The Hobbitware.com Intranet Project

(*You can also skip to the Table of Contents*)

In the year 1421 by the Shire Reckoning, Frodo Baggins ("Frodo") set up a small company *Hobbitware.com*, to sell stuff made by the Hobbits to Men. Along with some of his friends, he established an office in Hobbiton, where in a short period of time more than 80 bright and young hobbits were employed full−time to promote hobbitware on the world−wide−web.

Hobbitware.com was a forward looking company ably led by Frodo, and had a vibrant community of employees which thrived on the company's Intranet, managed and nurtured by Samwise Gamgee ("Sam"), the system−administrator who was a trusted friend of Frodo and had loyally supported him on many a past adventure. Sam was passionate about gardening and had brought the same passion and loving care to system administration.

These pages recount how Sam went about building and maintaining this Intranet for *Hobbitware.com*. It is hoped that these prove useful to system administrators who are looking to building an Intranet for their organisations.

> **NOTE:** These pages are not designed to be a step−by−step guide that spoon−feed the information needed to set things up. They merely serve as outlines of the processes involved and provide pointers to useful information and software. They are a direct result of the author's frustration with the unavailability of guides, at least at the time of this writing, that provided a useful summary of what is involved in setting up an Intranet for a small organisation with limited bandwidth.
>
> By no means are these software, or the way they have been used, the only way to do things. A system administrator must evolve a process that works best for his setup, hopefully using these documents to get a head start.

The Setup

It would be worthwhile to examine the general systems setup in the company before we see how Sam implemented the Intranet for it. Being a small company with limited resources, they had outsourced the hosting of their site to a service provider who managed the entire *hobbitware.com* domain for them on their servers, providing WWW, FTP and Email (SMTP/POP3) access to them, all easily administered by a secure web−based interface. Every hobbit in the company had been given an email account (with an address of the form *xyz@hobbitware.com*) with which they could stay in touch with their friends and customers. The company's site could be accessed as *www.hobbitware.com*, where customers could view their catalogue of products, place orders, track orders, etc.

The company initially had a SLIP/PPP dial−up connection (56Kbps) to their ISP, which they upgraded to 64Kbps ISDN, and later 128Kbps ISDN, as their size grew. Sam had the bright idea of using a hardware proxy, instead of wasting a machine for connecting to their ISP, which made connecting to the Internet a trivial matter for the hobbits − they just had to set the hardware proxy as their "default gateway" and they could access everything on the Internet transparently. The proxy also provided a limited amount of protection from external malicious crackers trying to connect to internal servers containing sensitive data.

The Problems

This setup was not without problems however. The hobbits used email as the primary means of communicating with each other (including for official communications) and wasted a lot of bandwidth unnecessarily when they sent mails to other hobbits *in the same office*. Moreover, as most of them had common interests (being young, unmarried, nerdy hobbits) they ended up visiting the same sites on the Internet − each hobbit browsing the same site independently of the other hobbits − which resulted in a lot of unnecessary traffic.

Pretty soon, everyone was complaining about the lack of bandwidth. Frodo was appalled that there was not much bandwidth available for essential tasks (processing customer orders, etc.) and he asked Sam to do something about the whole mess.

The Solution

Sam spent several days and nights analysing the problem and trying to come up with ways to solve it. He scoured countless Internet sites in search of a solution and was finally able to put together a proposal for an "Intranet server" that would try to eliminate extraneous traffic on their Internet link as much as possible. His idea was to cache as much information as possible locally, including web pages, common downloads, emails, DNS queries, etc. He got the go−ahead from Frodo and was given a machine to host the Intranet server (whatever it meant), with a somewhat low−end configuration, but with ample disk space. Frodo promised him a better server if his idea proved useful and reliable.

Sam wanted to get the best deal: he wanted a super−reliable system that cost him as less as possible (because his allocated budget was next to *zilch*) and was easy to maintain. And because he was new to it all, he wanted a good support base. After much evaluation, he decided to use the Linux operating system, which was a free, robust operating system supported by enthusiastic developers around the world, who contributed to the whole effort purely for the love of it. It was particularly suited to being a server for network services and ran extremely comfortably on low−end configurations. Moreover, Sam was able to find stable, mature versions of almost everything he needed to implement his vision, with a whole lot of supporting documentation. What more could a hobbit ask for?

Sam chose the Slackware distribution for installing Linux on the server, as it featured extremely stable and tested versions of all the packages and was quite flexible, when compared to other Linux distributions. He also opted to have ReiserFS filesystems on the server for better tolerance to power failures that could otherwise corrupt the regular *ext2fs* Linux filesystem and, more importantly, for better performance. In this, he was aided immensely by the SlackReiser package. Sam also decided to configure and set up a 2.4.2+ version of the Linux kernel because it featured immense performance improvements, native ReiserFS support and was even more stable than the previous versions.

With the operating system set up, Sam went about implementing the company Intranet following these steps:

1. DNS Server
2. Email Server
3. Caching Proxy Server
4. Shared Files Server (for MS Windows)

5. Intranet Web Server
6. Newsgroups Server

Alternatives

It is not as if Sam did not look at other operating systems or server software for his Intranet server − far from a Linux zealot, Sam in fact was most comfortable with Microsoft Windows NT, and the software available on it. However, with the kind of limited resources and budget that Sam had, Windows just did not work out for him. For example, the most natural choice for Sam was using Microsoft IIS on Windows NT. However, it required him to get a Windows NT Server license, which was just out of question for him. He just couldn't find *free* or cheap, yet reliable, server software on Windows. For example, the only reasonable caching web proxy servers that he could find for Windows were Microsoft Proxy Server and iPlanet Proxy Server, both of which were exorbitantly priced.

He also looked at the other operating systems for PCs, most notably FreeBSD, and though he liked the stability and general robustness of this system, he just didn't find enough resources on the Internet to help him out with the setup. And to be really frank, Sam *did* have some experience with Linux, so he was a bit biased towards it. ;−)

Conclusion

Sam's wisdom and effort helped the company make the maximum utilisation of their then scarce resources. The Intranet went beyond its original goals of helping save bandwidth and helped create a community of hobbits working for the company with a level of sharing and *camaraderie* hitherto unseen in the industry. *Hobbitware.com* became a model for other companies looking to improve resource utilisation and employee satisfaction. Frodo was immensely pleased with Sam and gave him a big bonus and a raise.

Over time the company grew in size and acquired a much fatter Internet connection, but the Intranet server continued to serve the community of hobbits merrily at work there, even though bandwidth−saving was no longer a primary requirement.

Acknowledgements

The author expresses his gratitude towards Torry Harris Business Solutions, his employers at the time of this writing, for providing the infrastructure and support needed to test the concepts given here.

Words can not do justice to the value of the effort of thousands of people from around the world who have provided us with software of such superior quality for free, and others who have written documentation that helps us use them effectively. The author hopes to contribute to this effort in his own small way.

And last, but not the least, the author wishes to specially thank the late J.R.R. Tolkien for creating "*The Lord of The Rings*" − quite possibly the best fantasy fiction ever penned − and Mark Fisher for creating the superb Encyclopedia of Arda, a fantastic guide to the worlds of Tolkien.

Back to the articles index

Pages maintained by Ranjit Mathew. (This page last updated on 28th June, 2001.)

« Previous Table of Contents Next »

The Hobbitware.com Intranet Project: DNS Server

The first order of business for Sam was to define a proper TCP/IP domain for the Intranet.

The hobbits of the company assigned IP addresses to their machines according to their whim and availability of the chosen address. The only saving grace here was that by convention, all the IP addresses had the form "`192.168.1.x`". Since IP addresses were too cumbersome to use to refer to machines, each hobbit's machine had a "`/etc/hosts`" file populated according to his needs, and sooner than later, containing incorrect entries as the IP addresses changed for some servers. And of course, everyone had a different idea of what domain name the machines belonged to: some configured their machines to belong to the `hobbiton.hobbitware.com` domain, while others configured `local.hobbitware.com` as their domain, and so on.

Sam pretty much decreed that thenceforth all the internal machines shall belong to the "`internal.hobbitware.com`" domain. All the machines were configured to use the Intranet server (which was assigned the IP address 192.168.1.1) for host name lookups (DNS Server) and the "`/etc/hosts`" files were banished. Everyone had to get their machine name registered with Sam and get an IP address, which Sam configured into the server.

Setup

For setting up the server, Sam was helped immensely by the excellent Linux DNS−HowTo. He installed the `bind` server that came with his Linux distribution. (He could have also compiled it himself using the sources he could find at this place, but there really was no point to it.) Sam still swears by the DNS−HowTo and recommends that everyone read it before attempting to configure a DNS server.

First Sam had to get the list of all the "root servers" in the DNS hierarchy − the "know−all" servers that knew how to contact the next level DNS servers. He just ran the command

```
dig @e.root−servers.net . ns >root.hints
```

to get this list in the proper format. (Of course he had to use the ISP's DNS server to get to this root server in the first place.) He saved this "`root.hints`" file in the `/var/named` folder, where he planned to keep all his zone configuration. And he made a note to himself to update this file periodically (even though it didn't change much over time).

Next, Sam needed to tell the DNS server `bind` where to look for the domain configuration information. Now `bind` looks in the file "`/etc/named.conf`" for configuration information, so Sam went about creating this file using his favourite text editor. His configuration file looked something like this:

```
options {
    directory "/var/named";
    forward first;
    forwarders {
        202.54.1.30;
        202.54.12.17;
    };
};
```

```
zone "." {
    type hint;
    file "root.hints";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "zone/127.0.0";
};

zone "internal.hobbitware.com" {
    notify no;
    type master;
    file "zone/internal.hobbitware.com";
};

zone "1.168.192.in-addr.arpa" {
    notify no;
    type master;
    file "zone/192.168.1";
};
```

Sam told `bind` to look for all the other files relative to the "`/var/named`" folder. He instructed it to first forward all non−local host name lookups to his ISP's DNS servers (202.54.1.30 and 202.54.12.17), and attempt to resolve the name itself only if these servers fail. As a typical complete DNS entry lookup takes several steps, it would have been unwise to carry it out on the slower link between the company and the ISP, as opposed to the much fatter link between the ISP itself and the Internet.

The server automatically caches all name lookups, so he didn't have to specify it explicitly. This caching tremendously helps extraneous name lookups over the slow Internet link as most hobbits used to go to the same sites.

The interesting entries were those for the zones "`internal.hobbitware.com`" and for "`1.168.192.in-addr.arpa`" − for both these zones, this server was going to be the "master" server (or the authoritative source), and the zone configuration files specified contained the respective zone data. The former zone helped resolve host names to IP addresses and the latter for reverse lookups (IP addresses to host names). This in effect was all Sam needed to define his own little internal domain.

Zones Configuration

Sam now created the folder "`/var/named/zone`". Here he placed three files named "`127.0.0`", "`internal.hobbitware.com`" and "`192.168.1`". The first file merely contained the following:

```
@   IN  SOA dns.internal.hobbitware.com. root.dns.internal.hobbitware.com. (
            1   ; Serial Number
            8H  ; Refresh Interval
            2H  ; Retry Period
            1W  ; Entry Expiry Period
            1D) ; Minimum Time-to-live
;
        NS  dns.internal.hobbitware.com.
;
1       PTR localhost.
```

This was of course as dictated by the DNS−HowTo. It was merely a fancy way of resolving "`127.0.0.1`" to "`localhost`".

The next file ("`internal.hobbitware.com`") was more interesting and contained something like this:

```
@   IN  SOA dns.internal.hobbitware.com. root.dns.internal.hobbitware.com. (
            1421041401   ; Serial Number
            8H  ; Refresh Interval
            2H  ; Retry Period
            1W  ; Entry Expiry Period
            1D) ; Minimum Time−to−live
;
        NS  dns     ; Name server for the entire domain
;
        MX  10  dns.internal.hobbitware.com     ; Primary Mail Exchanger
;
localhost   A   127.0.0.1
;
www     CNAME   dns
ftp     CNAME   dns
smtp    CNAME   dns
pop     CNAME   dns
proxy   CNAME   dns
news    CNAME   dns
;
dns     A   192.168.1.1
frodo   A   192.168.1.10
merry   A   192.168.1.11
pippin  A   192.168.1.12
sam     A   192.168.1.13
;
```

Again, in accordance with the guidelines set in the DNS HowTo, this file defined the domain `internal.hobbitware.com`. It designated this server as the primary DNS as well as the mail server for this domain. It also defined several aliases ("www", "pop", etc.) to this server so that one could access this server as `www.internal.hobbitware.com` or as `pop.internal.hobbitware.com`, etc. After that were just name to IP address mappings for hosts. As a convention, Sam decided to keep these names in alphabetic order.

Finally the last file ("`192.168.1`") contained stuff like this:

```
@   IN  SOA dns.internal.hobbitware.com. root.dns.internal.hobbitware.com. (
            1421041402   ; Serial Number
            8H  ; Refresh Interval
            2H  ; Retry Period
            1W  ; Entry Expiry Period
            1D) ; Minimum Time−to−live
;
    NS  dns.internal.hobbitware.com.
;
1   PTR dns
10  PTR frodo
11  PTR merry
12  PTR pippin
13  PTR sam
;
```

This file helped reverse DNS lookups and was quite similar to the previous one. Again as a convention, Sam decided to keep the entries here in numerically sorted (ascending) order.

Server Startup/Shutdown

Sam could very easily manage the DNS server using the `ndc` command: "`ndc start`" would stard the DNS server, while "`ndc stop`" would stop it (duh!). "`ndc restart`" restarted the DNS server.

Sam added an entry in the server startup scripts so that the DNS server software (`bind`) would automatically startup as the system booted up. Again, he found all that he needed in the DNS HowTo.

Maintenance

As new machines were added to the company, Sam had to assign a suitable name and an IP address to the machine and then update the last two files listed above in a suitable manner. He could ask the DNS server to update the zone configuration information by just issuing the command "`ndc reload`".

Once in a month, Sam updated the "`root.hints`" file as shown above. He also used to examine the files "`/var/adm/messages`" and "`/var/adm/debug`" for diagnostic or error messages, so that he could fix them before they became serious.

Conclusion

After this setup, the company had a much saner environment with respect to machine names and addresses. There was an improvment in surfing time as most of the host names were resolved locally from the DNS cache. Finally, this setup provided the foundation on which Sam could build the other services for the Intranet.

Resources

The following were some of the resources that Sam found immensely useful during the period he set up his DNS server:

1. The Linux DNS−HowTo: A very good and useful document that told Sam everything he wanted to know about the DNS system as well as how to set up a server on his machine.
2. BIND: The DNS server that Sam installed on his machine. He could have downloaded and compiled the sources available here, but he preferred to simply install the binaries available with his distribution.

« Previous Table of Contents Next »

Pages maintained by Ranjit Mathew. (This page last updated on 23rd June, 2001.)

« Previous Table of Contents Next »

The Hobbitware.com Intranet Project: Email Server

Hobbits are a merry lot and believed in sharing jokes, amusing anecdotes, etc. whenever they could. They mostly used emails for these, even though some (or all) of the recipients sat in the same office. In addition, all the official communications happened over email (hobbits did not quite like the idea of cutting down trees to produce paper and tried to aim for a "paperless office" long before the term became a fashionable buzzword). Sending and receiving these mails consumed most of the limited bandwidth that the company had, and therefore Sam had to find some way of reducing this unnecessary overhead.

The natural solution for this problem was to set up an internal mail server. Sam looked at all the available options and then settled on Sendmail because of its power and almost endless configurability, even though it was somewhat complex for a novice. He found that qmail was far simpler and much faster, but not quite as powerful as Sendmail, especially for some of the things that Sam ultimately wanted to do.

Again, he could have built Sendmail from its source code, which was freely available, but decided not to, as the binaries that came with his distribution were perfectly all right for his job.

To avoid unnecessary frustrations in the setup of the mail server, Sam made sure that he read up the excellent Linux Mail HowTos. And without the essential Sendmail FAQ, Sam would not have been able to set up anything.

Internal Mail Setup

Simple internal mail setup with Sendmail was a no−brainer − Sam just installed Sendmail and created user accounts on the server for all the hobbits of the company. Every hobbit in the company then had an internal email address of the form *username@internal.hobbitware.com*. To send and receive internal mails, each hobbit had to give the Intranet server as the SMTP and POP3 servers in the configuration for his favourite email client. The Linux distribution that Sam used had automatically installed a simple POP3 daemon, so he didn't have to install one separately. (Sam could also have used the IMAP and POP3 servers that come with the PINE distribution.)

In the DNS server configuration, the Intranet server was listed as an "MX (Mail Exchanger)" entry for the "`internal.hobbitware.com`" domain, so that the email address did not have to explicitly contain the address of the mail server (i.e. instead of having to specify "*merry@dns.internal.hobbitware.com*", they could just write "*merry@internal.hobbitware.com*"). This enabled location transparency for the addresses − Sam could now easily move the email server to a different machine if needed, without asking everyone to update their address books − all he had to do was to update the entry in the DNS server.

Sam used the Quota support in Linux (referring to the great Quota mini−HowTo of course), to limit the total size of files that the users could store on the server. This automatically limited the capacity of the mailboxes for the hobbits and prevented abuse of the server.

Linking With External Mail Accounts

For a while things looked great − the internal mail setup considerably reduced the load on the Internet link. However, now most hobbits had to deal with at least two email accounts − the internal mail account and the externally visible account − not to mention other accounts, mostly hosted by free web−based email services. The least Sam could do was to link the "official" mail accounts.

The first step for Sam was to ensure that the hobbits also get to see their "external" mails when they checked their internal mails. He used the really nifty Fetchmail program to enable this. He just installed the version that came with his distribution and then configured it. His Fetchmail configuration (stored in the home directory of the super−user as the file "`.fetchmailrc`") looked something like this:

```
set daemon 900
poll pop.hobbitware.com with protocol POP3:
no dns, aka hobbitware.com
user Frodo_Baggins%hobbitware.com is frodo here
password \x70\x61\x73\x73\x77\x6f\x72\x64
user Samwise_Gamgee%hobbitware.com is sam here
password \x70\x61\x73\x73\x77\x6f\x72\x64
```

The first line of the configuration file asked Fetchmail to run in the "daemon mode" and wake up every 900 seconds (or 15 minutes) to check user mails. The next line asked it to connect to the company's external POP3 server and retrieve the user mails. Each user's external id with the relevant password was listed alongwith the mapping to the internal mail id. Thus, any mail sent to *Frodo_Baggins@hobbitware.com* would be automatically fetched by the program and routed to the mailbox of the local user *frodo*. Frodo would then get to see the mail the next time he checks his internal mails.

Since this file contained passwords for users for the external mail accounts, it had to be protected by having read permissions only for the super−user. Furthermore, the passwords were obfuscated by using the following C program:

```
/**
 * ObPass: A very simple program that converts a given
 * word into C−style hex escape sequences for entering
 * into the runtime configuration file for Fetchmail.
 */

#include <stdio.h>
#include <unistd.h>

int main( int argc, char* argv[])
{
    char *str;

    if( argc != 1)
    {
        fprintf( stderr,
            "%s: Obfuscate a password into a C−style hex sequence\n",
            argv[0]);
        fprintf( stderr, "Usage: %s <ENTER>\n", argv[0]);

        return 1;
```

```
    } /* End if */
    else
    {
        str = getpass( "Please enter the password: ");

        for( ; *str != '\0'; str++)
        {
            printf( "\\x%x", *str);

        } /* End for */

        printf( "\n");

    } /* End else */

} /* End function main */
```

Sam added Fetchmail to the system startup scripts, so that it automatically started up when the system booted. Fetchmail ensured that the hobbits could see their internal and external emails in one place.

This setup was not without problems however. The default configuration of the SMTP server on the Intranet server did allow mails to be sent to external users, but almost always resulted in "`Sender domain must exist`" style errors. And for the mails that *did* get through, the recipients could not reply to the message as the sender address was the internal email address of the form *internal.hobbitware.com*, which was meaningless to their mail clients.

Here is where the power of Sendmail came to Sam's rescue: Sam could configure Sendmail to rewrite the sender's address so that was always of the form *hobbitware.com* with the correct mapping to the external id. Thus mails sent by *frodo@internal.hobbitware.com* were rewritten to appear to have come from *Frodo_Baggins@hobbitware.com*.

Sendmail usually reads its configuration information from the file "`/etc/sendmail.cf`", but this is usually too complex for the average person to tweak himself. Therefore it comes with scripts that automate the generation of this configuration file. Sam's Linux distribution placed these scripts and sample configuration files in the "`/usr/src/sendmail`" folder. Under the folder "`cf/cf`" in this folder, was a sample configuration file named "`linux.smtp.mc`". Sam tweaked this file until it looked something like this:

```
include(`../m4/cf.m4')
VERSIONID(`linux for smtp-only setup')dnl
OSTYPE(linux)
FEATURE(nouucp)dnl
FEATURE(genericstable, btree /etc/mail/genericstable)dnl
FEATURE(virtusertable, btree /etc/mail/virtuserstable)dnl
GENERICS_DOMAIN_FILE( /etc/mail/generics-domain)dnl
FEATURE(masquerade_envelope)dnl
FEATURE(relay_local_from)dnl
MAILER(local)dnl
MAILER(smtp)dnl
MASQUERADE_AS(hobbitware.com)
Cwinternal.hobbitware.com
Cwhobbitware.com
DSsmtp.shirenet.com
```

Of course, this was all according to the instructions given in the life-saving Sendmail FAQ. The reference to the server `smtp.shirenet.com` was there to ensure that all external mails were processed and delivered

by the ISP's SMTP server rather than the Intranet server. This process was called "relaying" and helped save the bandwidth further. Sam ran the `m4` macro processor on this file to generate the final Sendmail configuration file like this:

```
m4 <linux.smtp.mc >/etc/sendmail.cf
```

The next step was to create the "generics table" and the "virtual users map" (refer to the Sendmail FAQ) in the "`/etc/mail`" folder (Sam had to create this folder). These were in−file databases and were created using the "`updatemaildbs`" shell script that Sam wrote:

```
#!/bin/bash
makemap btree virtuserstable <virtualusermaps
makemap btree genericstable <genericusermaps
```

The "`virtualusermaps`" file looked like this:

```
Frodo_Baggins@hobbitware.com    frodo
Samwise_Gamgee@hobbitware.com   sam
@hobbitware.com %1@pop.hobbitware.com
```

And the "`genericusermaps`" file looked like this:

```
frodo   Frodo_Baggins
sam Samwise_Gamgee
```

Finally, the "`generics−domain`" file looked like this:

```
internal.hobbitware.com
```

The `updatemaildbs` shell script took these text files and generated the appropriate in−file databases (with the `.db` extension). These files together ensured that any mail sent to an external user will have the sender's address rewritten to use the external email address and any mail sent to the external address of a user will automatically be routed to his internal mailbox. Thus any mail sent to *Frodo_Baggins@hobbitware.com* would automatically be routed to his internal mailbox corresponding to the address *frodo@internal.hobbitware.com*, saving the unnecessary trip to the external SMTP server.

This was a setup as close to the ideal as possible − no mail would waste the available bandwidth if it could somehow be routed internally. And the hobbits got a single convenient "magical" mail account that made the best use of the bandwidth and could be used for both internal as well as external communications.

Maintenance

As more employees joined the company, Sam had to keep the mail server configuration up to date. He had to create the account for the new employee, update the Fetchmail configuration file, update the generic users table and the virtual users map for Sendmail. He had to ensure that the servers read the modified configuration (by sending them the "`HUP`" signal through the `kill` command).

From time to time, he looked at the messages produced via `syslog` to proactively weed out potential problems with the system.

Conclusion

The mail server setup considerably reduced the load on the available bandwidth. Further, the linking of the external and internal accounts provided a convenient single interface to the hobbits to access all of their "official" mails.

Since sending and receiving mails had become so easy, the Hobbit Resources (HR) department was able to roll in effective processes for various activities.

Resources

The following resources proved immensely useful to Sam during the setup of the mail server:

1. Linux Mail HowTos: The set of documents that helped Sam get started on the mail server setup and taught him various concepts.
2. Sendmail: The mail server implementation that Sam used. The Sendmail FAQ was an extremely useful resource without which he possibly couldn't have worked out the magic.
3. qmail: The other mail server that Sam could have used (and would have preferred) had his needs not been so complicated.
4. PINE: The email client which comes with the IMAP and POP3 servers that Sam could have used as an alternative to those that came with his distribution.
5. Fetchmail: The powerful, automated mail retriever program that enabled Sam to merge the external and internal email accounts.
6. Quota mini−HowTo: The document that Sam used to understand how to set up usage quotas for users under Linux.

« Previous Table of Contents Next »

Pages maintained by Ranjit Mathew. (This page last updated on 23st June, 2001.)

« Previous Table of Contents Next »

The Hobbitware.com Intranet Project: Caching Proxy Server

Most of the bandwidth available to the company was used up in browsing the Internet. By the very nature of their work, the hobbits found themselves visiting various sites on the WWW very often. However, most of the hobbits went to the same set of sites most of the time and therefore, there was an atrocious wastage of the bandwidth in downloading the same set of pages (and the associated images) as each hobbit individually visited a site. Sam wondered if there was a way to store the contents of these sites locally, so that precious bandwidth could be saved. This "caching" could tremendously reduce the load on the Internet connection and could provide a much better response time for content that hardly changed over time.

Sam found the solution in a caching web proxy server. And Squid was the powerful and configurable proxy server that Sam chose. Since Squid did not come bundled with his Linux distribution, Sam had to download the sources and compile it himself. However, the compilation was extremely simple and all that Sam had to do was to follow the installation instructions included with the software.

Proxy Server Setup

By default, Squid installed itself in the folder "`/usr/local/squid`". The configuration file was called "`squid.conf`" and was installed in the "`etc`" sub−folder. The configuration file was very well documented (through in−file comments) and Sam found it very easy to figure out how to configure the server.

Sam let the port on the which the proxy server listened for incoming requests, remain as the default 3128 (the `http_port` parameter). Squid needs a folder to cache the web pages, images, etc. and Sam specified one as the value to the `cache_dir` parameter (along with the total space used under this folder and the total number of level−1 and level−2 sub−directories to create in this folder). To allow everyone to use the proxy, he had to put in a line saying "`http_access allow all`". He also set the value of the "`cache_mgr`" parameter to "`sam@internal.hobbitware.com`", so that the users of the proxy servers would know who to contact in case of an error or some other problem.

Squid prefers to run as a non−root user and therefore Sam had to give the appropriate values for the "`cache_effective_user`" and the "`cache_effective_group`" parameters. He chose to run Squid under his own identity (`sam`). Of course he had to then ensure that his user−id had enough quota for storing the cached objects as well as his own mails (not to forget the big logs generated by Squid). Furthermore, the designated cache dir had to have read−write−execute permissions for his user id.

He set the value of the "`append_domain`" parameter to "`.internal.hobbitware.com`", so that hostnames that were not fully−qualified could be resolved. (There were a host of other parameters that would have enabled Sam to further tweak Squid to his needs, but he did not bother beyond this point.)

The next step was to create the folder that was designated for storing cached objects. Sam then gave the appropriate permissions (see above) to this folder. To create the cache indexing folder structure used by Squid under this folder, Sam had to run the command:

```
/usr/local/squid/bin/squid −z
```

That was it! Sam then added "`/usr/local/squid/bin/squid`" to the startup scripts so that Squid automatically started when the system booted up. Then he asked the hobbits to configure their browsers to use the intranet server as the "HTTP Proxy" and the "FTP Proxy" (both on the same port 3128). The hobbits could then immediately benefit from the immense speedups brought about by the caching proxy server.

If Sam wished, he could have added filters to the proxy server to prohibit the hobbits from visiting "offensive" sites. However, being a liberal person valuing individual freedom, he refrained from doing anything like this.

The caching proxy server brought extremely good response times for commonly visited sites, while freeing up the bandwidth for the uncommon ones. The overall response times were thus improved a lot due to the introduction of the caching proxy server.

Maintenance

Squid required a bit of maintenance from Sam from time to time. The cache access logs generated by Squid bloated rapidly and Sam had to periodically clear them. Moreover, the cache folder itself took a lot of space and Sam had to clean it up regularly.

This was a small price to pay however, for the immense benefits that Squid brought. Together with the internal mail server, the caching proxy server considerably reduced wasted bandwidth and helped improve response times.

Conclusion

The caching proxy server proved to be the single biggest saver of available bandwidth for the hobbits. It reduced the effective browsing time for the common sites to a minimum and surfing was no longer the drag it used to be.

Resources

The following resources proved immensely useful to Sam while setting up the proxy server:

1. Squid: The caching proxy server that Sam used for his Intranet Server.
2. Squid Configuration Manual: A guide that explained how to configure Squid and set it up for effective use.
3. Squid FAQ: A set of documents that helped Sam out with most of the problems he had with Squid.

« Previous Table of Contents Next »

Pages maintained by Ranjit Mathew. (This page last updated on 23rd June, 2001.)

« Previous Table of Contents Next »

The Hobbitware.com Intranet Project: File Server

Now that Sam had speeded up the Internet access considerably, the hobbits began spending more time exploring the Internet. Soon they were downloading freeware, shareware, trial programs, MP3s, etc. that very quickly saturated the bandwidth and slowed Internet access. Dismayed by this situation, Sam tried to figure out a way he could reduce the load.

Sam noticed that most hobbits downloaded the same programs, MP3 songs, etc. The bandwidth could be saved considerably if he could create a central repository of downloaded programs and songs, where hobbits could check for a file before attempting to download it. If a hobbit downloaded something useful that he thought could be of use to others, he could put it into the repository so that others could save themselves the trouble of having to download it. The repository could be organised into categories that could make searching simpler and faster.

Most of the hobbits worked on Microsoft Windows NT based computers, while some worked with Microsoft Windows 98. The most natural way to access a repository on these machines was using Windows shares (using NetBIOS). Sam wanted to set up the repository on his Intranet server that was running Linux. He wondered if there was a way the Windows machines could access this server as if it were another Windows machine.

Here is where the superlative Samba came to Sam's rescue. He read up the excellent book Using Samba, and of course, the online manual pages and the central repository was up quite soon. He also read the Linux SMB HOWTO, but it was not necessary this time − the other sources of information were more than sufficient for his purpose.

Setup

The workstations in the company did not belong to a particular Windows Domain − by convention, all of the workstations belonged to the "`HOBBITWARE`" Windows WorkGroup. This considerably eased the effort needed to set up the file server.

Sam downloaded Samba 2.2.0 sources from one of the mirrors, and compiled and installed it. By default, it installed itself in "`/usr/local/samba`". The configuration file "`smb.conf`", had to be provided in the "`lib`" sub−folder (he had to write one himself since there was no default, though there were a lot of sample configuration files).

Sam wrote the configuration file as given in the manual pages. There were a lot of options he could play with and tweak. Sam always had the excellent Samba tool "`testparm`" that could verify the configuration file after a modification. After a bit of experimentation, Sam came up with the following working Samba configuration file:

```
[global]
    workgroup = HOBBITWARE
    security = share
    server string = Hobbitware Intranet Server
```

```
    lm announce = yes
    lm interval = 900
    netbios name = INSERVER

[Downloads]
    comment = Repository of downloaded files
    path = /dump/Downloads
    guest ok = yes
    read only = yes

[Uploads]
    comment = Upload your files here
    path = /dump/Uploads
    guest ok = yes
    writeable = yes
```

This configuration file made the Intranet server advertise itself over NetBIOS as "`INSERVER`" belonging to the common workgroup. This enabled it to be visible in the "Network Neighbourhood" of the Windows workstations used by the hobbits.

The server exposed two shares − "Downloads", through which the hobbits could access the repository, and "Uploads", through which they could submit newly downloaded files. Sam requested them to supply a short description and a suggested category for cataloguing, whenever they uploaded a file. Sam periodically scanned the uploads area and moved the files into the appropriate category, except on the rare occasions that he rejected a file.

After Sam was satisfied with the way it worked, he added the Samba daemons "`/usr/local/samba/bin/smbd`" and "`/usr/local/samba/bin/nmbd`" to the system startup scripts.

Maintenance

Aside from the maintenance of the repository itself, Samba required very little interference from Sam. He was very happy with the way it performed without adversely affecting the performance of the Intranet server itself.

Conclusion

The common repository solution enabled the hobbits to rapidly get the files they wanted without having to wait for excruciatingly slow downloads, if the file was already there. The file server worked perfectly and once again, Internet access was speeded up.

Resources

The following resources proved immensely helpful to Sam during the file server setup:

1. Samba: The excellent server solution for Linux that helped the Intranet server masquerade as a Windows machine. Sam used the version 2.2.0 of this software.
2. Linux SMB HOWTO: A good set of documents that explains how to set up Samba on Linux.
3. Using Samba: The excellent book that explained how Windows shares worked, how Samba worked and how to set it up properly. Sam found this book immensely useful and practical.

« Previous Table of Contents Next »

Pages maintained by Ranjit Mathew. (This page last updated on 23rd June, 2001.)

« Previous Table of Contents Next »

The Hobbitware.com Intranet Project: Web Server

Some time after Sam set up the basic Intranet Server, Frodo asked him if he could set up the internal web server. The Hobbit Resources (HR) department was in the process of preparing an Employee Handbook that was supposed to answer most of the questions that employees had about company processes, practices, policies, etc. For maximum flexibility, Frodo wanted it to be in HTML, and if could be hosted on a suitable web server, easily accessible via any browser.

The most logical choice for hosting the internal web server was the Intranet Server, provided Sam could find a low overhead (in terms of processing power and storage needed) web server. And Sam found in the Apache Web Server, the perfect solution.

Setup

Sam just installed the Apache package that came with his Linux distribution. This package installed the Apache Web Server version 1.3.12 into the folder "`/var/lib/apache`" on the Intranet Server.

Apache ran right "out of the box" the way it was installed. Sam could conveniently run "`/var/lib/apache/sbin/apachectl start`" to start the web server and "`/var/lib/apache/sbin/apachectl stop`" to stop it. The "document root" was set to "`/var/lib/apache/htdocs`", where there was a welcome page and Apache documentation, including the manuals.

Sam read up the Apache manuals and quickly understood what to do to achieve what he wanted. The Apache server read its configuration from the file "`conf/httpd.conf`" in the folder into which it was installed. To Sam's delight, he found that the default configuration file already had reasonable values for most of the options and was very well documented by lots of useful comments placed in the appropriate sections of the file. This considerably eased his task.

The first thing to do was to set up the "document root" − the folder which contained all the HTML files, images, etc. for the internal web site. He located and modified the "`DocumentRoot`" directive and set it to "`/dump/LocalSite`" − the folder where he intended to put all the files comprising the internal web site. Sam created a simple welcome page named "`index.html`" and put it into this folder. (By giving it this name, Sam ensured that Apache serves the user this page when the user does not explicitly specify a file name in the URL.)

Sam changed the "`ServerAdmin`" directive to "`sam@internal.hobbitware.com`", so that the hobbits could know (as if they did not!) who to contact in case of problems with the web server.

Next, Sam changed the "`ServerName`" directive to send "`www.internal.hobbitware.com`" as the host name to the user browsers. Sam was confident that this would work properly, as he had already set up the internal DNS server appropriately.

Though not strictly needed, Sam changed the "`HostnameLookups`" to "`On`", so that Apache would log host names (instead of mere IP addresses) in its error and access logs.

To properly isolate logs created by Apache, Sam created the folder "`logs`" within the folder where Apache was installed ("`/var/lib/apache`"). He then set the "`ErrorLog`" directive to "`/var/lib/apache/logs/error_log`" and the "`CustomLog`" directive to "`/var/lib/apache/logs/access_log common`".

The last thing Sam needed to do was to add the command "`/var/lib/apache/sbin/apachectl start`" to the system startup scripts, so that the Apache web server was started whenever the Intranet Server came up.

After these basic and simple configuration changes, Apache was ready to serve the internal web site. Sam took the help of a few hobbits to create a better welcome page with more useful content. He took the Employee Handbook from the Hobbit Resources (HR) department and put it into a suitable folder, and created a link from the welcome page to the Handbook. The hobbits could very easily access the internal web site by typing in the URL "`http://www.internal.hobbitware.com/`" into their browsers.

Maintenance

Apache did not require much by way of maintenance. It performed very smoothly and efficiently. Every now and then, Sam examined the error logs to see if he could proactively weed out any problems with the web site.

The maintenance of the internal web site was taken over by enthusiastic volunteers who helped organise the content and kept it updated.

Conclusion

Apache performed very well and was quite efficient. The Employee Handbook proved to be very popular and soon the internal web site was expanded to serve commonly needed information, like technical documentation on the areas that the hobbits worked with, employee profiles, projects information, etc. Frodo was quite pleased with the result and congratulated Sam on a job well done.

Resources

The following resources proved immensely helpful to Sam while he set up the internal web server:

1. Apache Web Server: The superlative web server that Sam used for serving the internal web site. Its efficiency and ease of configuration considerably helped Sam in his job.
2. Apache Server Documentation: Online Apache server documents that helped Sam figure out what to change and how.
3. Apache Overview HOWTO: The Linux HOWTO document that Sam consulted for general information on Apache.

« Previous Table of Contents Next »

Pages maintained by Ranjit Mathew. (This page last updated on 23rd June, 2001.)

« Previous Table of Contents Next »

The Hobbitware.com Intranet Project: Newsgroups Server

Extremely pleased by the benefits brought about by the Intranet Server, Frodo asked Sam if he could better it. Kicked by his idea of an active community of the hobbits working for him, Frodo wanted them to be able to air their views, discuss issues, share news, etc. in a forum accessible to all.

Sam found the equivalent of forums in "newsgroups". With a suitable server, he could host his own newsgroups and even connect to external servers on the network called USENET and get newsgroups hosted by them. For example, the hobbits regularly checked out a few of these newsgroups, and without adequate caching of these groups, the bandwidth was unnecessarily wasted as each hobbit individually read a newsgroup.

After some research, Sam decided to use the excellent INN (InterNetNews) server. He downloaded the source code for version 2.3.2 of this software and compiled it himself after customising it a bit. Since he was quite new to this, he decided to take help from the INN FAQ and Elena's INN pages, and of course, the manual pages that came with the software itself.

Hosting Internal Forums

After INN was compiled and installed, Sam followed the excellent step−by−step configuration document (a text file called "`INSTALL`") that came with it to set up the forums. Instead of using the default location, he chose to install it in "`/dump/news`", since he had a separate high−capacity hard disk mounted at "`/dump`".

As recommended by the document, Sam verified that there was a user named "`news`" belonging to a group called "`news`" on the system. He didn't have to do anything special, as his Linux distribution already came with this user set up appropriately. He just made the home directory for this user the same as the folder where he installed INN. This was purely for convenience, as he had to do everything as this user rather than the root super−user. All of INN's configuration files were inside a folder called "`etc`" within the main installation folder.

Sam chose the "CNFS" format for article storage for the various advantages it had over the other methods and because it matched his requirements closely. The document clearly explained each method and its advantages as well as disadvantages.

Sam first had to edit the "`inn.conf`" file. This file was already filled in with reasonable defaults for most parameters and Sam just had to enter information like the organisation name, his email address to redirect complaints to, the domain for the server, etc. He also increased the expiry period for articles to 30 days instead of the default 10 days.

Sam skipped over the next few recommended configuration steps as he first wanted to set up an internal newsgroups server. He updated the "`incoming.conf`" file to list all the names of the Intranet Server for the self−referential peer entry named "`ME`".

Since he was using CNFS, Sam next had to edit the "`cycbuff.conf`" file. He configured this appropriately, setting aside two storage areas − one for hosting internal forums and another for caching external newsgroups. He updated the "`storage.conf`" file to reflect this configuration.

Sam then had to update the "`expire.ctl`" file to reflect the fact that he wanted articles to be retained for 30 days instead of the default 10 days.

The next important configuration file to update was the "`readers.conf`" file that told INN how to respond to the newsreader clients that the hobbits used. Apart from the default entry, Sam added the following entry:

```
auth "Hobbitware_Employees" {
    hosts: "192.168.1.*,*.internal.hobbitware.com"
    default: "<Hobbit>"
    default-domain: "internal.hobbitware.com"
}
access "Hobbitware_Employees" {
    users: "<Hobbit>@internal.hobbitware.com"
    newsgroups: "hobbitware.*,ibm.*,comp.*,alt.*,rec.*,weblogic.*"
    newsmaster: "sam@internal.hobbitware.com"
    access: RPA
}
```

This effectively allowed all hobbits accessing the server to read and post articles to the newsgroups. Apart from the default "`junk`" and "`control.*`" newsgroups, Sam wanted the server to host all internal newsgroups under the "`hobbitware.*`" hierarchy, while importing a few other newsgroups relevant to the hobbits' work. Since he had not listed the "`junk`" and "`control.*`" newsgroups explicitly in the entry above, these newsgroups were not visible to the newsreaders used by the hobbits − these were administrative groups used by INN, not meant for general use.

After this, Sam created the actual files to be used by the CNFS article storage format, using the "`dd`" command as explained by the document. Then he initialised the INN history database by using the "`makedbz`" command as recommended by the document. He also created the separate `syslog` files used by INN, as explained by the document.

Finally, Sam set up scripts to automate certain jobs for INN. For example, he had to set up a `cron` job that ran the "`news.daily`" script once in the middle of the night every day, that performed daily server maintenance. He added a call to the "`rc.news`" script to the system startup scripts, so that the news server was automatically started when the machine came up. All of this was clearly explained in the document.

Sam started the server manually by invoking the "`rc.news`" script. He then used the "`ctlinnd`" command to add some locally hosted newsgroups to the server.

The hobbits could then use their favourite newsreader software, usually the same as their e−mail clients, to connect to these groups and read and post articles.

Linking To External Newsgroups

After he had successfully configured internal forums, Sam set about configuring local caches of externally hosted newsgroups. To optimally use the available bandwidth, and to avoid unnecessary connections to external news servers, Sam decided to use a separate newsgroups updater called SUCK, rather than configuring INN to directly retrieve the newsfeed from these servers. Besides, some of these servers only accepted a "POST" from a newsreader and rejected incoming direct newsfeeds. For some reason, Sam could not connect to the "official" site and had to get SUCK from sunsite.unc.edu. (He could have also retrieved it from tsx−11.mit.edu.) Sam used version 4.2.5 of this software that worked perfectly with the version of INN he was using. He downloaded the source code and compiled it himself.

Sam decided to initially have local caches of the following newsgroups, based on the interests of the hobbits:

1. `comp.lang.java.programmer`
2. `comp.lang.javascript`
3. `comp.text.xml`
4. `ibm.software.websphere.application-server`
5. `weblogic.developer.interest`

Sam first added these groups to the INN server by using the "`ctlinnd`" command with the "`newgroup`" parameter. The only problem was that these groups were served by three different newsgroup servers, one of them being the newsgroup server provided by their ISP. After reading the manual pages that came with SUCK and after a bit of experimentation, Sam had the answer.

Sam created a file named "`myisp.cfg`" that contained options for SUCK to connect to their ISP's newsgroup server and download newsgroups. This file looked like this:

```
#
# SUCK config file for downloading from news.myisp.com
#
# Contact specified local host for the active groups list.
-A
-hl 192.168.1.1
#
# Clean up after exit.
-c
#
# Fresh connection after every so-many messages
-C 99
#
# Directories for storing stuff
-dd /dump/news/ext_feed/data
-dm /dump/news/ext_feed/multifile
-dt /dump/news/ext_feed/tmp
#
# Explicitly locate the history file for INN.
-HF /dump/news/db/history
#
# Extension to be used for files related to this server.
-p .myisp
#
# Batch processing of posts and feed it to the localhost.
-bp
#
# Be quiet.
-s -q
#
# END.
#
```

"`news.myisp.com`" was the newsgroup server provided by the ISP. Sam designated the folder "`/dump/news/ext_feed`" and sub-folders within it to serve as working folders for SUCK, as given in the file above. In the "`data`" sub-folder, Sam created a file called "`active-ignore.myisp`" that contained the groups that SUCK should avoid downloading while connecting to this server. This file looked something like this:

```
junk
control.*
hobbitware.*
ibm.*
weblogic.*
```

It effectively told SUCK to get all groups, except those internal to the company and those hosted by IBM and BEA. Sam then created similar pairs of these files for the newsgroup servers of IBM and BEA, that respectively excluded all but the groups that were to be retrieved from these servers.

SUCK came with a script named "`getnews.inn`" that downloaded articles from a remote server and posted articles, if any, to the server using the `rpost` program that came with the SUCK distribution. Sam modified this script a bit to reflect his setup and to accept the remote server name and the server configuration file as parameters. The script had code that recovered gracefully in the event that the remote server was down for some reason. An important modification that Sam made to this script was to make it use the "`put.news.sm`" output filter that came with it, as he was using the CNFS method of storage for articles − the default filter "`put.news`" assumed that each article was stored in a separate file, which was not a valid assumption for CNFS.

Sam then updated the `crontab` for the `news` user so that the "`getnews.inn`" script was called with these configuration files twice every day, for each of the remote servers. All of this was explained properly in the documentation that accompanied SUCK.

Sam had to update the "`newsfeeds`" INN configuration file so that the articles locally posted by the hobbits to external newsgroups were uploaded to the appropriate remote servers whenever SUCK connected to them, rather than immediately, as would have been the default behaviour for INN. After consulting the relevant manual pages, Sam added the following lines at the end of this configuration file:

```
# IBM's news.software.ibm.com for the WebSphere and IBM HTTP
# Server groups
ibm/news.software.ibm.com\
        :ibm.*\
        :Ae,Tf,Wnm:

# BEA's newsgroups.bea.com for the WebLogic groups
bea/newsgroups.bea.com\
        :weblogic.*\
        :Ae,Tf,Wnm:

# MyISP's news.myisp.com for other groups
myisp/news.myisp.com\
        :comp.*\
        :Ae,Tf,Wnm:
```

As a result, INN updated files named "`ibm`", "`bea`" and "`myisp`" respectively in the "`spool/outgoing`" sub−folder of the INN installation folder, with the indices of the articles to be uploaded. Whenever SUCK connected to a remote server, it posted the articles indicated in the respective file and "flushed" the file after successfully posting them. The file names indicated the "server codes", that were also used to name the respective SUCK configuration files (with a "`.cfg`" extension) mentioned earlier − this (artificial) relation enabled the "`getnews.inn`" script to correctly locate all the files needed to synchronise with a remote news server.

The effect of this setup was that SUCK automatically synchronised the local caches of these newsgroups twice every day, connecting to each newsgroup server in turn. The hobbits could now directly connect to the Intranet Server to get cached copies of these groups with near instant connectivity and even post articles to

them.

Maintenance

Sam regularly checked the log files created by INN to proactively weed out any potential problems with the server. Since Sam was using the CNFS article storage format, that rotated articles within a fixed−size buffer file, he did not have to worry about space taken up by the articles themselves.

The `ctlinnd` program allowed Sam to control various parameters related to the INN server while it was running, as well as add or remove newsgroups. Sam also regularly checked the mails for the user `news` as INN sent problem reports and general reports to this user's mailbox.

SUCK did not require much by way of maintenance, except for occasional errors that were reported when his ISP's newsgroup server went down.

Conclusion

The newsgroups proved immensely popular within the company and soon had a very active community taking part in discussions, debates, gossip, etc. Frodo was immensely pleased with the results and heaped praises on Sam.

The caching of the external newsgroups allowed the hobbits to quickly find solutions to their problems and benefit from the experiences of the people from around the world.

Resources

The following resources proved immensely useful to Sam as he set up the newsgroup server:

1. INN: The InterNetNews server software that hosted the newsgroups. Sam used version 2.3.2 of this software. The manual pages, and the installation document in particular, accompanying this software proved really useful in setting it up properly.
2. INN FAQ: This document helped Sam find answers to some of the questions he had as well as find solutions to some of the problems he was facing.
3. Elena's INN Pages: A set of documents that proved immensely helpful to Sam in understanding concepts related to USENET newsgroups and setting up a server to host them.
4. SUCK: The set of tools that allowed Sam to optimally download and synchronise local caches of external newsgroups. Since he could not connect to the main site, he had to get it from sunsite.unc.edu, though he could have also got it from tsx−11.mit.edu. Sam used version 4.2.5 of this software.

« Previous Table of Contents Next »

Pages maintained by Ranjit Mathew. (This page last updated on 12th August, 2001.)