT* as a TCP_HIT on B (or vice versa). If B does not fetch the object for A then the hierarchy eld will look like *NONE/-* (and A will have received an Access Denied or Forbidden message). This will happen if the object is not "available" on B and B does not have *miss_access* enabled for A (or is not acting as a parent for A).

16.13.4 How is the cause of a False Hit determined?

Assume A requests a URL from B and receives a False Hit

Using the *client* utility *PURGE* the URL from A, e.g.

```
./client -m PURGE 'URL'
```

Using the *client* utility request the object from A, e.g.

```
./client 'URL'
```

The HTTP headers of the request are available. Two header types are of particular interest:

X-Cache - this shows whether an object is available or not.

X-Cache-Lookup - this keeps the result of a store table lookup *before* refresh causing rules are checked (i.e. it indicates if the object is available before any validation would be attempted).

The X-Cache and X-Cache-Lookup headers from A should both show MISS.

If A requests the object from B (which it will if the digest lookup indicates B has it - assuming B is closest peer of course :-) then there will be another set of these headers from B.

If the X-Cache header from B shows a MISS a False Hit has occurred. This means that A thought B had an object but B tells A it does not have it available for retrieval. The reason why it is not available for retrieval is indicated by the X-Cache-Lookup header. If:

X-Cache-Lookup = MISS then either A's (version of B's) digest is out-of-date or corrupt OR a collision occurred in the digest (very small probability) OR B recently purged the object.

X-Cache-Lookup = HIT then B had the object, but refresh rules (or A's max-age requirements) prevent A from getting a HIT (validation failed).

16.13.5 Use The Source

If there is something else you need to check you can always look at the source code. The main Cache Digest functionality is organised as follows:

CacheDigest.c (debug section 70) Generic Cache Digest routines

store_digest.c (debug section 71) Local Cache Digest routines

peer_digest.c (debug section 72) Peer Cache Digest routines

Note that in the source the term *Store Digest* refers to the digest created locally. The Cache Digest code is fairly self-explanatory (once you understand how Cache Digests work):

16.14 What about ICP?

COMING SOON!

16.15 Is there a Cache Digest Specication?

There is now, thanks to *Martin Hamilton* `<mailto:martin@net.lut.ac.uk>` and *Alex Rousskov* `<mailto:rousskov@ircache.net>`.

Cache Digests, as implemented in Squid 2.1.PATCH2, are described in *cache-digest-v5.txt* `</CacheDigest/cache-digest-v5.txt>`.

You'll notice the format is similar to an Internet Draft. We decided not to submit this document as a draft because Cache Digests will likely undergo some important changes before we want to try to make it a standard.

16.16 Would it be possible to stagger the timings when cache_digests are retrieved from peers?

Note: The information here is current for version 2.2.

Squid already has code to spread the digest updates. The algorithm is currently controlled by a few hard-coded constants in *peer_digest.c*. For example, *GlobDigestReqMinGap* variable determines the minimum interval between two requests for a digest. You may want to try to increase the value of GlobDigestReqMinGap from 60 seconds to whatever you feel comfortable with (but it should be smaller than hour/number_of_peers, of course).

Note that whatever you do, you still need to give Squid enough time and bandwidth to fetch all the digests. Depending on your environment, that bandwidth may be more or less than an ICP would require. Upcoming digest deltas (x10 smaller than the digests themselves) may be the only way to solve the "big scale" problem.

17 Interception Caching/Proxying

How can I make my users' browsers use my cache without conguring the browsers for proxying?

First, it is *critical* to read the full comments in the squid.conf le! That is the only authoritative source for conguration information. However, the following instructions are correct as of this writing (July 1999.)

Getting interception caching to work requires four distinct steps:

1. **Compile and run a version of Squid which accepts connections for other addresses**. For some operating systems, you need to have congured and built a version of Squid which can recognize the hijacked connections and discern the destination addresses. For Linux this seems to work automatically. For *BSD-based systems, you probably have to congure squid with the *–enable-ipf-transparent* option. (Do a *make clean* if you previously congured without that option, or the correct settings may not be present.)

2. **Congure Squid to accept and process the connections** . You have to change the Squid conguration settings to recognize the hijacked connections and discern the destination addresses. Here are the important settings in *squid.conf*:

   ```
   http_port 8080
   httpd_accel_host virtual
   httpd_accel_port 80
   httpd_accel_with_proxy  on
   httpd_accel_uses_host_header on
   ```

3. **Get your cache server to accept the packets**. You have to congure your cache host to accept the redirected packets - any IP address, on port 80 - and deliver them to your cache application. This is typically done with IP ltering/forwarding features built into the kernel. On linux they call this *iptables* (kernel 2.4.x), *ipchains* (2.2.x) or *ipfwadm* (2.0.x). On FreeBSD its called *ipfw*. Other BSD systems may use *ip lter* or *ipnat*. On most systems, it may require rebuilding the kernel or adding a new loadable kernel module.

4. **Get the packets to your cache server**. There are several ways to do this. First, if your proxy machine is already in the path of the packets (i.e. it is routing between your proxy users and the Internet) then you don't have to worry about this step. This would be true if you install Squid on a rewall machine, or on a UNIX-based router. If the cache is not in the natural path of the connections, then you have to divert the packets from the normal path to your cache host using a router or switch. You may be able to do this with a Cisco router using their "route maps" feature, depending on your IOS version. You might also use a so-called layer-4 switch, such as the Alteon ACE-director or the Foundry Networks ServerIron. Finally, you might be able to use a stand-alone router/load-balancer type product, or routing capabilities of an access server.

Notes:

The *http_port 8080* in this example assumes you will redirect incoming port 80 packets to port 8080 on your cache machine. If you are running Squid on port 3128 (for example) you can leave it there via *http_port 3128*, and redirect to that port via your IP ltering or forwarding commands.

In the *httpd_accel_host* option, *virtual* is the magic word!

The *httpd_accel_with_proxy on* is required to enable interception proxy mode; essentially in interception proxy mode Squid thinks it is acting both as an accelerator (hence accepting packets for other IPs on port 80) and a caching proxy (hence serving les out of cache.)

You **must** use *httpd_accel_uses_host_header on* to get the cache to work properly in interception mode. This enables the cache to index its stored objects under the true hostname, as is done in a normal proxy, rather than under the IP address. This is especially important if you want to use a parent cache hierarchy, or to share cache data between interception proxy users and non-interception proxy users, which you can do with Squid in this conguration.

17.1 Interception caching for Solaris, SunOS, and BSD systems

NOTE: You don't need to use IP Filter on FreeBSD. Use the built-in *ipfw* feature instead. See the FreeBSD subsection below.

17.1.1 Install IP Filter

First, get and install the *IP Filter package* <http://coombs.anu.edu.au/ipfilter/>.

17.1.2 Congure ipnat

Put these lines in */etc/ipnat.rules*:

```
# Redirect direct web traffic to local web server.
rdr de0 1.2.3.4/32 port 80 -> 1.2.3.4 port 80 tcp

# Redirect everything else to squid on port 8080
rdr de0 0.0.0.0/0 port 80 -> 1.2.3.4 port 8080 tcp
```

Modify your startup scripts to enable ipnat. For example, on FreeBSD it looks something like this:

```
/sbin/modload /lkm/if_ipl.o
/sbin/ipnat -f /etc/ipnat.rules
chgrp nobody /dev/ipnat
chmod 644 /dev/ipnat
```

17.1.3 Congure Squid

Squid-2 Squid-2 (after version beta25) has IP lter support built in. Simple enable it when you run *congure* :

```
./configure --enable-ipf-transparent
```

Add these lines to your *squid.conf* le:

```
http_port 8080
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

Note, you don't have to use port 8080, but it must match whatever you used in the */etc/ipnat.rules* le.

Squid-1.1 Patches for Squid-1.X are available from *Quinton Dolan's Squid page* <http://www.fan.net.au/~q/squid/>. Add these lines to *squid.conf*:

```
http_port 8080
httpd_accel virtual 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

Thanks to *Quinton Dolan* <mailto:q@fan.net.au>.

17.2 Interception caching with Linux 2.0 and ipfwadm

by *Rodney van den Oever* <mailto:Rodney.van.den.Oever@tip.nl>

Note: Interception proxying does NOT work with Linux 2.0.30! Linux 2.0.29 is known to work well. If you're using a more recent kernel, like 2.2.X, then you should probably use an ipchains conguration, 17.3.

Warning: this technique has some shortcomings.

1. **This method only supports the HTTP protocol, not gopher or FTP**
2. Since the browser wasn't set up to use a proxy server, it uses the FTP protocol (with destination port 21) and not the required HTTP protocol. You can't setup a redirection-rule to the proxy server since the browser is speaking the wrong protocol. A similar problem occurs with gopher. Normally all proxy requests are translated by the client into the HTTP protocol, but since the client isn't aware of the redirection, this never happens.

If you can live with the side-eects, go ahead and compile your kernel with rewalling and redirection support. Here are the important parameters from */usr/src/linux/.cong* :

```
#
# Code maturity level options
#
CONFIG_EXPERIMENTAL=y
#
# Networking options
#
CONFIG_FIREWALL=y
# CONFIG_NET_ALIAS is not set
CONFIG_INET=y
CONFIG_IP_FORWARD=y
# CONFIG_IP_MULTICAST is not set
CONFIG_IP_FIREWALL=y
# CONFIG_IP_FIREWALL_VERBOSE is not set
CONFIG_IP_MASQUERADE=y
CONFIG_IP_TRANSPARENT_PROXY=y
CONFIG_IP_ALWAYS_DEFRAG=y
# CONFIG_IP_ACCT is not set
CONFIG_IP_ROUTER=y
```

You may also need to enable **IP Forwarding**. One way to do it is to add this line to your startup scripts:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Go to the *Linux IP Firewall and Accounting* <http://www.xos.nl/linux/ipfwadm/> page, obtain the source distribution to *ipfwadm* and install it. Older versions of *ipfwadm* may not work. You might need at least version **2.3.0**. You'll use *ipfwadm* to setup the redirection rules. I added this rule to the script that runs from */etc/rc.d/rc.inet1* (Slackware) which sets up the interfaces at boot-time. The redirection should be done before any other Input-accept rule. To really make sure it worked I disabled the forwarding (masquerading) I normally do.

/etc/rc.d/rc.rewall :

```
#!/bin/sh
# rc.firewall   Linux kernel firewalling rules
FW=/sbin/ipfwadm

# Flush rules, for testing purposes
for i in I O F # A      # If we enabled accounting too
do
        ${FW} -$i -f
done

# Default policies:
${FW} -I -p rej         # Incoming policy: reject (quick error)
${FW} -O -p acc         # Output policy: accept
${FW} -F -p den         # Forwarding policy: deny

# Input Rules:

# Loopback-interface (local access, eg, to local nameserver):
${FW} -I -a acc -S localhost/32 -D localhost/32

# Local Ethernet-interface:

# Redirect to Squid proxy server:
${FW} -I -a acc -P tcp -D default/0 80 -r 8080

# Accept packets from local network:
${FW} -I -a acc -P all -S localnet/8 -D default/0 -W eth0

# Only required for other types of traffic (FTP, Telnet):

# Forward localnet with masquerading (udp and tcp, no icmp!):
${FW} -F -a m -P tcp -S localnet/8 -D default/0
${FW} -F -a m -P udp -S localnet/8 -D default/0
```

Here all trac from the local LAN with any destination gets redirected to the local port 8080. Rules can be viewed like this:

```
IP firewall input rules, default policy: reject
type  prot source               destination          ports
acc   all  127.0.0.1            127.0.0.1            n/a
acc/r tcp  10.0.0.0/8           0.0.0.0/0            * -> 80 => 8080
acc   all  10.0.0.0/8           0.0.0.0/0            n/a
acc   tcp  0.0.0.0/0            0.0.0.0/0            * -> *
```

I did some testing on Windows 95 with both Microsoft Internet Explorer 3.01 and Netscape Communicator pre-release and it worked with both browsers with the proxy-settings disabled.

At one time *squid* seemed to get in a loop when I pointed the browser to the local port 80. But this could be avoided by adding a reject rule for client to this address:

```
${FW} -I -a rej -P tcp -S localnet/8 -D hostname/32 80

IP firewall input rules, default policy: reject
type  prot source              destination         ports
acc   all  127.0.0.1           127.0.0.1           n/a
rej   tcp  10.0.0.0/8          10.0.0.1            * -> 80
acc/r tcp  10.0.0.0/8          0.0.0.0/0           * -> 80 => 8080
acc   all  10.0.0.0/8          0.0.0.0/0           n/a
acc   tcp  0.0.0.0/0           0.0.0.0/0           * -> *
```

NOTE on resolving names: Instead of just passing the URLs to the proxy server, the browser itself has to resolve the URLs. Make sure the workstations are setup to query a local nameserver, to minimize outgoing trac.

If you're already running a nameserver at the rewall or proxy server (which is a good idea anyway IMHO) let the workstations use this nameserver.

Additional notes from *Richard Ayres* <mailto:RichardA@noho.co.uk>

> I'm using such a setup. The only issues so far have been that:
>
> 1. It's fairly useless to use my service providers parent caches (cache-?.www.demon.net) because by proxying squid only sees IP addresses, not host names and demon aren't generally asked for IP addresses by other users;
> 2. Linux kernel 2.0.30 is a no-no as interception proxying is broken (I use 2.0.29);
> 3. Client browsers must do host name lookups themselves, as they don't know they're using a proxy;
> 4. The Microsoft Network won't authorize its users through a proxy, so I have to specically *not* redirect those packets (my company is a MSN content provider).
>
> Aside from this, I get a 30-40% hit rate on a 50MB cache for 30-40 users and am quite pleased with the results.

See also *Daniel Kiracofe's page* <http://www.ibiblio.org/pub/Linux/docs/HOWTO/mini/other-formats/html_single/Tr

17.3 Interception caching with Linux 2.2 and ipchains

by *Martin Lyons* <mailto:Support@dnet.co.uk>

You need to congure your kernel for ipchains. Conguring Linux kernels is beyond the scope of this FAQ. One way to do it is:

```
# cd /usr/src/linux
# make menuconfig
```

The following shows important kernel features to include:

```
[*] Network firewalls
[ ] Socket Filtering
[*] Unix domain sockets
[*] TCP/IP networking
[ ] IP: multicasting
[ ] IP: advanced router
[ ] IP: kernel level autoconfiguration
[*] IP: firewalling
[ ] IP: firewall packet netlink device
[*] IP: always defragment (required for masquerading)
[*] IP: transparent proxy support
```

You must include the *IP: always defragment*, otherwise it prevents you from using the REDIRECT chain.

You can use this script as a template for your own *rc.rewall* to congure ipchains:

```
#!/bin/sh
# rc.firewall   Linux kernel firewalling rules
# Leon Brooks (leon at brooks dot fdns dot net)
FW=/sbin/ipchains
ADD="$FW -A"

# Flush rules, for testing purposes
for i in I O F # A      # If we enabled accounting too
do
        ${FW} -F $i
done

# Default policies:
${FW} -P input REJECT   # Incoming policy: reject (quick error)
${FW} -P output ACCEPT  # Output policy: accept
${FW} -P forward DENY   # Forwarding policy: deny

# Input Rules:

# Loopback-interface (local access, eg, to local nameserver):
${ADD} input -j ACCEPT -s localhost/32 -d localhost/32

# Local Ethernet-interface:

# Redirect to Squid proxy server:
${ADD} input -p tcp -d 0/0 80 -j REDIRECT 8080

# Accept packets from local network:
${ADD} input -j ACCEPT -s localnet/8 -d 0/0 -i eth0

# Only required for other types of traffic (FTP, Telnet):

# Forward localnet with masquerading (udp and tcp, no icmp!):
${ADD} forward -j MASQ -p tcp -s localnet/8 -d 0/0
${ADD} forward -j MASQ -P udp -s localnet/8 -d 0/0
```

Also, *Andrew Shipton* <mailto:andrew@careless.net> notes that with 2.0.x kernels you don't need to enable packet forwarding, but with the 2.1.x and 2.2.x kernels using ipchains you do. Packet forwarding is enabled with the following command:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

17.4 Interception caching with Linux 2.4 and netlter

NOTE: this information comes from Daniel Kiracofe's *Transparent Proxy with Squid mini-HOWTO* <http://www.linuxdoc.org/HOWTO/mini/TransparentProxy.html>.

To support netlter transparent interception on Linux 2.4 Squid must be compiled with the –enable-linux-netlter option.

To enable netwlter support you may need to build a new kernel. Be sure to enable all of these options:

Networking support

Sysctl support

Network packet ltering

TCP/IP networking

Connection tracking (Under "IP: Netlter Conguration" in menucong)

IP tables support

Full NAT

REDIRECT target support

/proc lesystem support

You must say NO to "Fast switching"

After building the kernel, install it and reboot.

You may need to enable packet forwarding (e.g. in your startup scripts):

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Use the *iptables* command to make your kernel intercept HTTP connections and send them to Squid:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 3128
```

17.5 Interception caching with Cisco routers

by *John Saunders* <mailto:John.Saunders@scitec.com.au>

This works with at least IOS 11.1 and later I guess. Possibly earlier, as I'm no CISCO expert I can't say for sure. If your router is doing anything more complicated that shuing packets between an ethernet interface and either a serial port or BRI port, then you should work through if this will work for you.

First dene a route map with a name of proxy-redirect (name doesn't matter) and specify the next hop to be the machine Squid runs on.

```
!
route-map proxy-redirect permit 10
 match ip address 110
 set ip next-hop 203.24.133.2
!
```

Dene an access list to trap HTTP requests. The second line allows the Squid host direct access so an routing loop is not formed. By carefully writing your access list as show below, common cases are found quickly and this can greatly reduce the load on your router's processor.

```
!
access-list 110 deny   tcp any any neq www
access-list 110 deny   tcp host 203.24.133.2 any
access-list 110 permit tcp any any
!
```

Apply the route map to the ethernet interface.

```
!
interface Ethernet0
 ip policy route-map proxy-redirect
!
```

17.5.1 possible bugs

Bruce Morgan <mailto:morgan@curtin.net> notes that there is a Cisco bug relating to interception proxying using IP policy route maps, that causes NFS and other applications to break. Apparently there are two bug reports raised in Cisco, but they are not available for public dissemination.

The problem occurs with o/s packets with more than 1472 data bytes. If you try to ping a host with more than 1472 data bytes across a Cisco interface with the access-lists and ip policy route map, the icmp request will fail. The packet will be fragmented, and the rst fragment is checked against the access-list and rejected - it goes the "normal path" as it is an icmp packet - however when the second fragment is checked against the access-list it is accepted (it isn't regarded as an icmp packet), and goes to the action determined by the policy route map!

John <mailto:John.Saunders@scitec.com.au> notes that you may be able to get around this bug by carefully writing your access lists. If the last/default rule is to permit then this bug would be a problem, but if the last/default rule was to deny then it won't be a problem. I guess fragments, other than the rst, don't have the information available to properly policy route them. Normally TCP packets should not be fragmented, at least my network runs an MTU of 1500 everywhere to avoid fragmentation. So this would aect UDP and ICMP trac only.

Basically, you will have to pick between living with the bug or better performance. This set has better performance, but suers from the bug:

```
access-list 110 deny   tcp any any neq www
access-list 110 deny   tcp host 10.1.2.3 any
access-list 110 permit tcp any any
```

Conversely, this set has worse performance, but works for all protocols:

```
access-list 110 deny   tcp host 10.1.2.3 any
```

```
access-list 110 permit tcp any any eq www
access-list 110 deny   tcp any any
```

17.6 Interception caching with LINUX 2.0.29 and CISCO IOS 11.1

Just for kicks, here's an email message posted to squid-users on how to make interception proxying work with a Cisco router and Squid running on Linux.

by *Brian Feeny* <mailto:signal@shreve.net>

Here is how I have Interception proxying working for me, in an environment where my router is a Cisco 2501 running IOS 11.1, and Squid machine is running Linux 2.0.33.

Many thanks to the following individuals and the squid-users list for helping me get redirection and interception proxying working on my Cisco/Linux box.

Lincoln Dale

Riccardo Vratogna

Mark White

Henrik Nordstrom

First, here is what I added to my Cisco, which is running IOS 11.1. In IOS 11.1 the route-map command is "process switched" as opposed to the faster "fast-switched" route-map which is found in IOS 11.2 and later. You may wish to be running IOS 11.2. I am running 11.1, and have had no problems with my current load of about 150 simultaneous connections to squid.:

```
!
interface Ethernet0
 description To Office Ethernet
 ip address 208.206.76.1 255.255.255.0
 no ip directed-broadcast
 no ip mroute-cache
 ip policy route-map proxy-redir
!
access-list 110 deny   tcp host 208.206.76.44 any eq www
access-list 110 permit tcp any any eq www
route-map proxy-redir permit 10
 match ip address 110
 set ip next-hop 208.206.76.44
```

So basically from above you can see I added the "route-map" declaration, and an access-list, and then turned the route-map on under int e0 "ip policy route-map proxy-redir"

ok, so the Cisco is taken care of at this point. The host above: 208.206.76.44, is the ip number of my squid host.

My squid box runs Linux, so I had to do the following on it:

my kernel (2.0.33) cong looks like this:

```
#
# Networking options
```

```
#
CONFIG_FIREWALL=y
# CONFIG_NET_ALIAS is not set
CONFIG_INET=y
CONFIG_IP_FORWARD=y
CONFIG_IP_MULTICAST=y
CONFIG_SYN_COOKIES=y
# CONFIG_RST_COOKIES is not set
CONFIG_IP_FIREWALL=y
# CONFIG_IP_FIREWALL_VERBOSE is not set
CONFIG_IP_MASQUERADE=y
# CONFIG_IP_MASQUERADE_IPAUTOFW is not set
CONFIG_IP_MASQUERADE_ICMP=y
CONFIG_IP_TRANSPARENT_PROXY=y
CONFIG_IP_ALWAYS_DEFRAG=y
# CONFIG_IP_ACCT is not set
CONFIG_IP_ROUTER=y
```

You will need Firewalling and Transparent Proxy turned on at a minimum.

Then some ipfwadm stu:

```
# Accept all on loopback
ipfwadm -I -a accept -W lo
# Accept my own IP, to prevent loops (repeat for each interface/alias)
ipfwadm -I -a accept -P tcp -D 208.206.76.44 80
# Send all traffic destined to port 80 to Squid on port 3128
ipfwadm -I -a accept -P tcp -D 0/0 80 -r 3128
```

it accepts packets on port 80 (redirected from the Cisco), and redirects them to 3128 which is the port my squid process is sitting on. I put all this in /etc/rc.d/rc.local

I am using *v1.1.20 of Squid* </Versions/1.1/1.1.20/> with *Henrik's patch* <http://devel.squid-cache.org/hno/patches/squid-1.1.20.host_and_virtual.patch> installed. You will want to install this patch if using a setup similar to mine.

17.7 The cache is trying to connect to itself...

by *Henrik Nordstrom* <mailto:hno@squid-cache.org>

I think almost everyone who have tried to build a interception proxy setup have been bitten by this one.

Measures you can take:

Deny Squid from fetching objects from itself (using ACL lists).

Apply a small patch that prevents Squid from looping innitely (available from *Henrik's Squid Patches* <http://devel.squid-cache.org/hno/>)

Don't run Squid on port 80, and redirect port 80 not destined for the local machine to Squid (redirection == iplter/ipfw/ipfadm). This avoids the most common loops.

If you are using iplter then you should also use transproxyd in front of Squid. Squid does not yet know how to interface to iplter (patches are welcome: squid-bugs@squid-cache.org).

17.8 Interception caching with FreeBSD

by Duane Wessels

I set out yesterday to make interception caching work with Squid and FreeBSD. It was, uh, fun.

It was relatively easy to congure a cisco to divert port 80 packets to my FreeBSD box. Conguration goes something like this:

```
access-list 110 deny   tcp host 10.0.3.22 any eq www
access-list 110 permit tcp any any eq www
route-map proxy-redirect permit 10
 match ip address 110
 set ip next-hop 10.0.3.22
int eth2/0
 ip policy route-map proxy-redirect
```

Here, 10.0.3.22 is the IP address of the FreeBSD cache machine.

Once I have packets going to the FreeBSD box, I need to get the kernel to deliver them to Squid. I started on FreeBSD-2.2.7, and then downloaded *IPFilter* `<ftp://coombs.anu.edu.au/pub/net/ip-filter/>`. This was a dead end for me. The IPFilter distribution includes patches to the FreeBSD kernel sources, but many of these had conicts. Then I noticed that the IPFilter page says "It comes as a part of [FreeBSD-2.2 and later]." Fair enough. Unfortunately, you can't hijack connections with the FreeBSD-2.2.X IPFIREWALL code (*ipfw*), and you can't (or at least I couldn't) do it with *natd* either.

FreeBSD-3.0 has much better support for connection hijacking, so I suggest you start with that. You need to build a kernel with the following options:

```
options         IPFIREWALL
options         IPFIREWALL_FORWARD
```

Next, its time to congure the IP rewall rules with *ipfw*. By default, there are no "allow" rules and all packets are denied. I added these commands to */etc/rc.local* just to be able to use the machine on my network:

```
ipfw add 60000 allow all from any to any
```

But we're still not hijacking connections. To accomplish that, add these rules:

```
ipfw add 49  allow tcp from 10.0.3.22 to any
ipfw add 50  fwd 127.0.0.1 tcp from any to any 80
```

The second line (rule 50) is the one which hijacks the connection. The rst line makes sure we never hit rule 50 for trac originated by the local machine. This prevents forwarding loops.

Note that I am not changing the port number here. That is, port 80 packets are simply diverted to Squid on port 80. My Squid conguration is:

```
http_port 80
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

If you don't want Squid to listen on port 80 (because that requires root privileges) then you can use another port. In that case your ipfw redirect rule looks like:

```
ipfw add 50 fwd 127.0.0.1,3128 tcp from any to any 80
```

and the *squid.conf* lines are:

```
http_port 3128
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

17.9 Interception caching with ACC Tigris digital access server

by *John Saunders* <mailto:John.Saunders@scitec.com.au>

This is to do with conguring interception proxy for an ACC Tigris digital access server (like a CISCO 5200/5300 or an Ascend MAX 4000). I've found that doing this in the NAS reduces trac on the LAN and reduces processing load on the CISCO. The Tigris has ample CPU for ltering.

Step 1 is to create lters that allow local trac to pass. Add as many as needed for all of your address ranges.

```
ADD PROFILE IP FILTER ENTRY local1 INPUT  10.0.3.0 255.255.255.0 0.0.0.0 0.0.0.0 NORMAL
ADD PROFILE IP FILTER ENTRY local2 INPUT  10.0.4.0 255.255.255.0 0.0.0.0 0.0.0.0 NORMAL
```

Step 2 is to create a lter to trap port 80 trac.

```
ADD PROFILE IP FILTER ENTRY http INPUT  0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 = 0x6 D= 80 NORMAL
```

Step 3 is to set the "APPLICATION_ID" on port 80 trac to 80. This causes all packets matching this lter to have ID 80 instead of the default ID of 0.

```
SET PROFILE IP FILTER APPLICATION_ID http 80
```

Step 4 is to create a special route that is used for packets with "APPLICATION_ID" set to 80. The routing engine uses the ID to select which routes to use.

```
ADD IP ROUTE ENTRY 0.0.0.0 0.0.0.0 PROXY-IP 1
SET IP ROUTE APPLICATION_ID 0.0.0.0 0.0.0.0 PROXY-IP 80
```

Step 5 is to bind everything to a lter ID called transproxy. List all local lters rst and the http one last.

```
ADD PROFILE ENTRY transproxy local1 local2 http
```

With this in place use your RADIUS server to send back the "Framed-Filter-Id = transproxy" key/value pair to the NAS.

You can check if the lter is being assigned to logins with the following command:

```
display profile port table
```

17.10 "Connection reset by peer" and Cisco policy routing

Fyodor <mailto:fygrave at tigerteam dot net> has tracked down the cause of unusual "connection reset by peer" messages when using Cisco policy routing to hijack HTTP requests.

When the network link between router and the cache goes down for just a moment, the packets that are supposed to be redirected are instead sent out the default route. If this happens, a TCP ACK from the client host may be sent to the origin server, instead of being diverted to the cache. The origin server, upon receiving an unexpected ACK packet, sends a TCP RESET back to the client, which aborts the client's request.

To work around this problem, you can install a static route to the *null0* interface for the cache address with a higher metric (lower precedence), such as 250. Then, when the link goes down, packets from the client just get dropped instead of sent out the default route. For example, if 1.2.3.4 is the IP address of your Squid cache, you may add:

```
ip route 1.2.3.4 255.255.255.255 Null0 250
```

This appears to cause the correct behaviour.

17.11 WCCP - Web Cache Coordination Protocol

Contributors: *Glenn Chisholm* <mailto:glenn@ircache.net>, *Lincoln Dale* <mailto:ltd@cisco.com> and *Reuben Farrelly* <mailto:reuben-squid@reub.net>.

17.11.1 Does Squid support WCCP?

CISCO's Web Cache Coordination Protocol V1.0 is supported in squid 2.3 and later. support WCCP V2.0. Now that WCCP V2 is an open protocol, Squid may be able to support it in the future.

17.11.2 Conguring your Router

There are two dierent methods of conguring WCCP on CISCO routers. The rst method is for routers that only support V1.0 of the protocol. The second is for routers that support both.

IOS Version 11.x It is possible that later versions of IOS 11.x will support V2.0 of the protocol. If that is the case follow the 12.x instructions. Several people have reported that the squid implimentation of WCCP does not work with their 11.x routers. If you experience this please mail the debug output from your router to *squid-bugs*.

```
conf t

wccp enable
!
interface [Interface carrying Outgoing Traffic]x/x
!
ip wccp web-cache redirect
!
CTRL Z
write mem
```

IOS Version 12.x Some of the early versions of 12.x do not have the 'ip wccp version' command. You will need to upgrade your IOS version to use V1.0.

You will need to be running at least IOS Software Release *12.0(5)T* if you're running the 12.0 T-train. IOS Software Releases *12.0(3)T* and *12.0(4)T* do not have WCCPv1, but *12.0(5)T* does.

```
conf t

ip wccp version 1
ip wccp web-cache redirect-list 150
!
interface [Interface carrying Outgoing/Incoming Traffic]x/x
ip wccp web-cache redirect out|in
!
CTRL Z
write mem
```

Replace 150 with an access list number (either standard or extended) which lists IP addresses which you do not wish to be transparently redirected to your cache. Otherwise simply user the word 'redirect' on it's own to redirect trac from all sources to all destinations.

17.11.3 IOS 12.x problems

Some people report problems with WCCP and IOS 12.x. They see truncated or fragmented GRE packets arriving at the cache. Apparently it works if you disable Cisco Express Forwarding for the interface:

```
conf t
ip cef          # some systems may already have 'ip cef global'
int Ethernet 0/0      (or int FastEthernet 0/0 or other internal interface)
no ip route-cache cef
CTRL Z
```

This may well be xed in later releases of IOS.

17.11.4 Conguring FreeBSD

FreeBSD rst needs to be congured to receive and strip the GRE encapsulation from the packets from the router. To do this you will need to patch and recompile your kernel. The steps depend on your kernel version.

FreeBSD-3.x

1. Apply the *patch for FreeBSD-3.x kernels* <../../WCCP-support/FreeBSD-3.x/gre.patch>:

   ```
   # cd /usr/src
   # patch -s < /tmp/gre.patch
   ```

2. Download *gre.c for FreeBSD-3.x* <../../WCCP-support/FreeBSD-3.x/gre.c>. Save this le as */usr/src/sys/netinet/gre.c*.

3. Add "options GRE" to your kernel cong le and rebuild your kernel. Note, the *opt_gre.h* le is created when you run *cong* . Once your kernel is installed you will need to 17.8.

FreeBSD-4.0 through 4.7 The procedure is nearly identical to the above for 3.x, but the source les are a little dierent.

1. Apply the most appropriate patch le from the list of *patches for 4.x kernels* <../../WCCP-support/FreeBSD-4.x>.
2. Download *gre.c for FreeBSD-3.x* <../../WCCP-support/FreeBSD-3.x/gre.c>. Save this le as */usr/src/sys/netinet/gre.c*.
3. Add "options GRE" to your kernel cong le and rebuild your kernel. Note, the *opt_gre.h* le is created when you run *cong* . Once your kernel is installed you will need to 17.8.

FreeBSD-4.8 and later The operating system now comes standard with some GRE support. You need to make a kernel with the GRE code enabled:

```
pseudo-device   gre
```

And then congure the tunnel so that the router's GRE packets are accepted:

```
# ifconfig gre0 create
# ifconfig gre0 $squid_ip $router_ip netmask 255.255.255.255 up
# ifconfig gre0 tunnel $squid_ip $router_ip
# route delete $router_ip
```

17.11.5 Conguring Linux 2.2

Al Blake has written a *Cookbook for setting up transparent WCCP using Squid on RedHat Linux and a cisco access server* <http://www.spc.int/it/TechHead/Wccp-squid.html>.

There are currently two methods for supporting WCCP with Linux 2.2. A specic purpose module. Or the standard Linux GRE tunneling driver. People have reported diculty with the standard GRE tunneling driver, however it does allow GRE functionality other than WCCP. You should choose the method that suits your enviroment.

Standard Linux GRE Tunnel Linux 2.2 kernels already support GRE, as long as the GRE module is compiled into the kernel.

Ensure that the GRE code is either built as static or as a module by chosing the appropriate option in your kernel cong. Then rebuild your kernel. If it is a module you will need to:

```
modprobe ip_gre
```

The next step is to tell Linux to establish an IP tunnel between the router and your host. Daniele Orlandi reports that you have to give the gre1 interface an address, but any old address seems to work.

```
iptunnel add gre1 mode gre remote <Router-IP> local <Host-IP> dev <interface>
ifconfig gre1 127.0.0.2 up
```

<Router-IP> is the IP address of your router that is intercepting the HTTP packets. <Host-IP> is the IP address of your cache, and <interface> is the network interface that receives those packets (probably eth0).

Joe Cooper's Patch Joe Cooper has a patch for Linux 2.2.18 kernel on his *Squid page* <http://www.swelltech.com/pengies/joe/patches/>.

WCCP Specic Module This module is not part of the standard Linux distributon. It needs to be compiled as a module and loaded on your system to function. Do not attempt to build this in as a static part of your kernel.

Download the *Linux WCCP module* <../../WCCP-support/Linux/ip_wccp.c> and compile it as you would any Linux network module.

Copy the module to */lib/modules/kernel-version/ipv4/ip_wccp.o*. Edit */lib/modules/kernel-version/modules.dep* and add:

```
/lib/modules/kernel-version/ipv4/ip_wccp.o:
```

Finally you will need to load the module:

```
modprobe ip_wccp
```

Common Steps The machine should now be striping the GRE encapsulation from any packets recieved and requeuing them. The system will also need to be congured for interception proxying, either with 17.2 or with 17.3.

17.11.6 Conguring Others

If you have managed to conguring your operating system to support WCCP with Squid please contact us with the details so we may share them with others.

17.12 Can someone tell me what version of cisco IOS WCCP is added in?

IOS releases:

11.1(19?)CA/CC or later

11.2(14)P or later

12.0(anything) or later

17.13 What about WCCPv2?

Cisco has published WCCPv2 as an *Internet Draft* <http://www.web-cache.com/Writings/Internet-Drafts/draft-wilso (expired Jan 2001). At this point, Squid does not support WCCPv2, but anyone is welcome to code it up and contribute to the Squid project.

17.14 Interception caching with Foundry L4 switches

by *Brian Feeny* <mailto:signal at shreve dot net>.

First, congure Squid for interception caching as detailed at the 17.

Next, congure the Foundry layer 4 switch to redirect trac to your Squid box or boxes. By default, the Foundry redirects to port 80 of your squid box. This can be changed to a dierent port if needed, but won't be covered here.

In addition, the switch does a "health check" of the port to make sure your squid is answering. If you squid does not answer, the switch defaults to sending trac directly thru instead of redirecting it. When the Squid comes back up, it begins redirecting once again.

This example assumes you have two squid caches:

```
squid1.foo.com  192.168.1.10
squid2.foo.com  192.168.1.11
```

We will assume you have various workstations, customers, etc, plugged into the switch for which you want them to be intercepted and sent to Squid. The squid caches themselves should be plugged into the switch as well. Only the interface that the router is connected to is important. Where you put the squid caches or other connections does not matter.

This example assumes your router is plugged into interface **17** of the switch. If not, adjust the following commands accordingly.

1. Enter conguration mode:

   ```
   telnet@ServerIron#conf t
   ```

2. Congure each squid on the Foundry:

   ```
   telnet@ServerIron(config)# server cache-name squid1 192.168.1.10
   telnet@ServerIron(config)# server cache-name squid2 192.168.1.11
   ```

3. Add the squids to a cache-group:

   ```
   telnet@ServerIron(config)#server cache-group 1
   telnet@ServerIron(config-tc-1)#cache-name squid1
   telnet@ServerIron(config-tc-1)#cache-name squid2
   ```

4. Create a policy for caching http on a local port

   ```
   telnet@ServerIron(config)# ip policy 1 cache tcp http local
   ```

5. Enable that policy on the port connected to your router

   ```
   telnet@ServerIron(config)#int e 17
   telnet@ServerIron(config-if-17)# ip-policy 1
   ```

Since all outbound trac to the Internet goes out interface **17** (the router), and interface **17** has the caching policy applied to it, HTTP trac is going to be intercepted and redirected to the caches you have congured.

The default port to redirect to can be changed. The load balancing algorithm used can be changed (Least Used, Round Robin, etc). Ports can be exempted from caching if needed. Access Lists can be applied so that only certain source IP Addresses are redirected, etc. This information was left out of this document since this was just a quick howto that would apply for most people, not meant to be a comprehensive manual of how to congure a Foundry switch. I can however revise this with any information necessary if people feel it should be included.

17.15 Can I use *proxy_auth* with interception?

No, you cannot. With interception proxying, the client thinks it is talking to an origin server and would never send the *Proxy-authorization* request header.

18 SNMP

Contributors: *Glenn Chisholm* <mailto:glenn@ircache.net>.

18.1 Does Squid support SNMP?

True SNMP support is available in squid 2 and above. A signicant change in the implimentation occured starting with the development 2.2 code. Therefore there are two sets of instructions on how to congure SNMP in squid, please make sure that you follow the correct one.

18.2 Enabling SNMP in Squid

To use SNMP, it must rst be enabled with the *congure* script, and squid rebuilt. To enable is rst run the script:

```
./configure --enable-snmp  [ ... other configure options ]
```

Next, recompile after cleaning the source tree :

```
make clean
make all
make install
```

Once the compile is completed and the new binary is installed the *squid.conf* le needs to be congured to allow access; the default is to deny all requests. The instructions on how to do this have been broken into two parts, the rst for all versions of Squid from 2.2 onwards and the second for 2.1 and below.

18.3 Conguring Squid 2.2

To congure SNMP rst specify a list of communities that you would like to allow access by using a standard acl of the form:

```
acl aclname snmp_community string
```

For example:

```
acl snmppublic snmp_community public
acl snmpjoebloggs snmp_community joebloggs
```

This creates two acl's, with two dierent communities, public and joebloggs. You can name the acl's and the community strings anything that you like.

To specify the port that the agent will listen on modify the "snmp_port" parameter, it is defaulted to 3401. The port that the agent will forward requests that can not be furlled by this agent to is set by "forward_snmpd_port" it is defaulted to o. It must be congured for this to work. Remember that as the requests will be originating from this agent you will need to make sure that you congure your access accordingly.

To allow access to Squid's SNMP agent, dene an *snmp_access* ACL with the community strings that you previously dened. For example:

```
snmp_access allow snmppublic localhost
snmp_access deny all
```

The above will allow anyone on the localhost who uses the community *public* to access the agent. It will deny all others access.

If you do not dene any *snmp_access* ACL's, then SNMP access is denied by default.

Finally squid allows to you to congure the address that the agent will bind to for incomming and outgoing trac. These are defaulted to 0.0.0.0, changing these will cause the agent to bind to a specic address on the host, rather than the default which is all.

```
snmp_incoming_address 0.0.0.0
snmp_outgoing_address 0.0.0.0
```

18.4 Conguring Squid 2.1

Prior to Squid 2.1 the SNMP code had a number of issues with the ACL's. If you are a frequent user of SNMP with Squid, please upgrade to 2.2 or higher.

A sort of default, working conguration is:

```
snmp_port 3401
snmp_mib_path /local/squid/etc/mib.txt

snmp_agent_conf view all .1.3.6 included
snmp_agent_conf view squid .1.3.6 included
snmp_agent_conf user squid - all all public
snmp_agent_conf user all all all all  squid
snmp_agent_conf community public squid squid
snmp_agent_conf community readwrite all all
```

Note that for security you are advised to restrict SNMP access to your caches. You can do this easily as follows:

```
acl snmpmanagementhosts 1.2.3.4/255.255.255.255 1.2.3.0/255.255.255.0
snmp_acl public deny all !snmpmanagementhosts
snmp_acl readwrite deny all
```

You must follow these instructions for 2.1 and below exactly or you are likely to have problems. The parser has some issues which have been corrected in 2.2.

18.5 How can I query the Squid SNMP Agent

You can test if your Squid supports SNMP with the *snmpwalk* program (*snmpwalk* is a part of the *NET-SNMP project* <http://net-snmp.sourceforge.net/>). Note that you have to specify the SNMP port, which in Squid defaults to 3401.

```
snmpwalk -p 3401 hostname communitystring .1.3.6.1.4.1.3495.1.1
```

If it gives output like:

```
enterprises.nlanr.squid.cacheSystem.cacheSysVMsize = 7970816
enterprises.nlanr.squid.cacheSystem.cacheSysStorage = 2796142
enterprises.nlanr.squid.cacheSystem.cacheUptime = Timeticks: (766299) 2:07:42.99
```

then it is working ok, and you should be able to make nice statistics out of it.

For an explanation of what every string (OID) does, you should refer to the *Squid SNMP web pages* </SNMP/>.

18.6 What can I use SNMP and Squid for?

There are a lot of things you can do with SNMP and Squid. It can be useful in some extent for a longer term overview of how your proxy is doing. It can also be used as a problem solver. For example: how is it going with your ledescriptor usage? or how much does your LRU vary along a day. Things you can't monitor very well normally, aside from clicking at the cachemgr frequently. Why not let MRTG do it for you?

18.7 How can I use SNMP with Squid?

There are a number of tools that you can use to monitor Squid via SNMP. Many people use MRTG. Another good combination is *NET-SNMP* <http://net-snmp.sourceforge.net/> plus *RRDTool* <http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/>. You might be able to nd more information at the *Squid SNMP web pages* </SNMP/> or *ircache rrdtool scipts* <http://wessels.squid-cache.org/squid-rrd/>

18.8 Where can I get more information/discussion about Squid and SNMP?

General Discussion: *cache-snmp@ircache.net* <mailto:cache-snmp@ircache.net> These messages are *archived* <http://www.squid-cache.org/mail-archive/cache-snmp/>.

Subscriptions should be sent to: *cache-snmp-request@ircache.net* <mailto:cache-snmp-request@ircache.net>.

18.9 Monitoring Squid with MRTG

Some people use *MRTG* <http://www.mrtg.org/> to query Squid through its SNMP interface.

To get instruction on using MRTG with Squid please visit these pages:

1. *Cache Monitoring - How to set up your own monitoring* <http://www.cache.dfn.de/DFN-Cache/Development/Monito by DFN-Cache
2. *Using MRTG to monitor Squid* <http://www.serassio.it/SquidNT/mrtg.htm> by Guido Serassio
3. *Squid Conguration Manual - Monitoring Squid* <http://squid.visolve.com/related/snmp/monitoringsquid.htm> by Visolve
4. *Using MRTG for Squid monitoring* <http://www.arnes.si/~matija/utrecht/lecture.html> Desire II caching workshop session by Matija Grabnar
5. *How do I monitor my Squid 2 cache using MRT* <http://hermes.wwwcache.ja.net/FAQ/FAQ-2.html#mrtg> by The National Janet Web Cache Service

Further examples of Squid MRTG congurations can be found here:

1. *MRTG HOWTO Collection / Squid* <http://howto.aphroland.de/HOWTO/MRTG/SquidMonitoringWithMRTG> from MRTG
2. *using mrtg to monitor Squid* <http://people.ee.ethz.ch/~oetiker/webtools/mrtg/squid.html> from MRTG
3. *Chris' MRTG Resources* <http://www.psychofx.com/chris/unix/mrtg/>
4. *MRTG & Squid* <http://thproxy.jinr.ru/file-archive/doc/squid/cache-snmp/mrtg-demo/> by Glenn Chisholm
5. *Braindump* <http://www.braindump.dk/en/wiki/?catid=7&wikipage=ConfigFiles> by Joakim Recht

19 Squid version 2

19.1 What are the new features?

persistent connections.

Lower VM usage; in-transit objects are not held fully in memory.

Totally independent swap directories.

Customizable error texts.

FTP supported internally; no more ftpget.

Asynchronous disk operations (optional, requires pthreads library).

Internal icons for FTP and gopher directories.

snprintf() used everywhere instead of sprintf().

SNMP.

URN support </urn-support.html>

Routing requests based on AS numbers.

Cache Digests <FAQ-16.html>

...and many more!

19.2 How do I congure 'ssl _proxy' now?

By default, Squid connects directly to origin servers for SSL requests. But if you must force SSL requests through a parent, rst tell Squid it can not go direct for SSL:

```
acl SSL method CONNECT
never_direct allow SSL
```

With this in place, Squid *should* pick one of your parents to use for SSL requests. If you want it to pick a particular parent, you must use the *cache_peer_access* conguration:

```
cache_peer parent1 parent 3128 3130
cache_peer parent2 parent 3128 3130
cache_peer_access parent2 allow !SSL
```

The above lines tell Squid to NOT use *parent2* for SSL, so it should always use *parent1*.

19.3 Logle rotation doesn't work with Async I/O

It is a know limitation when using Async I/O on Linux. The Linux Threads package steals (uses internally) the SIGUSR1 signal that squid uses to rotate logs.

In order to not disturb the threads package SIGUSR1 use is disabled in Squid when threads is enabled on Linux.

19.4 Adding a new cache disk

Simply add your new *cache_dir* line to *squid.conf*, then run *squid -z* again. Squid will create swap directories on the new disk and leave the existing ones in place.

19.5 Squid 2 performs badly on Linux

by *Henrik Nordstrom* <mailto:hno@squid-cache.org>

You may have enabled Asyncronous I/O with the *–enable-async-io* congure option. Be careful when using threads on Linux. Most versions of libc5 and very early versions of glibc have problems with threaded applications. I would not recommend *–enable-async-io* on Linux unless your system uses glibc 2.1.3 or later.

You should also know that *–enable-async-io* is not optimal unless you have a very busy cache. For low loads the cache performs slightly better without *–enable-async-io*.

Try recompiling Squid without *–enable-async-io*. If a non-threaded Squid performs better then your libc probably can't handle threads correctly. (don't forget "make clean" after running congure)

19.6 How do I congure proxy authentication with Squid-2?

For Squid-2, the implementation and conguration has changed. Authentication is now handled via external processes. Arjan's *proxy auth page* <http://www.iae.nl/users/devet/squid/proxy_auth/> describes how to set it up. Some simple instructions are given below as well.

1. We assume you have congured an ACL entry with proxy _auth, for example:

   ```
   acl foo proxy_auth REQUIRED
   http_access allow foo
   ```

2. You will need to compile and install an external authenticator program. Most people will want to use *ncsa_auth*. The source for this program is included in the source distribution, in the *auth_modules/NCSA* directory.

   ```
   % cd auth_modules/NCSA
   % make
   % make install
   ```

You should now have an *ncsa_auth* program in the same directory where your *squid* binary lives.

3. You may need to create a password le. If you have been using proxy authentication before, you probably already have such a le. You can get *apache's htpasswd program* `<../../htpasswd/>` from our server. Pick a pathname for your password le. We will assume you will want to put it in the same directory as your squid.conf.

4. Congure the external authenticator in *squid.conf*. For *ncsa_auth* you need to give the pathname to the executable and the password le as an argument. For example:

```
authenticate_program /usr/local/squid/bin/ncsa_auth /usr/local/squid/etc/passwd
```

After all that, you should be able to start up Squid. If we left something out, or haven't been clear enough, please let us know (squid-faq@squid-cache.org).

19.7 Why does proxy-auth reject all users with Squid-2.2?

The ACL for proxy-authentication has changed from:

```
acl foo proxy_auth timeout
```

to:

```
acl foo proxy_auth username
```

Please update your ACL appropriately - a username of *REQUIRED* will permit all valid usernames. The timeout is now specied with the conguration option:

```
authenticate_ttl timeout
```

19.8 Delay Pools

by *David Luyer* `<mailto:david@luyer.net>`.

The information here is current for version 2.2. It is strongly recommended that you use at least Squid 2.2 if you wish to use delay pools.

Delay pools provide a way to limit the bandwidth of certain requests based on any list of criteria. The idea came from a Western Australian university who wanted to restrict student trac costs (without aecting sta trac, and still getting cache and local peering hits at full speed). There was some early Squid 1.0 code by Central Network Services at Murdoch University, which I then developed (at the University of Western Australia) into a much more complex patch for Squid 1.0 called "DELAY_HACK." I then tried to code it in a much cleaner style and with slightly more generic options than I personally needed, and called this "delay pools" in Squid 2. I almost completely recoded this in Squid 2.2 to provide the greater exibility requested by people using the feature.

To enable delay pools features in Squid 2.2, you must use the *–enable-delay-pools* congure option before compilation.

Terminology for this FAQ entry:

pool

a collection of bucket groups as appropriate to a given class

bucket group

a group of buckets within a pool, such as the per-host bucket group, the per-network bucket group or the aggregate bucket group (the aggregate bucket group is actually a single bucket)

bucket

an individual delay bucket represents a trac allocation which is replenished at a given rate (up to a given limit) and causes trac to be delayed when empty

class

the class of a delay pool determines how the delay is applied, ie, whether the dierent client IPs are treated seperately or as a group (or both)

class 1

a class 1 delay pool contains a single unied bucket which is used for all requests from hosts subject to the pool

class 2

a class 2 delay pool contains one unied bucket and 255 buckets, one for each host on an 8-bit network (IPv4 class C)

class 3

contains 255 buckets for the subnets in a 16-bit network, and individual buckets for every host on these networks (IPv4 class B)

Delay pools allows you to limit trac for clients or client groups, with various features:

can specify peer hosts which aren't aected by delay pools, ie, local peering or other 'free' trac (with the *no-delay* peer option).

delay behavior is selected by ACLs (low and high priority trac, sta vs students or student vs authenticated student or so on).

each group of users has a number of buckets, a bucket has an amount coming into it in a second and a maximum amount it can grow to; when it reaches zero, objects reads are deferred until one of the object's clients has some trac allowance.

any number of pools can be congured with a given class and any set of limits within the pools can be disabled, for example you might only want to use the aggregate and per-host bucket groups of class 3, not the per-network one.

This allows options such as creating a number of class 1 delay pools and allowing a certain amount of bandwidth to given object types (by using URL regular expressions or similar), and many other uses I'm sure I haven't even though of beyond the original fair balancing of a relatively small trac allocation across a large number of users.

There are some limitations of delay pools:

delay pools are incompatible with slow aborts; quick abort should be set fairly low to prevent objects being retrived at full speed once there are no clients requesting them (as the trac allocation is based on the current clients, and when there are no clients attached to the object there is no way to determine the trac allocation).

delay pools only limits the actual data transferred and is not inclusive of overheads such as TCP overheads, ICP, DNS, icmp pings, etc.

it is possible for one connection or a small number of connections to take all the bandwidth from a given bucket and the other connections to be starved completely, which can be a major problem if there are a number of large objects being transferred and the parameters are set in a way that a few large objects will cause all clients to be starved (potentially xed by a currently experimental patch).

19.8.1 How can I limit Squid's total bandwidth to, say, 512 Kbps?

```
acl all src 0.0.0.0/0.0.0.0            # might already be defined
delay_pools 1
delay_class 1 1
delay_access 1 allow all
delay_parameters 1 64000/64000         # 512 kbits == 64 kbytes per second
```

For an explanation of these tags please see the conguration le.

The 1 second buer (max = restore = 64kbytes/sec) is because a limit is requested, and no responsiveness to a busrt is requested. If you want it to be able to respond to a burst, increase the aggregate_max to a larger value, and trac bursts will be handled. It is recommended that the maximum is at least twice the restore value - if there is only a single object being downloaded, sometimes the download rate will fall below the requested throughput as the bucket is not empty when it comes to be replenished.

19.8.2 How to limit a single connection to 128 Kbps?

You can not limit a single HTTP request's connection speed. You *can* limit individual hosts to some bandwidth rate. To limit a specic host, dene an *acl* for that host and use the example above. To limit a group of hosts, then you must use a delay pool of class 2 or 3. For example:

```
acl only128kusers src 192.168.1.0/255.255.192.0
acl all src 0.0.0.0/0.0.0.0
delay_pools 1
delay_class 1 3
delay_access 1 allow only128kusers
delay_access 1 deny all
delay_parameters 1 64000/64000 -1/-1 16000/64000
```

For an explanation of these tags please see the conguration le.

The above gives a solution where a cache is given a total of 512kbits to operate in, and each IP address gets only 128kbits out of that pool.

19.8.3 How do you personally use delay pools?

We have six local cache peers, all with the options 'proxy-only no-delay' since they are fast machines connected via a fast ethernet and microwave (ATM) network.

For our local access we use a dstdomain ACL, and for delay pool exceptions we use a dst ACL as well since the delay pool ACL processing is done using "fast lookups", which means (among other things) it won't wait for a DNS lookup if it would need one.

Our proxy has two virtual interfaces, one which requires student authentication to connect from machines where a department is not paying for trac, and one which uses delay pools. Also, users of the main Unix system are allowed to choose slow or fast trac, but must pay for any trac they do using the fast cache.

Ident lookups are disabled for accesses through the slow cache since they aren't needed. Slow accesses are delayed using a class 3 delay pool to give fairness between departments as well as between users. We recognize users of Lynx on the main host are grouped together in one delay bucket but they are mostly viewing text pages anyway, so this isn't considered a serious problem. If it was we could take those hosts into a class 1 delay pool and give it a larger allocation.

I prefer using a slow restore rate and a large maximum rate to give preference to people who are looking at web pages as their individual bucket lls while they are reading, and those downloading large objects are disadvantaged. This depends on which clients you believe are more important. Also, one individual 8 bit network (a residential college) have paid extra to get more bandwidth.

The relevant parts of my conguration le are (IP addresses, etc, all changed):

```
# ACL definitions
# Local network definitions, domains a.net, b.net
acl LOCAL-NET dstdomain a.net b.net
# Local network; nets 64 - 127.  Also nearby network class A, 10.
acl LOCAL-IP dst 192.168.64.0/255.255.192.0 10.0.0.0/255.0.0.0
# Virtual i/f used for slow access
acl virtual_slowcache myip 192.168.100.13/255.255.255.255
# All permitted slow access, nets 96 - 127
acl slownets src 192.168.96.0/255.255.224.0
# Special 'fast' slow access, net 123
acl fast_slow src 192.168.123.0/255.255.255.0
# User hosts
acl my_user_hosts src 192.168.100.2/255.255.255.254
# "All" ACL
acl all src 0.0.0.0/0.0.0.0

# Don't need ident lookups for billing on (free) slow cache
ident_lookup_access allow my_user_hosts !virtual_slowcache
ident_lookup_access deny all

# Security access checks
http_access [...]

# These people get in for slow cache access
http_access allow virtual_slowcache slownets
http_access deny virtual_slowcache

# Access checks for main cache
http_access [...]

# Delay definitions (read config file for clarification)
delay_pools 2
delay_initial_bucket_level 50

delay_class 1 3
delay_access 1 allow virtual_slowcache !LOCAL-NET !LOCAL-IP !fast_slow
delay_access 1 deny all
delay_parameters 1 8192/131072 1024/65536 256/32768
```

```
        delay_class 2 2
        delay_access 2 allow virtual_slowcache !LOCAL-NET !LOCAL-IP fast_slow
        delay_access 2 deny all
        delay_parameters 2 2048/65536 512/32768
```

The same code is also used by a some of departments using class 2 delay pools to give them more exibility in giving dierent performance to dierent labs or students.

19.8.4 Where else can I nd out about delay pools?

This is also pretty well documented in the conguration le, with examples. Since people seem to loose their cong les, here's a copy of the relevant section.

```
# DELAY POOL PARAMETERS (all require DELAY_POOLS compilation option)
# -----------------------------------------------------------------------------

#  TAG: delay_pools
#       This represents the number of delay pools to be used.  For example,
#       if you have one class 2 delay pool and one class 3 delays pool, you
#       have a total of 2 delay pools.
#
#       To enable this option, you must use --enable-delay-pools with the
#       configure script.
#delay_pools 0

#  TAG: delay_class
#       This defines the class of each delay pool.  There must be exactly one
#       delay_class line for each delay pool.  For example, to define two
#       delay pools, one of class 2 and one of class 3, the settings above
#       and here would be:
#
#delay_pools 2      # 2 delay pools
#delay_class 1 2    # pool 1 is a class 2 pool
#delay_class 2 3    # pool 2 is a class 3 pool
#
#       The delay pool classes are:
#
#               class 1         Everything is limited by a single aggregate
#                               bucket.
#
#               class 2         Everything is limited by a single aggregate
#                               bucket as well as an "individual" bucket chosen
#                               from bits 25 through 32 of the IP address.
#
#               class 3         Everything is limited by a single aggregate
#                               bucket as well as a "network" bucket chosen
#                               from bits 17 through 24 of the IP address and a
#                               "individual" bucket chosen from bits 17 through
#                               32 of the IP address.
#
```

```
#       NOTE: If an IP address is a.b.c.d
#               -> bits 25 through 32 are "d"
#               -> bits 17 through 24 are "c"
#               -> bits 17 through 32 are "c * 256 + d"

#  TAG: delay_access
#       This is used to determine which delay pool a request falls into.
#       The first matched delay pool is always used, ie, if a request falls
#       into delay pool number one, no more delay are checked, otherwise the
#       rest are checked in order of their delay pool number until they have
#       all been checked.  For example, if you want some_big_clients in delay
#       pool 1 and lotsa_little_clients in delay pool 2:
#
#delay_access 1 allow some_big_clients
#delay_access 1 deny all
#delay_access 2 allow lotsa_little_clients
#delay_access 2 deny all

#  TAG: delay_parameters
#       This defines the parameters for a delay pool.  Each delay pool has
#       a number of "buckets" associated with it, as explained in the
#       description of delay_class.  For a class 1 delay pool, the syntax is:
#
#delay_parameters pool aggregate
#
#       For a class 2 delay pool:
#
#delay_parameters pool aggregate individual
#
#       For a class 3 delay pool:
#
#delay_parameters pool aggregate network individual
#
#       The variables here are:
#
#               pool            a pool number - ie, a number between 1 and the
#                               number specified in delay_pools as used in
#                               delay_class lines.
#
#               aggregate       the "delay parameters" for the aggregate bucket
#                               (class 1, 2, 3).
#
#               individual      the "delay parameters" for the individual
#                               buckets (class 2, 3).
#
#               network         the "delay parameters" for the network buckets
#                               (class 3).
#
#       A pair of delay parameters is written restore/maximum, where restore is
#       the number of bytes (not bits - modem and network speeds are usually
```

```
#       quoted in bits) per second placed into the bucket, and maximum is the
#       maximum number of bytes which can be in the bucket at any time.
#
#       For example, if delay pool number 1 is a class 2 delay pool as in the
#       above example, and is being used to strictly limit each host to 64kbps
#       (plus overheads), with no overall limit, the line is:
#
#delay_parameters 1 -1/-1 8000/8000
#
#       Note that the figure -1 is used to represent "unlimited".
#
#       And, if delay pool number 2 is a class 3 delay pool as in the above
#       example, and you want to limit it to a total of 256kbps (strict limit)
#       with each 8-bit network permitted 64kbps (strict limit) and each
#       individual host permitted 4800bps with a bucket maximum size of 64kb
#       to permit a decent web page to be downloaded at a decent speed
#       (if the network is not being limited due to overuse) but slow down
#       large downloads more significantly:
#
#delay_parameters 2 32000/32000 8000/8000 600/8000
#
#       There must be one delay_parameters line for each delay pool.

#  TAG: delay_initial_bucket_level      (percent, 0-100)
#       The initial bucket percentage is used to determine how much is put
#       in each bucket when squid starts, is reconfigured, or first notices
#       a host accessing it (in class 2 and class 3, individual hosts and
#       networks only have buckets associated with them once they have been
#       "seen" by squid).
#
#delay_initial_bucket_level 50
```

19.9 Can I preserve my cache when upgrading from 1.1 to 2?

At the moment we do not have a script which will convert your cache contents from the 1.1 to the Squid-2 format. If enough people ask for one, then somebody will probably write such a script.

If you like, you can congure a new Squid-2 cache with your old Squid-1.1 cache as a sibling. After a few days, weeks, or however long you want to wait, shut down the old Squid cache. If you want to force-load your new cache with the objects from the old cache, you can try something like this:

1. Install Squid-2 and congure it to have the same amount of disk space as your Squid-1 cache, even if there is not currently that much space free.
2. Congure Squid-2 with Squid-1 as a parent cache. You might want to enable *never_direct* on the Squid-2 cache so that all of Squid-2's requests go through Squid-1.
3. Enable the 7.5 on Squid-1.
4. Set the refresh rules on Squid-1 to be very liberal so that it does not generate IMS requests for cached objects.

5. Create a list of all the URLs in the Squid-1 cache. These can be extracted from the access.log, store.log and swap logs.
6. For every URL in the list, request the URL from Squid-2, and then immediately send a PURGE request to Squid-1.
7. Eventually Squid-2 will have all the objects, and Squid-1 will be empty.

19.10 Customizable Error Messages

Squid-2 lets you customize your error messages. The source distribution includes error messages in dierent languages. You can select the language with the congure option:

```
--enable-err-language=lang
```

Furthermore, you can rewrite the error message template les if you like. This list describes the tags which Squid will insert into the messages:

%B

URL with FTP %2f hack

%c

Squid error code

%d

seconds elapsed since request received (not yet implemented)

%e

errno

%E

strerror()

%f

FTP request line

%F

FTP reply line

%g

FTP server message

%h

cache hostname

%H

server host name

%i

client IP address

%I

server IP address

%L

contents of *err_html_text* cong option

%M

Request Method

%m

Error message returned by external auth helper

%p

URL port \#

%P

Protocol

%R

Full HTTP Request

%S

squid default signature

%s

caching proxy software with version

%t

local time

%T

UTC

%U

URL without password

%u

URL with password (Squid-2.5 and later only)

%w

cachemgr email address

%z

dns server error message

The Squid default signature is added automatically unless %s is used in the error page. To change the signature you must manually append the signature to each error page.

The default signature reads like:

```
<BR clear="all">
<HR noshade size="1px">
<ADDRESS>
Generated %T by %h (%s)
</ADDRESS>
</BODY></HTML>
```

19.11 My squid.conf from version 1.1 doesn't work!

Yes, a number of conguration directives have been renamed. Here are some of them:

cache_host

This is now called *cache_peer*. The old term does not really describe what you are conguring, but the new name tells you that you are conguring a peer for your cache.

cache_host_domain

Renamed to *cache_peer_domain*.

local_ip, local_domain

The functaionality provided by these directives is now implemented as access control lists. You will use the *always_direct* and *never_direct* options. The new *squid.conf* le has some examples.

cache_stoplist

This directive also has been reimplemented with access control lists. You will use the *no_cache* option. For example:

```
acl Uncachable url_regex cgi ?
no_cache deny Uncachable
```

cache_swap

This option used to specify the cache disk size. Now you specify the disk size on each *cache_dir* line.

cache_host_acl

This option has been renamed to *cache_peer_access* **and** the syntax has changed. Now this option is a true access control list, and you must include an *allow* or *deny* keyword. For example:

```
acl that-AS dst_as 1241
cache_peer_access thatcache.thatdomain.net allow that-AS
cache_peer_access thatcache.thatdomain.net deny all
```

This example sends requests to your peer *thatcache.thatdomain.net* only for origin servers in Autonomous System Number 1241.

units

In Squid-1.1 many of the conguration options had implied units associated with them. For example, the *connect_timeout* value may have been in seconds, but the *read_timeout* value had to be given in minutes. With Squid-2, these directives take units after the numbers, and you will get a warning if you leave o the units. For example, you should now write:

```
connect_timeout 120 seconds
read_timeout 15 minutes
```

20 httpd-accelerator mode

20.1 What is the httpd-accelerator mode?

Occasionally people have trouble understanding accelerators and proxy caches, usually resulting from mixed up interpretations of "incoming" and "outgoing" data. I think in terms of requests (i.e., an outgoing request is from the local site out to the big bad Internet). The data received in reply is incoming, of course. Others think in the opposite sense of "a request for incoming data".

An accelerator caches incoming requests for outgoing data (i.e., that which you publish to the world). It takes load away from your HTTP server and internal network. You move the server away from port 80 (or whatever your published port is), and substitute the accelerator, which then pulls the HTTP data from the "real" HTTP server (only the accelerator needs to know where the real server is). The outside world sees no dierence (apart from an increase in speed, with luck).

Quite apart from taking the load of a site's normal web server, accelerators can also sit outside rewalls or other network bottlenecks and talk to HTTP servers inside, reducing trac across the bottleneck and simplifying the conguration. Two or more accelerators communicating via ICP can increase the speed and resilience of a web service to any single failure.

The Squid redirector can make one accelerator act as a single front-end for multiple servers. If you need to move parts of your lesystem from one server to another, or if separately administered HTTP servers should logically appear under a single URL hierarchy, the accelerator makes the right thing happen.

If you wish only to cache the "rest of the world" to improve local users browsing performance, then accelerator mode is irrelevant. Sites which own and publish a URL hierarchy use an accelerator to improve other sites' access to it. Sites wishing to improve their local users' access to other sites' URLs use proxy caches. Many sites, like us, do both and hence run both.

Measurement of the Squid cache and its Harvest counterpart suggest an order of magnitude performance improvement over CERN or other widely available caching software. This order of magnitude performance improvement on hits suggests that the cache can serve as an httpd accelerator, a cache congured to act as a site's primary httpd server (on port 80), forwarding references that miss to the site's real httpd (on port 81).

In such a conguration, the web administrator renames all non-cachable URLs to the httpd's port (81). The cache serves references to cachable objects, such as HTML pages and GIFs, and the true httpd (on port 81) serves references to non-cachable objects, such as queries and cgi-bin programs. If a site's usage characteristics tend toward cachable objects, this conguration can dramatically reduce the site's web workload.

Note that it is best not to run a single *squid* process as both an httpd-accelerator and a proxy cache, since these two modes will have dierent working sets. You will get better performance by running two separate caches on separate machines. However, for compatability with how administrators are accustomed to running other servers that provide both proxy and Web serving capability (eg, CERN), the Squid supports operation as both a proxy and an accelerator if you set the `httpd_accel_with_proxy` variable to `on` inside your *squid.conf* conguration le.

20.2 How do I set it up?

First, you have to tell Squid to listen on port 80 (usually), so set the 'http_port' option:

```
http_port 80
```

Next, you need to move your normal HTTP server to another port and/or another machine. If you want to run your HTTP server on the same machine, then it can not also use port 80 (except see the next FAQ

entry below). A common choice is port 81. Congure squid as follows:

```
httpd_accel_host localhost
httpd_accel_port 81
```

Alternatively, you could move the HTTP server to another machine and leave it on port 80:

```
httpd_accel_host otherhost.foo.com
httpd_accel_port 80
```

You should now be able to start Squid and it will serve requests as a HTTP server.

If you are using Squid has an accelerator for a virtual host system, then you need to specify

```
httpd_accel_host virtual
```

Finally, if you want Squid to also accept *proxy* requests (like it used to before you turned it into an accelerator), then you need to enable this option:

```
httpd_accel_with_proxy on
```

20.3 When using an httpd-accelerator, the port number for redirects is wrong

Yes, this is because you probably moved your real httpd to port 81. When your httpd issues a redirect message (e.g. 302 Moved Temporarily), it knows it is not running on the standard port (80), so it inserts *:81* in the redirected URL. Then, when the client requests the redirected URL, it bypasses the accelerator.

How can you x this?

One way is to leave your httpd running on port 80, but bind the httpd socket to a *specic* interface, namely the loopback interface. With *Apache* <http://www.apache.org/> you can do it like this in *httpd.conf*:

```
Port 80
BindAddress 127.0.0.1
```

Then, in your *squid.conf* le, you must specify the loopback address as the accelerator:

```
httpd_accel_host 127.0.0.1
httpd_accel_port 80
```

Note, you probably also need to add an */etc/hosts* entry of 127.0.0.1 for your server hostname. Otherwise, Squid may get stuck in a forwarding loop.

21 Related Software

21.1 Clients

21.1.1 Wget

Wget <ftp://gnjilux.cc.fer.hr/pub/unix/util/wget/> is a command-line Web client. It supports HTTP and FTP URLs, recursive retrievals, and HTTP proxies.

21.1.2 echoping

If you want to test your Squid cache in batch (from a cron command, for instance), you can use the *echoping* <ftp://ftp.internatif.org/pub/unix/echoping/> program, which will tell you (in plain text or via an exit code) if the cache is up or not, and will indicate the response times.

21.2 Logle Analysis

Rather than maintain the same list in two places, please see the *Logle Analysis Scripts* </Scripts/> page on the Web server.

21.3 Conguration Tools

21.3.1 3Dhierarchy.pl

Kenichi Matsui has a simple perl script which generates a 3D hierarchy map (in VRML) from squid.conf. *3Dhierarchy.pl* <ftp://ftp.nemoto.ecei.tohoku.ac.jp/pub/Net/WWW/VRML/converter/3Dhierarchy.pl>.

21.4 Squid add-ons

21.4.1 transproxy

transproxy <http://www.transproxy.nlc.net.au/> is a program used in conjunction with the Linux Transparent Proxy networking feature, and ipfwadm, to intercept HTTP and other requests. Transproxy is written by *John Saunders* <mailto:john@nlc.net.au>.

21.4.2 Iain's redirector package

A *redirector package* <ftp://ftp.sbs.de/pub/www/cache/redirector/redirector.tar.gz> from *Iain Lea* <mailto:iain@ecrc.de> to allow Intranet (restricted) or Internet (full) access with URL deny and redirection for sites that are not deemed acceptable for a userbase all via a single proxy port.

21.4.3 Junkbusters

Junkbusters <http://internet.junkbuster.com> Corp has a copyleft privacy-enhancing, ad-blocking proxy server which you can use in conjunction with Squid.

21.4.4 Squirm

Squirm <http://squirm.foote.com.au/> is a congurable, ecient redirector for Squid by *Chris Foote* <mailto:chris@senet.com.au>. Features:

Very fast

Virtually no memory usage

It can re-read it's cong les while running by sending it a HUP signal

Interactive test mode for checking new congs

Full regular expression matching and replacement

Cong les for patterns and IP addresses.

If you mess up the cong le, Squirm runs in Dodo Mode so your squid keeps working :-)

21.4.5 chpasswd.cgi

Pedro L Orso <mailto:orso@ineparnet.com.br> has adapated the Apache's *htpasswd* <../../htpasswd/> into a CGI program called *chpasswd.cgi* <http://web.onda.com.br/orso/chpasswd.html>.

21.4.6 jesred

jesred <http://ivs.cs.uni-magdeburg.de/~elkner/webtools/jesred/> by *Jens Elkner* <mailto:elkner@wotan.cs.Uni-Magdeburg.DE>.

21.4.7 squidGuard

squidGuard <http://www.squidguard.org/> is a free (GPL), exible and ecient lter and redirector program for squid. It lets you dene multiple access rules with dierent restrictions for dierent user groups on a squid cache. squidGuard uses squid standard redirector interface.

21.4.8 Central Squid Server

The Smart Neighbour [URL disappeared] (or 'Central Squid Server' - CSS) is a cut-down version of Squid without HTTP or object caching functionality. The CSS deals only with ICP messages. Instead of caching objects, the CSS records the availability of objects in each of its neighbour caches. Caches that have smart neighbours update each smart neighbour with the status of their cache by sending ICP_STORE_NOTIFY/ICP_RELEASE_NOTIFY messages upon storing/releasing an object from their cache. The CSS maintains an up to date 'object map' recording the availability of objects in its neighbouring caches.

21.5 Ident Servers

For *Windows NT* <http://ftp.tdcnorge.no/pub/windows/Identd/>, *Windows 95/98* <http://identd.sourceforge.net/>, and *Unix* <http://www2.lysator.liu.se/~pen/pidentd/>.

22 DISKD

22.1 What is DISKD?

DISKD refers to some features in Squid-2.4 to improve Disk I/O performance. The basic idea is that each *cache_dir* has its own *diskd* child process. The diskd process performs all disk I/O operations (open, close, read, write, unlink) for the cache_dir. Message queues are used to send requests and responses between the Squid and diskd processes. Shared memory is used for chunks of data to be read and written.

22.2 Does it perform better?

Yes. We benchmarked Squid-2.4 with DISKD at the *Second IRCache Bake-O* <http://polygraph.ircache.net/Results/bakeoff-2/>. The results are also described *here*

</Benchmarking/bakeoff-02/>. At the bakeo, we got 160 req/sec with diskd. Without diskd, we'd have gotten about 40 req/sec.

22.3 How do I use it?

You need to run Squid version *2.4* </Versions/v2/2.4> or later. Your operating system must support message queues, and shared memory.

To congure Squid for DISKD, use the *–enable-storeio* option:

```
% ./configure --enable-storeio=diskd,ufs
```

22.4 FATAL: Unknown cache_dir type 'diskd'

You didn't put *diskd* in the list of storeio modules as described above. You need to run *congure* and and recompile Squid.

22.5 If I use DISKD, do I have to wipe out my current cache?

No. Diskd uses the same storage scheme as the standard "UFS" type. It only changes how I/O is performed.

22.6 How do I congure message queues?

Most Unix operating systems have message queue support by default. One way to check is to see if you have an *ipcs* command.

However, you will likely need to increase the message queue parameters for Squid. Message queue implementations normally have the following parameters:

MSGMNB

Maximum number of bytes per message queue.

MSGMNI

Maximum number of message queue identiers (system wide).

MSGSEG

Maximum number of message segments per queue.

MSGSSZ

Size of a message segment.

MSGTQL

Maximum number of messages (system wide).

MSGMAX

Maximum size of a whole message. On some systems you may need to increase this limit. On other systems, you may not be able to change it.

The messages between Squid and diskd are 32 bytes for 32-bit CPUs and 40 bytes for 64-bit CPUs. Thus, MSGSSZ should be 32 or greater. You may want to set it to a larger value, just to be safe.

We'll have two queues for each *cache_dir* – one in each direction. So, MSGMNI needs to be at least two times the number of *cache_dir*'s.

I've found that 75 messages per queue is about the limit of decent performance. If each diskd message consists of just one segment (depending on your value of MSGSSZ), then MSGSEG should be greater than 75.

MSGMNB and MSGTQL aect how many messages can be in the queues at one time. Diskd messages shouldn't be more than 40 bytes, but let's use 64 bytes to be safe. MSGMNB should be at least 64*75. I recommend rounding up to the nearest power of two, or 8192.

MSGTQL should be at least 75 times the number of *cache_dir*'s that you'll have.

22.6.1 FreeBSD

Your kernel must have

```
options         SYSVMSG
```

You can set the parameters in the kernel as follows. This is just an example. Make sure the values are appropriate for your system:

```
options         MSGMNB=8192     # max # of bytes in a queue
options         MSGMNI=40       # number of message queue identifiers
options         MSGSEG=512      # number of message segments per queue
options         MSGSSZ=64       # size of a message segment
options         MSGTQL=2048     # max messages in system
```

22.6.2 OpenBSD

You can set the parameters in the kernel as follows. This is just an example. Make sure the values are appropriate for your system:

```
option          MSGMNB=16384    # max characters per message queue
option          MSGMNI=40       # max number of message queue identifiers
option          MSGSEG=2048     # max number of message segments in the system
option          MSGSSZ=64       # size of a message segment (Must be 2^N)
option          MSGTQL=1024     # max amount of messages in the system
```

22.6.3 Digital Unix

Message queue support seems to be in the kernel by default. Setting the options is as follows:

```
options         MSGMNB="8192"     # max # bytes on queue
options         MSGMNI="40"       # # of message queue identifiers
options         MSGMAX="2048"     # max message size
options         MSGTQL="2048"     # # of system message headers
```

by *Brenden Phillips* <mailto:B.C.Phillips at massey dot ac dot nz>

If you have a newer version (DU64), then you can probably use *syscong* instead. To see what the current IPC settings are run

```
# sysconfig -q ipc
```

To change them make a le like this called ipc.stanza:

```
ipc:
        msg-max = 2048
        msg-mni = 40
        msg-tql = 2048
        msg-mnb = 8192
```

then run

```
# sysconfigdb -a -f ipc.stanza
```

You have to reboot for the change to take eect.

22.6.4 Linux

Stefan Kopsell reports that if you compile sysctl support into your kernel, then you can change the following values:

kernel.msgmnb

kernel.msgmni

kernel.msgmax

Winfried Truemper reports: The default values should be large enough for most common cases. You can modify the message queue conguration by writing to these les:

/proc/sys/kernel/msgmax

/proc/sys/kernel/msgmnb

/proc/sys/kernel/msgmni

22.6.5 Solaris

Refer to *Demangling Message Queues* <http://www.sunworld.com/sunworldonline/swol-11-1997/swol-11-insidesolar in Sunworld Magazine.

I don't think the above article really tells you how to set the parameters. You do it in */etc/system* with lines like this:

```
set msgsys:msginfo_msgmax=2048
set msgsys:msginfo_msgmnb=8192
set msgsys:msginfo_msgmni=40
set msgsys:msginfo_msgssz=64
set msgsys:msginfo_msgtql=2048
```

Of course, you must reboot whenever you modify */etc/system* before changes take eect.

22.7 How do I congure shared memory?

Shared memory uses a set of parameters similar to the ones for message queues. The Squid DISKD implementation uses one shared memory area for each cache_dir. Each shared memory area is about 800 kilobytes in size. You may need to modify your system's shared memory parameters:

SHMSEG

Maximum number of shared memory segments per process.

SHMMNI

Maximum number of shared memory segments for the whole system.

SHMMAX

Largest shared memory segment size allowed.

SHMALL

Total amount of shared memory that can be used.

For Squid and DISKD, *SHMMNI* and *SHMMNI* must be greater than or equal to the number of *cache_dir*'s that you have. *SHMMAX* must be at least 800 kilobytes. *SHMALL* must be at least *SHMMAX* 800 kilobytes multiplied by the number of *cache_dir*'s.

22.7.1 FreeBSD

Your kernel must have

```
options         SYSVSHM
```

You can set the parameters in the kernel as follows. This is just an example. Make sure the values are appropriate for your system:

```
options         SHMSEG=16       # max shared mem id's per process
options         SHMMNI=32       # max shared mem id's per system
options         SHMMAX=2097152  # max shared memory segment size (bytes)
options         SHMALL=4096     # max amount of shared memory (pages)
```

22.7.2 Digital Unix

Message queue support seems to be in the kernel by default. Setting the options is as follows:

```
options         SHMSEG="16"       # max shared mem id's per process
options         SHMMNI="32"       # max shared mem id's per system
options         SHMMAX="2097152"  # max shared memory segment size (bytes)
options         SHMALL=4096       # max amount of shared memory (pages)
```

by *Brenden Phillips* <mailto:B.C.Phillips at massey dot ac dot nz>

If you have a newer version (DU64), then you can probably use *syscong* instead. To see what the current IPC settings are run

```
# sysconfig -q ipc
```

To change them make a le like this called ipc.stanza:

```
ipc:
        shm-seg = 16
        shm-mni = 32
        shm-max = 2097152
        shm-all = 4096
```

then run

```
# sysconfigdb -a -f ipc.stanza
```

You have to reboot for the change to take eect.

22.7.3 Linux

Winfried Truemper reports: The default values should be large enough for most common cases. You can modify the shared memory conguration by writing to these les:

/proc/sys/kernel/shmall

/proc/sys/kernel/shmmax

/proc/sys/kernel/shmmni

/proc/sys/kernel/shm-use-bigpages

Stefan Kpsell reports that if you compile sysctl support into your kernel, then you can change the following values:

kernel.shmall

kernel.shmmni

kernel.shmmax

22.7.4 Solaris

Refer to *Shared memory uncovered* <http://www.sunworld.com/swol-09-1997/swol-09-insidesolaris.html> in Sunworld Magazine.

To set the values, you can put these lines in */etc/system*:

```
set shmsys:shminfo_shmmax=2097152
set shmsys:shminfo_shmmni=32
set shmsys:shminfo_shmseg=16
```

22.8 Sometimes shared memory and message queues aren't released when Squid exits.

Yes, this is a little problem sometimes. Seems like the operating system gets confused and doesn't always release shared memory and message queue resources when processes exit, especially if they exit abnormally. To x it you can "manually" clear the resources with the *ipcs* command. Add this command into your *RunCache* or *squid_start* script:

```
ipcs | grep '^[mq]' | awk '{printf "ipcrm -%s %s\n", $1, $2}' | /bin/sh
```

22.9 What are the Q1 and Q2 parameters?

In the source code, these are called *magic1* and *magic2*. These numbers refer to the number of oustanding requests on a message queue. They are specied on the *cache_dir* option line, after the L1 and L2 directories:

```
cache_dir diskd /cache1 1024 16 256 Q1=72 Q2=64
```

If there are more than Q1 messages outstanding, then Squid will intentionally fail to open disk les for reading and writing. This is a load-shedding mechanism. If your cache gets really really busy and the disks can not keep up, Squid bypasses the disks until the load goes down again.

If there are more than Q2 messages outstanding, then the main Squid process "blocks" for a little bit until the diskd process services some of the messages and sends back some replies.

Q1 should be larger than Q2. You want Squid to get to the "blocking" condition before it gets to the "refuse to open les" condition.

Reasonable values for Q1 and Q2 are 72 and 64, respectively.

23 Authentication

23.1 How does Proxy Authentication work in Squid?

Note: The information here is current for version 2.4.

Users will be authenticated if squid is congured to use *proxy_auth* ACLs (see next question).

Browsers send the user's authentication credentials in the *Authorization* request header.

If Squid gets a request and the *http_access* rule list gets to a *proxy_auth* ACL, Squid looks for the *Authorization* header. If the header is present, Squid decodes it and extracts a username and password.

If the header is missing, Squid returns an HTTP reply with status 407 (Proxy Authentication Required). The user agent (browser) receives the 407 reply and then prompts the user to enter a name and password. The name and password are encoded, and sent in the *Authorization* header for subsequent requests to the proxy.

NOTE: The name and password are encoded using "base64" (See section 11.1 of *RFC 2616* <ftp://ftp.isi.edu/in-notes/rfc2616.txt>). However, base64 is a binary-to-text encoding only, it does NOT encrypt the information it encodes. This means that the username and password are essentially "clear-text" between the browser and the proxy. Therefore, you probably should not use the same username and password that you would use for your account login.

Authentication is actually performed outside of main Squid process. When Squid starts, it spawns a number of authentication subprocesses. These processes read usernames and passwords on stdin, and reply with "OK" or "ERR" on stdout. This technique allows you to use a number of dierent authentication schemes, although currently you can only use one scheme at a time.

The Squid source code comes with a few authentcation processes. These include:

LDAP: Uses the Lightweight Directory Access Protocol

NCSA: Uses an NCSA-style username and password le.

MSNT: Uses a Windows NT authentication domain.

PAM: Uses the Linux Pluggable Authentication Modules scheme.

SMB: Uses a SMB server like Windows NT or Samba.

getpwam: Uses the old-fashioned Unix password le.

In order to authenticate users, you need to compile and install one of the supplied authentication modules, one of *the others* <http://www.squid-cache.org/related-software.html#auth>, or supply your own.

You tell Squid which authentcation program to use with the *authenticate_program* option in squid.conf. You specify the name of the program, plus any command line options if necessary. For example:

```
authenticate_program /usr/local/squid/bin/ncsa_auth /usr/local/squid/etc/passwd
```

23.2 How do I use authentication in access controls?

Make sure that your authentication program is installed and working correctly. You can test it by hand.

Add some *proxy_auth* ACL entries to your squid conguration. For example:

```
acl foo proxy_auth REQUIRED
acl all src 0/0
http_access allow foo
http_access deny all
```

The REQURIED term means that any authenticated user will match the ACL named *foo*.

Squid allows you to provide ne-grained controls by specifying individual user names. For example:

```
acl foo proxy_auth REQUIRED
acl bar proxy_auth lisa sarah frank joe
acl daytime time 08:00-17:00
acl all src 0/0
http_access allow bar
http_access allow foo daytime
http_access deny all
```

In this example, users named lisa, sarah, joe, and frank are allowed to use the proxy at all times. Other users are allowed only during daytime hours.

23.3 Does Squid cache authentication lookups?

Yes. Successful authentication lookups are cached for one hour by default. That means (in the worst case) its possible for someone to keep using your cache up to an hour after he has been removed from the authentication database.

You can control the expiration time with the *authenticate_ttl* option.

23.4 Are passwords stored in clear text or encrypted?

Squid stores cleartext passwords in itsmemory cache.

Squid writes cleartext usernames and passwords when talking to the external authentication processes. Note, however, that this interprocess communication occors over TCP connections bound to the loopback interface. Thus, its not possile for processes on other comuters to "snoop" on the authentication trac.

Each authentication program must select its own scheme for persistent storage of passwords and usernames.

23.5 How do I use the Winbind authenticators?

by *Jerry Murdock* <mailto: jmurdock at itraktech dot com>

Winbind is a recent addition to Samba providing some impressive capabilities for NT based user accounts. From Squid's perspective winbind provides a robust and ecient engine for both basic and NTLM challenge/response authentication against an NT domain controller.

The winbind authenticators have been used successfully under Linux, FreeBSD and Solaris.

23.5.1 Supported Samba Releases

Squid 2.5 uses an internal Samba interface to communicate with the winbindd daemon. It is therefore sensitive to any changes the Samba team may make to the interface.

Squid-2.5.STABLE2 supports Samba-2.2.6 to Samba-2.2.7a and hopefully later Samba versions. To use Squid-2.5.STABLE2 with Samba versions 2.2.5 or ealier the new –with-samba-sources=... congure option is required. This may also be the case with Samba-2.2.X versions later than 2.2.7a or if you have applied any winbind related patches to your Samba tree.

Squid-2.5.STABLE1 supported Samba 2.2.4 or 2.2.5 only. Use of Squid-2.5.STABLE2 or later recommended with current Samba releases.

The Samba and Squid teams are actively working together to insure future Samba stable releases will be supported, with the goal that Samba-3.X will allow Squid and other NTLM appliations to operate without depending on Samba internals. The exact details of how this will work is not yet dened.

23.5.2 Congure Samba

Build/Install Samba

Samba must be built with congure options:

```
--with-winbind
--with-winbind-auth-challenge (needed for ntlm)
```

Optionally, if building Samba 2.2.5, apply the *smbpasswd.di* <http://www.squid-cache.org/mail-archive/squid-dev/20 patch. See 23.5.2 below to determine if the patch is worthwhile.

Test Samba's winbindd

1. Edit smb.conf for winbindd functionality. The following entries in the [global] section of smbd.conf may be used as a template.

   ```
   workgroup = mydomain
   password server = myPDC
   security = domain
   winbind uid = 10000-20000
   winbind gid = 10000-20000
   winbind use default domain = yes
   ```

2. Join the NT domain as outlined in the winbindd man page for your version of samba.

3. Test winbindd functionality.

   Start nmbd (required to insure proper operation).

Start winbindd.

Test basic winbindd functionality "wbinfo -t":

```
# wbinfo -t
Secret is good
```

Test winbindd user authentication:

```
# wbinfo -a mydomain\\myuser%mypasswd
plaintext password authentication succeeded
error code was NT_STATUS_OK (0x0)
challenge/response password authentication succeeded
error code was NT_STATUS_OK (0x0)
```

NOTE: both plaintext and challenge/response should return "succeeded." If there is no "challenge/response" status returned then Samba was not built with "–with-winbind-auth-challenge" and cannot support ntlm authentication.

SMBD and Machine Trust Accounts

Samba 2.2.x

Samba's smbd daemon, while not strictly required by winbindd may be needed to manage the machine's trust account.

Well behaved domain members change the account password on a regular basis. Windows and Samba servers default to changing this password every seven days.

The Samba component responsible for managing the trust account password is smbd. Smbd needs to receive requests to trigger the password change. If the machine will be used for le and print services, then just running smbd to serve routine requests should keep everything happy.

However, in cases where Squid's winbind helpers are the only reason Samba components are running, smbd may sit idle. Indeed, there may be no other reason to run smbd at all.

There are two sample options to change the trust account. Either may be scheduled daily via a cron job to change the trust password.

UglySolution.pl <http://www.squid-cache.org/mail-archive/squid-dev/200207/att-0076/02-UglySolution.pl> is a sample perl script to load smbd, connect to a Samba share using smbclient, and generate enough dummy activity to trigger smbd's machine trust account password change code.

smbpasswd.di <http://www.squid-cache.org/mail-archive/squid-dev/200207/att-0117/01-smbpasswd.diff> is a patch to Samba 2.2.5's smbpasswd utility to allow changing the machine account password at will. It is a minimal patch simply exposing a command line interface to an existing Samba function.

Note: This patch has been included in Samba as of 2.2.6pre2.

Once patched, the smbpasswd syntax to change the password is:

```
smbpasswd -t DOMAIN -r PDC
```

Samba 3.x

The Samba team has incorporated functionality to change the machine trust account password in the new "net" command. A simple daily cron job scheduling "`net rpc changetrustpw`" is all that is needed.

23.5.3 Congure Squid

Build/Install Squid

Squid must be built with the congure options:

```
--enable-auth="ntlm,basic"
--enable-basic-auth-helpers="winbind"
--enable-ntlm-auth-helpers="winbind"
```

Test Squid without auth

Before going further, test basic Squid functionality. Make sure squid is functioning without requiring authorization.

Test the helpers

Testing the winbind ntlm helper is not really possible from the command line, but the winbind basic authenticator can be tested like any other basic helper:

```
# /usr/local/squid/libexec/wb_auth -d
/wb_auth[65180](wb_basic_auth.c:136): basic winbindd auth helper ...
mydomain\myuser mypasswd
/wb_auth[65180](wb_basic_auth.c:107): Got 'mydomain\myuser mypasswd' from squid (length: 24).
/wb_auth[65180](wb_basic_auth.c:54): winbindd result: 0
/wb_auth[65180](wb_basic_auth.c:57): sending 'OK' to squid
OK
```

The helper should return "OK" if given a valid username/password.

Edit squid.conf

1. Setup the authenticators. Add the following to enable both the winbind basic and ntlm authenticators. IE will use ntlm and everything else basic:

```
auth_param ntlm program /usr/local/squid/libexec/wb_ntlmauth
auth_param ntlm children 5
auth_param ntlm max_challenge_reuses 0
auth_param ntlm max_challenge_lifetime 2 minutes

auth_param basic program /usr/local/squid/libexec/wb_auth
auth_param basic children 5
auth_param basic realm Squid proxy-caching web server
auth_param basic credentialsttl 2 hours
```

2. Add acl entries to require authentication:

```
acl AuthorizedUsers proxy_auth REQUIRED
..
http_access allow all AuthorizedUsers
```

Test Squid with auth

1. Internet Explorer: Test browsing through squid with IE. If logged into the domain, a password prompt should NOT pop up.

Conrm the trac really is being authorized by tailing access.log. The domain \username should be present.

2. Netscape, mozilla, opera...: Test with a non-IE browser. A standard password dialog should appear.

   Entering the domain should not be required if the user is in the default domain and "winbind use default domain = yes" is set in smb.conf. Otherwise, the username must be entered in "domain\username" format.

If no usernames appear in access.log and/or no password dialogs appear in either browser, then the acl/http_access portions of squid.conf are not correct.

References

Samba Winbind Overview <http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection.html#WINBIND>

Joining a Domain in Samba 2.2.x <http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection.html#AEN1134>

winbindd man page <http://www.samba.org/samba/docs/man/winbindd.8.html>

wbinfo man page <http://www.samba.org/samba/docs/man/wbinfo.1.html>

nmbd man page <http://www.samba.org/samba/docs/man/nmbd.8.html>

smbd man page <http://www.samba.org/samba/docs/man/smbd.8.html>

smb.conf man page <http://www.samba.org/samba/docs/man/smb.conf.5.html>

smbclient man page <http://www.samba.org/samba/docs/man/smbclient.1.html>

24 Terms and Denitions

24.1 Neighbor

In Squid, *neighbor* usually means the same thing as *peer*. A neighbor cache is one that you have dened with the *cache_host* conguration option. Neighbor refers to either a parent or a sibling.

In Harvest 1.4, neighbor referred to what Squid calls a sibling. That is, Harvest had *parents* and *neighbors*. For backward compatability, the term neighbor is still accepted in some Squid conguration options.

24.2 Regular Expression

Regular expressions are patterns that used for matching sequences of characters in text. For more information, see *A Tao of Regular Expressions* <http://jmason.org/software/sitescooper/tao_regexps.html> and *Newbie's page* <http://www.newbie.org/gazette/xxaxx/xprmnt02.html>.

25 Security Concerns

25.1 Open-access proxies

Squid's default conguration le denies all client requests. It is the administrator's responsibility to congure Squid to allow access only to trusted hosts and/or users.

If your proxy allows access from untrusted hosts or users, you can be sure that people will nd and abuse your service. Some people will use your proxy to make their browsing anonymous. Others will intentionally

use your proxy for transactions that may be illegal (such as credit card fraud). A number of web sites exist simply to provide the world with a list of open-access HTTP proxies. You don't want to end up on this list.

Be sure to carefully design your access control scheme. You should also check it from time to time to make sure that it works as you expect.

25.2 Mail relaying

SMTP and HTTP are rather similar in design. This, unfortunately, may allow someone to relay an email message through your HTTP proxy. To prevent this, you must make sure that your proxy denies HTTP requests to port 25, the SMTP port.

Squid is congured this way by default. The default *squid.conf* le lists a small number of trusted ports. See the *Safe_ports* ACL in *squid.conf*. Your conguration le should always deny unsafe ports early in the *http_access* lists:

```
http_access deny !Safe_ports
(additional http_access lines ...)
```

Do NOT add port 25 to *Safe_ports* (unless your goal is to end up in the *RBL* <http://mail-abuse.org/rbl/>). You may want to make a cron job that regularly veries that your proxy blocks access to port 25.

$Id: FAQ.sgml,v 1.193 2003/08/26 16:50:13 wessels Exp $