Usenet News HOWTO

Shuvam Misra (usenet at starcomsoftware dot com)

Revision History

Revision 2.1	2002−08−20	Revised by: sm
New sections on Security and Software History, lots of other small additions and cleanup		
Revision 2.0	2002−07−30	Revised by: sm
Rewritten by new authors at Starcom Software		
Revision 1.4	1995−11−29	Revised by: vs
Original document; authored by Vince Skahan.		

Table of Contents

Table of Contents

Table of Contents

1. What is the Usenet?

1.1. Discussion groups

The Usenet is a huge worldwide collection of discussion groups. Each discussion group has a name, *e.g.* `comp.os.linux.announce`, and a collection of messages. These messages, usually called *articles*, are posted by readers like you and me who have access to Usenet servers, and are then stored on the Usenet servers.

This ability to both read and write into a Usenet newsgroup makes the Usenet very different from the bulk of what people today call ``the Internet.'' The Internet has become a colloquial term to refer to the World Wide Web, and the Web is (largely) read−only. There are online discussion groups with Web interfaces, and there are mailing lists, but Usenet is probably more convenient than either of these for most large discussion communities. This is because the articles get replicated to your local Usenet server, thus allowing you to read and post articles without accessing the global Internet, something which is of great value for those with slow Internet links. Usenet articles also conserve bandwidth because they do not come and sit in each member's mailbox, unlike email based mailing lists. This way, twenty members of a mailing list in one office will have twenty copies of each message copied to their mailboxes. However, with a Usenet discussion group and a local Usenet server, there's just one copy of each article, and it does not fill up anyone's mailbox.

Another nice feature of having your own local Usenet server is that articles stay on the server even after you've read them. You can't accidentally delete a Usenet articles the way you can delete a message from your mailbox. This way, a Usenet server is an *excellent* way to archive articles of a group discussion on a local server without placing the onus of archiving on any group member. This makes local Usenet servers very valuable as archives of internal discussion messages within corporate Intranets, provided the article expiry configuration of the Usenet server software has been set up for sufficiently long expiry periods.

1.2. How it works, loosely speaking

Usenet news works by the reader first firing up a Usenet news program, which in today's GUI world will highly likely be something like Netscape Messenger or Microsoft's Outlook Express. There are a lot of proven, well−designed character−based Usenet news readers, but a proper review of the user agent software is outside the scope of this HOWTO, so we will just assume that you are using whatever software you like. The reader then selects a Usenet newsgroup from the hundreds or thousands of newsgroups which are hosted by her local server, and accesses all unread articles. These articles are displayed to her. She can then decide to respond to some of them.

When the reader writes an article, either in response to an existing one or as a start of a brand−new thread of discussion, her software *posts* this article to the Usenet server. The article contains a list of newsgroups into which it is to be posted. Once it is accepted by the server, it becomes available for other users to read and respond to. The article is automatically *expired* or deleted by the server from its internal archives based on expiry policies set in its software; the author of the article usually can do little or nothing to control the expiry of her articles.

A Usenet server rarely works on its own. It forms a part of a collection of servers, which automatically exchange articles with each other. The flow of articles from one server to another is called a *newsfeed*. In a simplistic case, one can imagine a worldwide network of servers, all configured to replicate articles with each other, busily passing along copies across the network as soon as one of them receives a new articles posted by a human reader. This replication is done by powerful and fault−tolerant processes, and gives the Usenet network its power. Your local Usenet server literally has a copy of all current articles in all relevant newsgroups.

1.3. About sizes, volumes, and so on

Any would−be Usenet server administrator or creator *must* read the "Periodic Posting about the basic steps involved in configuring a machine to store Usenet news," also known as the Site Setup FAQ, available from `ftp://rtfm.mit.edu/pub/usenet/news.answers/usenet/site-setup` or `ftp://ftp.uu.net/usenet/news.answers/news/site-setup.Z`. It was last updated in 1997, but trends haven't changed much since then, though absolute volume figures have.

If you want your Usenet server to be a repository for all articles in all newsgroups, you will probably not be reading this HOWTO, or even if you do, you will rapidly realise that anyone who needs to read this HOWTO may not be ready to set up such a server. This is because the volumes of articles on the Usenet have reached a point where very specialised networks, very high end servers, and large disk arrays are required for handling such Usenet volumes. Those setups are called ``carrier−class'' Usenet servers, and will be discussed a bit later on in this HOWTO. Administering such an array of hardware may not be the job of the new Usenet administrator, for which this HOWTO (and most Linux HOWTO's) are written.

Nevertheless, it may be interesting to understand what volumes we are talking about. Usenet news article volumes have been doubling every fourteen months or so, going by what we hear in comments from carrier class Usenet administrators. In the beginning of 1997, this volume was 1.2 GBytes of articles a day. Thus, the volumes should have roughly done five doublings, or grown 32 times, by the time we reach mid−2002, at the time of this writing. This gives us a volume of 38.4 GBytes per day. Assume that this transfer happens using uncompressed NNTP (the norm), and add 50% extra for the overheads of NNTP, TCP, and IP. This gives you a raw data transfer volume of 57.6 GBytes/day or about 460 Gbits/day. If you have to transfer such volumes of data in 24 hours (86400 seconds), you'll need raw bandwidth of about 5.3 Mbits per second just to *receive all these articles*. You'll need more bandwidth to send out feeds to other neighbouring Usenet servers, and then you'll need bandwidth to allow your readers to access your servers and read and post articles in retail quantities. Clearly, these volume figures are outside the network bandwidths of most corporate organisations or educational institutions, and therefore only those who are in the business of offering Usenet news can afford it.

At the other end of the scale, it is perfectly feasible for a small office to subscribe to a well−trimmed subset of Usenet newsgroups, and exclude most of the high−volume newsgroups. Starcom Software, where the authors of this HOWTO work, has worked with a fairly large subset of 600 newsgroups, which is still a tiny fraction of the 15,000+ newsgroups that the carrier class services offer. Your office or college may not even need 600 groups. And our company had excluded specific high−volume but low−usefulness newsgroups like the `talk`, `comp.binaries`, and `alt` hierarchies. With the pruned subset, the total volume of articles per day may amount to barely a hundred MBytes a day or so, and can be easily handled by most small offices and educational institutions. And in such situations, a single Intel Linux server can deliver excellent performance as a Usenet server.

Then there's the *internal* Usenet service. By internal here, we mean a private set of Usenet newsgroups, not a private computer network. Every company or university which runs a Usenet news service creates its own hierarchy of internal newsgroups, whose articles never leave the campus or office, and which therefore do not consume Internet bandwidth. These newsgroups are often the ones most hotly accessed, and will carry more *internally generated* traffic than all the ``public'' newsgroups you may subscribe to, within your organisation. After all, how often does a guy have something to say which is relevant to the world at large, unless he's discussing a globally relevant topic like ``Unix rules!''? If such internal newsgroups are the focus of your

Usenet servers, then you may find that fairly modest hardware and Internet bandwidth will suffice, depending on the size of your organisation.

The new Usenet server administrator has to undertake a sizing exercise to ensure that he does not bite off more than he, or his network resources, can chew. We hope we have provided sufficient information for him to get started with the right questions.

2. Principles of Operation

Here we discuss the basic concepts behind the operation of a Usenet news system.

2.1. Newsgroups and articles

A Usenet news article sits in a file or in some other on−disk data structure on the disks of a Usenet server, and its contents look like this:

```
Xref: news.starcomsoftware.com starcom.tech.misc:211 starcom.tech.security:452
Newsgroups: starcom.tech.misc,starcom.tech.security
Path: news.starcomsoftware.com!purva!shuvam
From: Shuvam <shuvam@starcomsoftware.com>
Subject: "You just throw up your hands and reboot" (fwd)
Content-Type: TEXT/PLAIN; charset=US-ASCII
Distribution: starcom
Organization: Starcom Software Pvt Ltd, India
Message-ID: <Pine.LNX.4.31.0107022153490.30462-100000@starcomsoftware.com>
Mime-Version: 1.0
Date: Mon, 2 Jul 2001 16:27:57 GMT

Interesting quote, and interesting article.

Incidentally, comp.risks may be an interesting newsgroup to follow. We
must be receiving the feed for this group on our server, since we
receive all groups under comp.*, unless specifically cancelled. Check it
out sometime.

comp.risks tracks risks in the use of computer technology, including
issues in protecting ourselves from failures of such stuff.

Shuvam

> Date: Thu, 14 Jun 2001 08:11:00 -0400
> From: "Chris Norloff" <cnorloff@norloff.com>
> Subject: NYSE: "Throw up your hands and reboot"
>
> When the New York Stock Exchange computer systems crashed for 85
> minutes (8 Jun 2001), Andrew Brooks, chief of equity trading at
> Baltimore mutual fund giant T. Rowe Price, was quoted as saying "Hey,
> we're all subject to the vagaries of technology. It happens on your
> own PC at home. You just throw up your hands and reboot."
>
> http://www.washingtonpost.com/ac3/ContentServer?articleid=A42885-2001Jun8&pagename=article
>
> Chris Norloff
>
>
> This is from --
>
> From: risko@csl.sri.com (RISKS List Owner)
> Newsgroups: comp.risks
> Subject: Risks Digest 21.48
> Date: Mon, 18 Jun 2001 19:14:57 +0000 (UTC)
> Organization: University of California, Berkeley
>
> RISKS-LIST: Risks-Forum Digest  Monday 19 June 2001
> Volume 21 : Issue 48
```

```
>
>    FORUM ON RISKS TO THE PUBLIC IN COMPUTERS AND RELATED SYSTEMS (comp.risks)
>    ACM Committee on Computers and Public Policy,
>    Peter G. Neumann, moderator
>
> This issue is archived at <URL:http://catless.ncl.ac.uk/Risks/21.48.html>
> and by anonymous ftp at ftp.sri.com, cd risks .
>
```

A Usenet article's header is very interesting if you want to learn about the functioning of the Usenet. The `From`, `Subject`, and `Date` headers are familiar to anyone who has used email. The `Message-ID` header contains a unique ID for each message, and is present in each email message, though not many non−technical email users know about it. The `Content-Type` and `Mime-Version` headers are used for MIME encoding of articles, attaching files and other attachments, and so on, just like in email messages.

The `Organisation` header is an informational header which is supposed to carry some information identifying the organisation to which the author of the article belongs. What remains now are the `Newsgroups`, `Xref`, `Path` and `Distributions` headers. These are special to Usenet articles and are very important.

The `Newsgroups` header specifies which newsgroups this article should belong to. The `Distributions` header, sadly under−utilised in today's globalised Internet world, allows the author of an article to specify how far the article will be re−transmitted. The author of an article, working in conjunction with well−configured networks of Usenet servers, can control the ``radius'' of replication of his article, thus posting an article of local significance into a newsgroup but setting the `Distribution` header to some suitable setting, *e.g.* `local` or `starcom`, to prevent the article from being relayed to servers outside the specified domain.

The `Xref` header specifies the precise ***article number*** of this article in each of the newsgroups in which it is inserted, for the current server. When an article is copied from one server to another as part of a newsfeed, the receiving server throws away the old `Xref` header and inserts its own, with its own article numbers. This indicates an interesting feature of the Usenet system: each article in a Usenet server has a unique number (an integer) for each newsgroup it is a part of. Our sample above has been added to two newsgroups on our server, and has the article numbers 211 and 452 in those groups. Therefore, any Usenet client software can query our server and ask for article number 211 in the newsgroup `starcom.tech.misc` and get this article. Asking for article number 452 in `starcom.tech.security` will fetch the article too. On another server, the numbers may be very different.

The `Path` specifies the list of machines through which this article has travelled before it has reached the current server. UUCP−style syntax is used for this string. The current example indicates that a user called `shuvam` first wrote this article and posted it onto a computer which calls itself `purva`, and this computer then transferred this article by a newsfeed to `news.starcomsoftware.com`. The `Path` header is critical for breaking loops in newsfeeds, and will be discussed in detail later.

Our sample article will sit in the two newsgroups listed above forever, unless expired. The Usenet software on a server is usually configured to expire articles based on certain conditions, *e.g.* after it's older than a certain number of days. The C−News software we use allows expiry control based on the newsgroup hierarchy and the type of newsgroup, *i.e.* moderated or unmoderated. Against each class of newsgroups, it allows the administrator to specify a number of days after which the article will be expired. It is possible for an article to control its own expiry, by carrying an `Expires` header specifying a date and time. Unless overriden in the Usenet server software, the article will be expired only after its explicit expiry time is reached.

2.2. Of readers and servers

Computers which access Usenet articles are broadly of two classes: the readers and the servers. A Usenet server carries a repository of articles, manages them, handles newsfeeds, and offers its repository to authorised readers to read. A Usenet reader is merely a computer with the appropriate software to allow a user to access a software, fetch articles, post new articles, and keep track of which articles it has read in each newsgroup. In terms of functionality, Usenet reading software is less interesting to a Usenet administrator than a Usenet server software. However, in terms of lines of code, the Usenet reader software can often be much larger than Usenet server software, primarily because of the complexities of modern GUI code.

Most modern computers almost exclusively access Usenet servers using the NNTP (Network News Transfer Protocol) for reading and posting. This protocol can also be used for inter–server communication, but those aspects will be discussed later. The NNTP protocol, like any other well–designed TCP–based Internet protocol, carries ASCII commands and responses terminated with `CR–LF`, and comprises a sequence of commands, somewhat reminiscent of the POP3 protocol for email. Using NNTP, a Usenet reader program connects to a Usenet server, asks for a list of active newsgroups, and receives this (often huge) list. It then sets the ``current newsgroup'' to one of these, depending on what the user wants to browse through. Having done this, it gets the meta–data of all current articles in the group, including the author, subject line, date, and size of each article, and displays an index of articles to the user.

The user then scans through this list, selects an article, and asks the reader to fetch it. The reader gives the article number of this article to the server, and fetches the full article for the user to read through. Once the user finishes his NNTP session, he exits, and the reader program closes the NNTP socket. It then (usually) updates a local file in the user's home area, keeping track of which news articles the user has read. These articles are typically not shown to the user next time, thus allowing the user to progress rapidly to new articles in each session. The reader software is helped along in this endeavour by the `Xref` header, using which it knows all the different identities by which a single article is identified in the server. Thus, if you read the sample article given above by accessing `starcom.tech.misc`, you'll never be shown this article again when you access `starcom.tech.misc` or `starcom.tech.security`; your reader software will do this by tracking the `Xref` header and mapping article numbers.

When a user posts an article, he first composes his message using the user interface of his reader software. When he finally gives the command to send the article, the reader software contacts the Usenet server using the pre–existing NNTP connection and sends the article to it. The article carries a `Newsgroups` header with the list of newsgroups to post to, often a `Distribution` header with a distribution specification, and other headers like `From`, `Subject` *etc.* These headers are used by the server software to do the right thing. Special and rare headers like `Expires` and `Approved` are acted upon when present. The server assigns a new article number to the article for each newsgroup it is posted to, and creates a new `Xref` header for the article.

Transfer of articles between servers is done in various ways, and is discussed in quite a bit of detail in Section XXX titled ``Newsfeeds'' below.

2.3. Newsfeeds

2.3.1. Fundamental concepts

When we try to analyse newsfeeds in real life, we begin to see that, for most sites, traffic flow is not symmetrical in both directions. We usually find that one server will feed the bulk of the world's articles to one or more secondary servers every day, and receive a few articles written by the users of those secondary servers in exchange. Thus, we usually find that articles flow down from the stem to the branches to the leaves of the worldwide Usenet server network, and not exactly in a totally balanced mesh flow pattern. Therefore, we use the term ``upstream server'' to refer to the server from which we receive the bulk of our daily dose of articles, and ``downstream server'' to refer to those servers which receive the bulk dose of articles from us.

Newsfeeds relay articles from one server to their ``next door neighbour'' servers, metaphorically speaking. Therefore, articles move around the globe, not by a massive number of single−hop transfers from the originating server to every other server in the world, but in a sequence of hops, like passing the baton in a relay race. This increases the latency time for an article to reach a remote tertiary server after, say, ten hops, but it allows tighter control of what gets relayed at every hop, and helps in redundancy, decentralisation of server loads, and conservation of network bandwidth. In this respect, Usenet newsfeeds are more complex than HTTP data flows, which typically use single−hop techniques.

Each Usenet news server therefore has to worry about newsfeeds each time it receives an article, either by a fresh post or from an incoming newsfeed. When the Usenet server digests this article and files it away in its repository, it simultaneously looks through its database to see which other server it should feed the article to. In order to do this, it carries out a sequence of checks, described below.

Each server knows which other servers are its ``next door neighbours;'' this information is kept in its newsfeed configuration information. Against each of its ``next door neighbours,'' there will be a list of newsgroups which it wants, and a list of distributions. The new article's list of newsgroups will be matched against the newsgroup list of the ``next door neighbour'' to see whether there's even a single common newsgroup which makes it necessary to feed the article to it. If there's a matching newsgroup, and the server's distribution list matches the article's distribution, then the article is marked for feeding to this neighbour.

When the neighbour receives the article as part of the feed, it performs some sanity checks of its own. The first check it performs is on the `Newsgroups` header of the new article. If none of the newsgroups listed there are part of the active newsgroups list of this server, then the article can be rejected. An article rejected thus may even be queued for outgoing feeds to other servers, but will not be digested for incorporation into the local article repository.

The next check performed is against the `Path` header of the incoming article. If this header lists the name of the current Usenet server anywhere, it indicates that it has already passed through this server at least once before, and is now re−appearing here erroneously because of a newsfeed loop. Such loops are quite often configured into newsfeed topologies for redundancy: ``I'll get the articles from Server X if not Server Y, and may the first one in win.'' The Usenet server software automatically detects a duplicate feed of an article and rejects it.

The next check is against what is called the server's *history database*. Every Usenet server has a history database, which is a list of the message IDs of all current articles in the local repository. Oftentimes the history database also carries the message IDs of all messages recently expired. If the incoming article's message ID matches any of the entries in the database, then again it is rejected without being filed in the local repository. This is a second loop detection method. Sometimes, the mere checking of the article's `Path` header does not detection of all potential problems, because the problem may be a re−insertion instead of a loop. A re−insertion happens when the same incoming batch of news articles is re−fed into the local server, perhaps after recovering the system's data from tapes after a system crash. In such cases, there's no newsfeed loop, but there's still the risk that one article may be digested into the local server twice. The history database prevents this.

All these simple checks are very effective, and work across server and software types, as per the Internet standards. Together, they allow robust and fail−safe Usenet article flow across the world.

2.3.2. Types of newsfeeds

This section explains the basics of newsfeeds, without getting into details of software and configuration files.

2.3.2.1. Queued feeds

This is the commonest method of sending articles from one server to another, and is followed whenever large volumes of articles are to be transferred per day. This approach needs a one−time modification to the upstream server's configuration for each outgoing feed, to define a new *queue*.

In essence all queued feeds work in the following way. When the sending server receives an article, it processes it for inclusion into its local repository, and also checks through all its outgoing feed definitions to see whether the article needs to be queued for any of the feeds. If yes, it is added to a *queue file* for each outgoing feed. The precise details of the queue file can change depending on the software implementation, but the basic processes remain the same. A queue file is a list of queued articles, but does not contain the article contents. Typical queue files are ASCII text files with one line per article giving the path to a copy of the article in the local spool area.

Later, a separate process picks up each queue file and creates one or more *batches* for each outgoing feed. A *batch* is a large file containing multiple Usenet news articles. Once the batches are created, various transport mechanisms can be used to move the files from sending server to receiving server. You can even use scripted FTP. You only need to ensure that the batch is picked up from the upstream server and somehow copied into a designated incoming batch directory in the downstream server.

UUCP has traditionally been the mechanism of choice for batch movement, because it predates the Internet and wide availability of fast packet−switched data networks. Today, with TCP/IP everywhere, UUCP once again emerges as the most logical choice of batch movement, because it too has moved with the times: it can work over TCP.

NNTP is the *de facto* mechanism of choice for moving queued newsfeeds for carrier−class Usenet servers on the Internet, and unfortunately, for a lot of other Usenet servers as well. The reason why we find this choice unfortunate is discussed in Section 12.1> below. But in NNTP feeds, an intermediate step of building batches out of queue files can be eliminated −−− this is both its strength and its weakness.

In the case of queued NNTP feeds, articles get added to queue files as described above. An NNTP transmit process periodically wakes up, picks up a queue file, and makes an NNTP connection to the downstream server. It then begins a processing loop where, for each queued article, it uses the NNTP `IHAVE` command to inform the downstream server of the article's message~ID. The downstream server checks its local repository to see whether it already has the message. If not, it responds with a `SENDME` response. The transmitting server then pumps out the article contents in plaintext form. When all articles in the queue have been thus processed, the sending server closes the connection. If the NNTP connection breaks in between due to any reason, the sending server truncates the queue file and retains only those articles which are yet to be transmitted, thus minimising repeat transmissions.

> A queued NNTP feed works with the sending server making an NNTP connection to the receiving server. This implies that the receiving server must have an IP address which is known to the sending server or can be looked up in the DNS. If the receiving server connects to the Internet periodically using a dialup connection and works with a dynamically assigned IP address, this can get tricky. UUCP feeds suffer no such problems because the sending server for the newsfeed can be the UUCP server, *i.e.* passive. The receiving server for the

feed can be the UUCP master, *i.e.* the active party. So the receiving server can then initiate the UUCP connection and connect to the sending server. Thus, if even one of the two parties has a static IP address, UUCP queued feeds can work fine.

Thus, NNTP feeds can be sent out a little faster than the batched transmission processes used for UUCP and other older methods, because no batches need to be constructed. However, NNTP is often used in newsfeeds where it is not necessary and it results in colossal waste of bandwidth. Before we study efficiency issues of NNTP versus batched feeds, we will cover another way feeds can be organised using NNTP: the pull feeds.

2.3.2.2. Pull feeds

This method of transferring a set of articles works only over NNTP, and requires absolutely no configuration on the transmitting, or upstream, server. In fact, the upstream server cannot even easily detect that the downstream server is pulling out a feed ––– it appears to be just a heavy and thorough newsreader, that's all.

This pull feed works by the downstream server pulling out articles i one by one, just like any NNTP newsreader, using the NNTP `ARTICLE` command with the Message–ID as parameter. The interesting detail is how it gets the message~IDs to begin with. For this, it uses an NNTP command, specially designed for pull feeds, called `NEWNEWS`. This command takes a hierarchy and a date,

```
NEWNEWS comp 15081997
```

This command is sent by the downstream server over NNTP to the upstream server, and in effect asks the upstream server to list out all news articles which are newer than 15 August 1997 in the `comp` hierarchy. The upstream server responds with a (often huge) list of message~IDs, one per line, ending with a period on a line by itself.

The pulling server then compares each newly received message~ID with its own article database and makes a (possibly shorter) list of all articles which it does not have, thus eliminating duplicate fetches. That done, it begins fetching articles one by one, using the NNTP `ARTICLE` command as mentioned above.

In addition, there is another NNTP command, `NEWGROUPS`, which allows the NNTP client ––– *i.e.* the downstream server in this case ––– to ask its upstream server what were the new newsgroups created since a given date. This allows the downstream server to add the new groups to its `active` file.

The `NEWNEWS` based approach is usually one of the most inefficient methods of pulling out a large Usenet feed. By inefficiency, here we refer to the CPU loads and RAM utilisation on the upstream server, not on bandwidth usage. This inefficiency is because most Usenet news servers do not keep their article databases indexed by hierarchy and date; CNews certainly does not. This means that a `NEWNEWS` command issued to an upstream server will put that server into a sequential search of its article database, to see which articles fit into the hierarchy given and are newer than the given date.

If pull feeds were to become the most common way of sending out articles, then all upstream servers would badly need an efficient way of sorting their article databases to allow each `NEWNEWS` command to rapidly generate its list of matching articles. A slow upstream server today might take minutes to begin responding to a `NEWNEWS` command, and the downstream server may time out and close its NNTP connection in the meanwhile. We have often seen this happening, till we tweak timeouts.

There are basic efficiency issues of bandwidth utilisation involved in NNTP for news feeds, which are applicable for both queued and pull feeds. But the problem with `NEWNEWS` is unique to pull feeds, and relates to server loads, not bandwidth wastage.

2.4. Control messages

The Usenet is a massive dispersed collection of servers which operate almost without any supervision, provided they have adequate disk space and do not suffer disk corruption due to power failures, *etc.* (It is indeed surprising how self−managing a good Usenet server is, provided these two pre−requisites are met.) These servers are each under the control of human administrators, but it is preferable that certain routine actions be performed across all these servers remotely from one location, without the manual intervention of these humans.

One common need for centralised operations is the creation of new groups in the standard eight hierarchies. The Usenet follows a fairly formal process which asks for votes from readers worldwide before deciding on the restructuring of its newsgroups list, including merging of low−volume groups, splitting of high−volume groups into many specialised groups, creating new groups, and even deleting groups. Once the voting process for a change concludes and the change action is to be carried out, it would be extremely tedious to send email to the hundreds of thousands of Usenet administrators and hope that they make the changes right, and answer their doubts if they get confused. It would be much better to have an *automatic* way to make the changes across all servers, of course with proper authorisation.

The solution to this does not lie in giving some central authority the ability to run an OS−level command of his choice on all the world's Usenet servers, because OS commands differ from OS to OS, and because few Usenet administrators would trust a stranger from another part of the world with OS level access. Therefore, the solution lay in defining a small set of common Usenet maintenance actions, and permitting only these actions to be triggered on all servers through the passing of special command messages, called *control messages*.

Control messages look like ordinary Usenet articles, more or less. They have an extra header line, with its value in a specific format, but they usually carry body text which looks like a normal human−written article. Here is a control message (a spurious one at that, but it'll do for now):

```
Xref: news.starcomsoftware.com control:814217
Path: news.starcomsoftware.com!linux594.dn.net!news.dn.hoopoo.com!
        feed-out.newsfeeds.com!newsfeeds.com!feed.newsfeeds.com!
        newsfeeds.com!news-spur1.maxwell.syr.edu!news.maxwell.syr.edu!
        newsfeed.icl.net!newsfeed.skycache.com!Cidera!newsfeed.gamma.ru!
        Gamma.RU!carrier.kiev.ua!goblin.nadrabank.kiev.ua!not-for-mail
From: tale@uunet.uu.net (David C Lawrence)
Newsgroups: news.groups,humanities.hipcrime
Subject: cmsg newgroup humanities.hipcrime
Control: newgroup humanities.hipcrime
Date: Sun, 18 Feb 2001 11:50:28 GMT
Organization: The Cabal
Lines: 20
Approved: tale@uunet.uu.net
Message-ID: <3afWYZTIR.G5YOC2@uunet.uu.net>
NNTP-Posting-Host: 203.145.147.67
X-Trace: goblin.nadrabank.kiev.ua 982528840 21455 203.145.147.67
         (18 Feb 2001 20:40:40 GMT)
X-Complaints-To: usenet@nadrabank.kiev.ua
NNTP-Posting-Date: 18 Feb 2001 20:40:40 GMT
X-No-Archive: Yes

humanities.hipcrime is an unmoderated newsgroup which passed its
vote for creation by 326:10 as reported in news.announce.newgroups
```

```
on 18 Feb 2001.

For your newsgroups file:
humanities.hipcrime     HipCrime for Humanity - you committed one now!

Anyone can create a newsgroup in the alt, biz, comp, earth,
humanities, misc, news, meow, rec, sci, soc, talk, us, or
any other Usenet hierarchy.  New newsgroup proposals may be
optionally discussed in news.groups. Please be sure that your
/usr/lib/news/control.ctl is configured correctly:

##      NEWGROUP MESSAGES
## honor them all and log in \${LOG}/newgroup.log
newgroup:*:alt.*|biz.*|comp.*|earth.*|humanities.*|misc.*|news.*|\
        meaw.*|rec.*|sci.*|soc.*|talk.*|us.*:doit=newgroup

##      RMGROUP MESSAGES
## drop them all and don't log
rmgroup:*:*:drop

Meow!
David C Lawrence
```

A control message must have a `Control` header. Besides, all control messages *will* have an `Approved` header, like messages posted to moderated newsgroups. The `Control` header actually specifies a command to run on the local server, and the parameter(s) to supply to it. The local Usenet server software is supposed to figure out its own way to get the task done. In this example, the command in the `Control` header is `newgroup`, which creates a new newsgroup. And its parameter is `humanities.hipcrime`, which gives the name of the newsgroup to create.

In C–News, the control message implementation works through separate shellscripts kept in a fixed directory, `$NEWSBIN/ctl/`, as a security measure; if the executable script isn't present there, the control message command will be ignored. The control message types supported are:

- `checkgroups`: control message to check whether the list of newsgroups in your active file are all correct as per a master list of newsgroups sent in the control message
- `newgroup`: control message to create a new newsgroup
- `rmgroup`: control message to delete a newsgroup and all articles in it
- `sendsys`: control message to cause an email response to be sent to the author with the `sys` file of your server in it. This results in a response storm of emails from all the Usenet servers in the world to the author. These responses allow the sender of the control message to analyse all the `sys` files of the world's Usenet servers and create the directed graph of Usenet newsfeeds. Why someone would want to do this is hard to guess, but the result is surely an awesome picture of one facet of networked human civilisation, like looking at a giant world map.

  Incidentally, there is no invasion of privacy here, because your server's `sys` file is supposed to be public information, if you take feeds from the public Usenet.
- `version`: control message which results in your Usenet software sending an email to the author of the message, containing the type and version of the Usenet news software you are using. This too is not an invasion of privacy, because this information is supposed to be public knowledge.
- The cancel message: the most frequently occurring type of control messages. They specify the message ID of an article, and result in the cancellation (deletion) of that article. If you post an article and regret it a moment later, your Usenet newsreader software usually allows you to ``cancel'' it by generating a cancel message.

The Usenet news software maintains a pseudo–newsgroup called `control`, where it files all control messages it receives. If you have an incoming newsfeed from the public Usenet, your server's `control` group will usually be full with thousands of cancel messages from trigger–happy fingers all over the world. Usenet news server software like C–News allows you to filter the incoming feed based on newsgroups, and will discard articles for groups they do not subscribe to. But since all servers have to receive and process control messages, they will all accept these cancel messages, though many of them may apply to articles which are not part of your highly–pruned subset of groups. `C'est la vie`.

Remember to set expiry for the `control` group to one day or even shorter, so that the junk can be cleaned out as rapidly as possible, just like the `junk` newsgroup.

The beauty of the control message architecture is that it integrates seamlessly into the newsfeed mechanism for automatic control of the network of servers. No separate channel of connection is needed for the control actions. And article replication automatically propagates control messages with human–readable articles, thus guaranteeing reach across heterogenous networks technologies.

What your Usenet server does on receiving a control message is governed by an authorisation file: `$NEWSCTL/controlperms` in the case of C–News and `control.ctl` in the case of INN, for instance. The security measures implemented by this module are further enhanced by the `pgpcontrol` package with its `pgpverify` script. Using `pgpverify`, your server can check that all control messages (except for article cancellation messages) are digitally signed by a trusted party using military–spec public key cryptography. Our integrated Usenet news software distribution includes integration with `pgpverify`.

3. Usenet news software

3.1. A brief history of Usenet systems

Towards the end of this HOWTO, we have added some information about the history of Usenet server software by quoting sections from an earlier Usenet Periodic Posting. We consider this historical perspective, and the Usenix papers and other documents referred to in it, essential reading for any Usenet server administrator. Please see the section titled "Usenet software: a historical perspective>".

3.2. C−News and NNTPd

C−News was written by Henry Spencer and Geoff Collyer of the Department of Zoology, University of Toronto, almost entirely in shell and `awk`, as a replacement for an earlier system called B−News. The focus was on adding some extra features and a lot of performance. The first release was called Shellscript Release, which was deployed by a very large number of servers worldwide, as a natural upgrade to B−News. This version of C−News had upward compatibility with B−News meta−data, *e.g.* history files. This was the version of C−News which was initially rolled out in 1991 or so at the National Centre for Software Technology (NCST, `http://www.ncst.ernet.in`) and the Indian Institutes of Technology in India as part of the Indian educational and research network (ERNET). We received guidance from the NCST about Usenet news installation and management.

The Shellscript Release was soon followed by a re−write with a lot more C code, called Performance Release, and then a set of cleanup and component integration steps leading to the last release called the Cleanup Release. This Cleanup Release was patched many times by the authors, and the last one was CR.G (Cleanup Release revision G). The version of C−News discussed in this HOWTO is a set of small bug fixes on CR.G.

Since C−News came from shellscript−based antecedents, its architecture followed the set−of−programs style so typical of Unix, rather than large monolothic software systems traditional to some other OSs. All pieces had well−defined roles, and therefore could be easily replaced with other pieces as needed. This allowed easy adaptations and upgradations. This never affected performance, because key components which did a lot of work at high speed, *e.g.* `newsrun`, had been rewritten in C by that time. Even within the shellscripts, crucial components which handled binary data, *e.g.* a component called `dbz` to manipulate efficient on−disk hash arrays, were C programs with command−line interfaces, called from scripts.

C−News was born in a world with widely varying network line speeds, where bandwidth utilisation was a big issue and dialup links with UUCP file transfers was common. Therefore, it has strong support for batched feeds, specially with a variety of compression techniques and over a variety of fast and slow transport channels. And C−News virtually does not know the existence of TCP/IP, other than one or two tiny batch transport programs like `viarsh`. However, its design was so modular that there was absolutely no problem in plugging in NNTP functionality using a separate set of C programs without modifying a single line of C−News. This was done by a program suite called NNTP Reference Implementation, which we call NNTPd.

This software suite could work with B−News and C−News article repositories, and provided the full NNTP functionality. Since B−News died a gradual death, the combination of C−News and NNTPd became a freely redistributable, portable, modern, extensible, and high−performance software suite for Unix Usenet servers. Further refinements were added later, *e.g.* `nov`, the News Overview package and `pgpverify`, a public−key−based digital signature module to protect Usenet news servers against fraudulent control messages.

3.3. INN

INN is one of the two most widely used Usenet news server solutions. It was written by Rich Salz for Unix systems which have a socket API −−− probably all Unix systems do, today.

INN has an architecture diametrically opposite to CNews. It is a monolithic program, which is started at bootup time, and keeps running till your server OS is shut down. This is like the way high performance HTTP servers are run in most cases, and allows INN to cache a lot of things in its memory, including message−IDs of recently posted messages, *etc.* This interesting architecture has been discussed in an interesting paper by the author, where he explains the problems of the older B−News and C−News systems that he tried to address. Anyone interested in Usenet software in general and INN in particular should study this paper.

INN addresses a Usenet news world which revolves around NNTP, though it has support for UUCP batches −−− a fact that not many INN administrators seem to talk about. INN works faster than the CNews−NNTPd combination when processing multiple parallel incoming NNTP feeds. For multiple readers reading and posting news over NNTP, there is no difference between the efficiency of INN and NNTPd. Section 5.7> discusses the efficiency issues of INN over the earlier C−News architecture, based on Rich Salz' paper and our analyses of usage patterns.

INN's architecture has inspired a lot of high−performance Usenet news software, including a lot of commercial systems which address the ``carrier class'' market. That is the market for which the INN architecture has clear advantages over C−News.

3.4. Leafnode

This is an interesting software system, to set up a ``small'' Usenet news server on one computer which only receives newsfeeds but does not have the headache of sending out bulk feeds to other sites, *i.e.* it is a ``leaf node'' in the newsfeed flow diagram. According to its homepage (`www.leafnode.org`), ``Leafnode is a USENET software package designed for small sites running any flavour of Unix, with a few tens of readers and only a slow link to the net. [...] The current version is 1.9.24.''

This software is a sort of combination of article repository and NNTP news server, and receives articles, digests and stores them on the local hard disks, expires them periodically, and serves them to an NNTP reader. It is claimed that it is simple to manage and is ideal for installation on a desktop−class Unix or Linux box, since it does not take up much resources.

Leafnode is based on an appealing idea, but we find no problem using C−News and NNTPd on a desktop−class box. Its resource consumption is somewhat proportional to the volume of articles you want it to process, and the number of groups you'll want to retain for a small team of users will be easily handled by C−News on a desktop−class computer. An office of a hundred users can easily use C−News and NNTPd on a desktop computer running Linux, with 64 MBytes of RAM, IDE drives, and sufficient disk space. Of course, ease of configuration and management is dependent on familiarity, and we are more familiar with C−News than with Leafnode. We hope this HOWTO will help you in that direction.

There *is*, however, one area in which Leafnode is far easier to administer than INN or C−News. Leafnode constantly monitors the actual usage of the newsgroups it carries, based on readership statistics of its NNTP readers. If a particular newsgroup is not read at all by any user for a week, then Leafnode will delete all articles in that newsgroup, free up disk space, and stop fetching new articles for it. If it finds that a previously abandoned newsgroup is now again receiving attention, even from one user, then it'll fetch all articles for that group from its upstream server the next time it connects. This self−tuning feature of Leafnode is really an excellent advantage which makes a Leafnode site easier to manage, specially for small setups with bandwidth and disk space constraints.

The Leafnode Website gives a lot of details in an easily understood format.

TO BE EXTENDED AND CORRECTED.

3.5. Suck

Suck is a program which lets you pull out an NNTP feed from an NNTP server and file it locally. It does not contain any article repository management software, expecting you to do it using some other software system, *e.g.* C−News or INN. It can create batchfiles which can be fed to C−News, for instance. (Well, to be fair, Suck *does* have an option to store the fetched articles in a spool directory tree very much like what is used by C−News or INN in their article area, with one file per article. You can later read this raw message spool area using a mail client which supports the `msgdir` file layout for mail folders, like MH, perhaps. We don't find this option useful if you're running Suck on a Usenet server.) Suck finally boils down to a single command−line program which is invoked periodically, typically from `cron`. It has a zillion command−line options which are confusing at first, but later show how mature and finely tunable the software is.

If you need an NNTP pull feed, then we know of no better programs than Suck for the job. The `nntpxfer` program which forms part of the NNTPd package also implements an NNTP pull feed, for instance, but does not have one−tenth of the flexibility and fine−tuning of Suck. One of the banes of the NNTP pull feed is connection timeouts; Suck allows a lot of special tuning to handle this problem. If we had to set up a Usenet server with an NNTP pull feed, we'd use Suck right away.

TO BE EXTENDED AND CORRECTED.

3.6. Carrier class software

Carrier−class servers are expected to handle a complete feed of all articles in all newsgroups, including a lot of groups which have what we call a ``high noise−to−signal ratio.'' They do not have the luxury of choosing a ``useful'' subset like administrators of internal corporate Usenet servers do. Secondly, carrier−class servers are expected to turn articles around very fast, *i.e.* they are expected to have very low latency from the moment they receive an article to the time they retransmit it by NNTP to downstream servers. Third, they are supposed to provide very high availability, like other ``carrier class'' services. This usually means that they have parallel arrays of computers in load sharing configurations. And fourth, they usually do not cater to retail connections for reading and posting articles by human users. Usenet news carriers usually reserve separate computers to handle retail connections.

Thus, carrier−class servers do not need to maintain a repository of articles; they only need to focus on super−efficient real−time re−transmission. These highly specialised servers have software which receive an article over NNTP, parse it, and immediately re−queue it for outward transmission to dozens or hundreds of other servers. And since they work at these high throughputs, their downstream servers are also expected to be live on the Internet round the clock to receive incoming NNTP connections, or be prepared to lose articles. Therefore, there's no batching or long queueing needed, and C−News−style batching in fact is totally inapplicable.

Therefore, these carrier−class Usenet servers are more like packet routers than servers with repositories. They are referred to nowadays as NNTP routers or news routers.

It can be seen why batch−oriented repository management software like C−News is a total anachronism here, and why they need an NNTP−oriented, online, real−time design. The INN antecedents of some of these systems is therefore natural. We would love to hear from any Linux HOWTO reader whose Usenet server requirements include carrier−class behaviour.

We are aware of only one freely redistributable NNTP router: NNTPRelay (see `http://nntprelay.maxwell.syr.edu/`); this software runs on NT. There is no reason why such services cannot run off Linux servers, even Intel Linux, provided you have fast network links and arrays of servers. Linux as an OS platform is not an issue here.

TO BE EXTENDED AND CORRECTED.

4. Setting up CNews + NNTPd

4.1. Getting the sources and stuff

4.1.1. The sources

C–News software can be obtained from `ftp://ftp.uu.net/networking/news/transport/cnews/cnews.tar.Z` and will need to be uncompressed using the BSD `uncompress` utility or a compatible program. The tarball is about 650 KBytes in size. It has its own highly intelligent configuration and installation processes, which are very well documented. The version that is available is Cleanup Release revision G, on which our own version is based.

NNTPd (the NNTP Reference Implementation) is available from `ftp://ftp.uu.net/networking/news/nntp/nntp.1.5.12.1.tar.Z`. It has no automatic scripts and processes to configure itself. After fetching the sources, you will have to follow a set of directions given in the documentation and configure some C header files. These configuration settings must be done keeping in mind what you have specified when you build the C–News sources, because NNTPd and C–News must work together. Therefore, some key file formats, directory paths, *etc.*, will have to be specified identically in both software systems.

The third software system we use is Nestor. This too is to be found in the same place where the NNTPd software is kept, at `ftp://ftp.uu.net/networking/news/nntp/nestor.tar.Z`. This software compiles to one binary program, which must be run periodically to process the logs of `nntpd`, the NNTP server which is part of NNTPd, and report usage statistics to the administrator. We have integrated Nestor into our source base.

The fourth piece of the system, without which no Usenet server administrator dares venture out into the wild world of public Internet newsfeeds, is `pgpverify`.

We have been working with C–News and NNTPd for many years now, and have fixed a few bugs in both packages. We have also integrated the four software systems listed above, and added a few features here and there to make things work more smoothly. We offer our entire source base to anyone for free download from `http://www.starcomsoftware.com/proj/usenet/src/news.tar.gz`. There are no licensing restrictions on our sources; they are as freely redistributable as the original components we started with.

When you download our software distribution, you will extract it to find a directory tree with the following subdirectories and files:

- `c-news`: the source tree of the CR.G software release, with our additions like `pgpverify` integration, our scripts like `mail2news`, and pre–created configuration files.
- `nntp-1.5.12.1`: the source tree of the original NNTPd release, with header files pre–configured to fit in with our configuration of C–News, and our addition of bits and pieces like Nestor, the log analysis program.
- `howto`: this document, and its SGML sources and Makefile.
- `build.sh`: a shellscript you can run to compile the entire combined source tree and install binaries in the right places, if you are lucky and all goes well.

Needless to say, we believe that our source tree is a better place to start with than the original components, specially if you are installing a Usenet server on a Linux box and for the first time. We will be available on email to provide technical assistance should you run into trouble.

4.1.2. The key configuration files

Once you get the sources, you will need some key configuration files to seed your C−News system. These configuration files are actually database tables, and are changing frequently, whenever newsgroups are created, modified or deleted. These files specify the list of active newsgroups in the ``public'' Usenet. You can, and should, add your organisation's internal newsgroups to this list when you set up your own server, but you will need to know the list of public standard newsgroups to begin with. This list can be obtained from the same FTP server by downloading the files `active.gz` and `newsgroups.gz` from `ftp://ftp.uu.net/networking/news/config/`. You can create your own `active` and `newsgroups` files by retaining a subset of the entries in these two files. Both these are ASCII text files.

Getting the sources from our server will not obviate the need to get the latest versions of these files from `ftp.uu.net`. We do not (yet) maintain an up−to−date copy of these files on our server, and we will add no value to the original by just mirroring them.

4.2. Compiling and installing

For installing, first make sure you have an entry for a user called `news` in your `/etc/password` file. This is setting the news−database owner to `news`. Now download the source from us and untar it in the home directory of news. This creates two main directories *viz.* `c−news` and `nntp`. To install and compile, run the script `build.sh` as root in the directory that contains the script. It is important that the script run as `root` as it sets ownerships, installs and compiles the source as user `news`. This is a one−step process that puts in place both the C−News and the NNTP software, setting correct permissions and paths. Following is a brief description of what build.sh does:

- Checks for the `OS` platform and exits if it is not `Linux`.
- Again, exits if you are not running as `root`.
- Looks for and exits if cannot find the above two directories.
- Compiles `C−News` and performs regression tests if the compilation was successfull. Sends out a warning to read the error file `make.out.r` and to fix 'em. Compilation erros are written to a file called `make.out`.
- Performs the above operation in the `nntp` directory, too.
- Checks for the presence of the three key directories: `$NEWSARTS − (/var/spool/news)` that houses the artciles, `$NEWSCTL −(/var/lib/news)` that contain configuration, log and status files and `$NEWSBIN − (/usr/lib/newsbin)` that contain binaries and executables for the working of the Usenet News system. Tries to create them if non−existent and exits if it results in failure.
- Changes the ownership of these directories to `news.news`. This is important since the entire Usenet News System runs as user `news.` It will not function properly as any other user.
- Then starts the installation process of C News. It runs `make install` to install binaries at the right locations; `make setup` to set the correct paths and umask, create directories for newsgroups, determine who will receive reports; `make ui` to set up inews and injnews and `make readpostcheck` to use readnews, postnews and checknews scripts provided by C News. The errors, if any are to be found in the respective `make.out` files. e.g. make.setup will write errors to `make.out.setup`
- `Newsspool`, which queues incoming batches in `$NEWSARTS/in.coming` directory should run as set−userid and set−groupid. This is done.
- A softlink is made to `/var/lib/news` from `/usr/lib/news.`

* The NNTP software is installed.
* Sets up the manpages for C News and makes it world readable. The NNTP manpages get installed when the software is installed. Compiles the C News documentation `guide.ps` and makes it readable and available in `/usr/doc/packages/news` or `/usr/doc/news`.
* Checks for the PGP binary and asks the administrator to get it, if not found.

4.3. Configuring the system: What and how to configure files?

Once installed, you have to now configure the system to accept feeds and batch them for your neighbours. You will have to do the following:

* `nntpd`: Copy the compiled `nntpd` into a directory where executables are kept and activate it. It runs on port 119 as a daemon through `inetd` unless you have compiled it as stand−alone. An entry in the `/etc/services` file for nntp would look like this:

  ```
  nntp    119/tcp    \# Network News Transfer Protocol
  ```

  An entry in the `inetd.conf` file will be:

  ```
   nntp    stream    tcp    nowait   news    path−to−tcpd  path−to−nntpd
  ```

  The last two fields in the `inetd.conf` file are paths to binaries of the `tcp` and the `nntp` daemon respectively.
* *Configuring control files:* There are plenty of control files in `$NEWSCTL` that will need to be configured before you can start using the news system. The files mentioned here are also discussed in the first section of the section titled "Components of a running system>". These control files are dealt in detail in the following below.

  ♦ `sys`: One line per system/NDN listing all the newsgroup hierarchies each system subscribes to. Each line is prefixed with the system name and the one beginning with

  ```
  ME:
  ```

  indicates what your server is willing to receive. Following are typical entries that go into this file:

  ```
  ME:comp,news,misc,netscape
  ```

  This line indicates what newsgroups your server has subscribed to.

  ```
  server/server.starcomsoftware.com:all,!general/all:f
  ```

  This is a list of newsgroups your server will pass on to your NDN. The newsgroups specified should be a comma separated list and the entire line should contain no spaces. The *f* flag indicates that the newsgroup name and the article number alongwith its size will make up one entry in the `togo` file in the `$NEWSARTS/out.going` directory.

  ♦ `explist`: This file has entries indicating which articles expire and when and whether they have to be archived. The order in which the newsgroups are listed is important. An example follows:

  ```
  comp.lang.java.3d    x    60    /var/spool/news/Archive
  ```

  This means that the articles of comp.lang.java expire after 60 days and shall be archived in the directory mentioned in the fourth field. Archiving is an option. The second field indicates that this line applies to both moderated and unmoderted newsgroups. *m* would specify moderated and *u* would specify unmoderated groups. If you want to specify an extremely large no. as the expiry period you can use the keyword "never".

  ♦ `batchparms`: `sendbatches` is a program that administers batched transmission of news articles to other sites. To do this it consults the `batchparms` file. Each line in the file specifies the behaviour for each of your NDN mentioned in the `sys` file. There are five fields for each site to be specified.

```
server   u     100000    100    batcher | gzip -9 | viauux -d gunzip
```

The first field is the site name which matches the entry in the `sys` file and has a corresponding directory in `$NEWSARTS/out.going` by that name.

The second field is the class of the site,*u* for UUCP and *n* for NNTP feeds. A "!" in this field means that batching for this site has been disabled.

The third field is the size of batches to be prepared in bytes.

The fourth field is the maximum length of the output queue for transmission to that site.

The fifth field is the command line to be used to build, compress and transmit batches to that site. The contents of the `togo` file are made available on standard input.

♦ `controlperm`: This file controls how the news system responds to control messages. Each line consists of 4−5 fields separated by white space. Control messages has been discussed in "Section 2.4>".

```
comp,sci    tale@uunet.uu.net   nrc    pv   news.announce.newsgroups
```

The first field is a newsgroup pattern to which the line applies.

The second field is either the keyword "any" or an e−mail address. The latter specifies that the line applies to control messages from only that author.

The third field is a set of opcode letters indicating what control operations need to be performed on messages emanating from the e−mail address mentioned in the second field. *n* stands for creating a newgroup, *r* stands for deleting a newsgroup and *c* stands for checkgroup.

The fourth field is a set of flag letters indicating how to respond to a control message that meets all the applicability tests:

```
            y  Do it.
            n  Don't do it.
            v  Report it and include the entire control
               message in the report.
            q  Don't report it.
            p  Do it iff the control message carries a valid PGP signature.
```

Exactly one of y, n or p must be present.

The fifth field, which is optional, will be used if the fourth field contains a *p*. It must contain the PGP key ID of the public key to be used for signature verification.

♦ `mailpaths`: This file describes how to reach the moderators of various hierarchies of newsgroups by mail. Each line consists of two fields: a news group pattern and an e−mail address. The first line whose group pattern matches the newsgroup is used. As an example:

```
          comp.lang.java.3d            somebody@mydomain.com
          all                          %s@moderators.uu.net
```

In the second example, the `%s` gets replaced with the groupname and all dots appearing in the newsgroup name are substituted with dashes.

♦ *Miscellaneous files:* The other files to be modified are:

- ◊ `mailname:` Contains the Internet domain name of the news system. Consider getting one if you don't have it.
- ◊ `organization:` Contains the default value for the `Organization:` header for postings originating locally.
- ◊ `whoami:` Contains the name of the news system. This is the site name used in the `Path:` headers and hence should concur with the names your neighbours use in their `sys` files.

- ♦ `active` file: This file specifies one line for each newsgroup (not just the hierarchy) to be found on your news system. You will have to get the most recent copy of the active file from `ftp://ftp.isc.org/usenet/CONFIG/active` and prune it to delete newsgroups that you have not subscribed to. Run the script `addgroup` for each newsgroup in this file which will create relevant directories in the `$NEWSARTS` area. The `addgroup` script takes two paramters: the newsgroup name being created and a flag. The flag can be any one of the following:

```
            y           local postings are allowed
            n           no local postings, only remote ones
            m           postings to this group must be approved
                        by the moderator
            j           articles in this group are only passed and not kept
            x           posting to this newsgroup is disallowed
            =foo.bar    articles are locally filed in
                        "foo.bar" group
```

  An entry in this file looks like this:

```
comp.lang.java.3d       0000003716      01346   m
```

  The first field is the name of the newsgroup. The second field is the highest article number that has been used in that newsgroup. The third field is the lowest article number in the group. The fourth field is a flag as explained above.
- ♦ `newsgroups` file: This contains a one–line description of each newsgroup to be found in the active file. You will have to get the most recent file from `ftp://ftp.isc.org/usenet/CONFIG/newsgroups` and prune it to remove unwanted information. As an example:

```
comp.lang.java.3d       3D Graphics APIs for the Java language
```

- ♦ *Aliases:* These aliases are required for trouble reporting. Once the system is in place and scripts are run, anomalies/problems are reported to addresses in the `/etc/aliases` file. These entries include email addresses for `newsmaster, newscrisis, news, usenet, newsmap`. They should ideally point to an email address that will be accessed at regularly. Arrange the emails for `newsmap` to be discarded to minimize the effect of `sendsys bombing` by practical jokers.
- ♦ *Cron jobs:* Certain scripts like `newsrun` that picks up incoming batches and maintenance scripts, should run through news–database owner's cron which is `news`. The cron entries ideally will be for the following: A more detailed report can be found in "Section 9.4>"

  1. `newsrun:` This script processes incoming batches of article. Run this as frequently as you want them to get digested.
  2. `sendbatches:` This script transmit batches to the NDNs. Set the frequency according to your requirements.
  3. `newsdaily:` This should be run ideally once a day since it reports errors and anomalies in the news system.
  4. `newswatch:` This looks for errors/anomalies at a more detailed level and hence should be run atleast once every hour

5. `doexpire:` This script expires old articles as determined by the explist file. Run this once a day.

- `newslog:` Make an entry in the system's `syslog.conf` file for logging messages spewed out by `nntpd` in `newslog` . It should be located in `$NEWSCTL`. The entry will look like this:

```
news.debug              -/var/lib/news/newslog
```

- `Newsboot:` Have this run (as `news` the news–database owner) when the system boots to clear out debris left around by crashes.
- Add a Usenet mailer in sendmail: The `mail2news` program provided as part of the source code is a handy tool to send an e–mail to a newsgroup which gets digested as an article. You will have to add the following ruleset and mailer definition in your `sendmail.cf` file:

  - Under SParse1, add the following:

```
            R$+ . USENET < @ $=w . >      $#usenet     $: $1
```

  - Under mailer definitions, define the mailer Usenet as:

```
            MUsenet      P=/usr/lib/newsbin/mail2news/m2nmailer, F=lsDFMmn,
                S=10, R=0, M=2000000, T=X-Usenet/X-Usenet/X-Unix, A=m2nmailer $u
```

  In order to send a mail to a newsgroup you will now have to suffix the newsgroup name with usenet *i.e.* your `To:` header will look like this:

```
To: misc.test.usenet@yourdomain.
```

  The mailer definition of usenet will intercept this mail and post it to the respective newsgroup, in this case, `misc.test`

This, more or less, completes the configuration part.

4.4. Testing the system

To locally test the system, follow the steps given below:

- post an article: Create a local newsgroup

```
    cnewsdo addgroup mysite.test y
```

  and using `postnews` post an article to it.
- Has it arrived in `$NEWSARTS/in.coming`?: The article should show up in the directory mentioned. Note the nomenclature of the article.
- When newsrun runs: When newsrun runs from `cron` , the article disappears from `in.coming` directory and appears in `$NEWSARTS/mysite/test`. Look how the `newsgroup, active, log and history` (not the errorlog) files and `.overview` file in `$NEWSARTS/mysite/test` reflect the digestion of the file into the news system.
- reading the article: Try to read the article through `readnews` or any news client. If you are able to, then you have set most everything right.

4.5. `pgpverify` and `controlperms`

As mentioned in "Section 2.4>", it becomes necessary to authenticate control messages to protect yourself from being attacked by pranksters. For this, you will have to configure the `$NEWSCTL/controlperm` file to declare whose control messages you are willing to honour and for what newsgroups alongwith their public key ID. The `controlperm` manpage shall give you details on the format.

This will work only in association with `pgpverify` which verifies the Usenet control messages that have been signed using the `signcontrol` process. The script can be found at `ftp://ftp.isc.org/pub/pgpcontrol/pgpverify`. `pgpverify` internally uses the PGP binary which will have to be made available in the default executables directory. If you wish to send control messages for your local news system, you will have to digitally sign them using the above mentioned `signcontrol` program which is available at `ftp://ftp.isc.org/pub/pgpcontrol/signcontrol`. You will also have to configure the `signcontrol` program accordingly.

4.6. Feeding off an upstream neighbour

For external feeds, commercial customers will have to buy them from a regular News Provider like `dejanews.com` or `newsfeeds.com`. You will have to specify to them what hierarchies you want and decide on the mode of transmission, *i.e.* UUCP or NNTP, based on your requirements. Once that is done, you will have to ask them to initiate feeds, and check `$NEWSARTS/in.coming` directory to see if feeds are coming in.

If your organisation belongs to the academic community or is otherwise lucky enough to have an NDN server somewhere which is willing to provide you a free newsfeed, then the payment issue goes out of the picture, but the rest of the technical requirements remain the same.

One problem with incoming NNTP feeds is that it is far easier to use (relatively) efficient NNTP inflows if you have a server with a permanent Internet connection and a fixed IP address. If you are a small office with a dialup Internet connection, this may not be possible. In that case, the only way to get incoming newsfeeds by NNTP may be by using a highly inefficient pull feed.

4.7. Configuring outgoing feeds

If you are a leaf node, you will only have to send feeds back to your news provider for your postings in public newsgroups to propagate to the outside world. To enable this, you need one line in the `sys` and `batchparms` files and one directory in `$NEWSARTS/out.going`. If you are willing to transmit articles to your neighbouring sites, you will have to configure `sys` and `batchparms` with more entries. The number of directories in `$NEWSARTS/out.going` shall increase, too. Refer to first two sections of the chapter titled "Components of a running system>"for a better understanding of outgoing feeds. Again, you will have to determine how you wish to transmit the feed: UUCP or NNTP.

4.7.1. By UUCP

For outgoing feeds by UUCP, we recommend that you start with Taylor UUCP. In fact, this is the UUCP version which forms part of the GNU Project and is the default UUCP on Linux systems.

A full treatment of UUCP configuration is beyond the scope of this document. However, the basic steps will be as follows. First, you will have to define a "system" in your Usenet server for the NDN (next door neighbour) host. This definition will include various parameters, including the manner in which your server will call the remote server, the protocol it will use, *etc.* Then an identical process will have to be followed on the NDN server's UUCP configuration, for your server, so that *that* server can recognize *your* Usenet server.

Finally, you will need to set up appropriate `cron` jobs for the user `uucp` to run `uucico` periodically. Taylor UUCP comes with a script called `uusched` which may be modified to your requirements; this script calls `uucico`. One `uucico` connection will both upload and download news batches. Smaller sites can run `uusched` even once or twice a day.

Later versions of this document will include the `uusched` scripts that we use in Starcom. We use UUCP over TCP/IP, and we run the `uucico` connection through an SSH tunnel, to prevent transmission of UUCP passwords in plain text over the Internet, and our SSH tunnel is established using public−key cryptography, without passwords being used anywhere.

4.7.2. By NNTP

For NNTP feeds, you will have to decide whether your server will be the connection initiator or connection recipient. If you are the connection initiator, you can send outgoing NNTP feeds more easily. If you are the connection recipient, then outgoing feeds will have to be pulled out of your server using the NNTP `NEWNEWS` command, which will place heavy loads on your server. This is not recommended.

Connecting to your NDN server for pushing out outgoing feeds will require the use of the `nntpsend.sh` script, which is part of the NNTPd source tree. This script will perform some housekeeping, and internally call the `nntpxmit` binary to actually send the queued set of articles out. You may have to provide authentication information like usernames and passwords to `nntpxmit` to allow it to connect to your NDN server, in case that server insists on checking the identity of incoming connections. (You can't be too careful in today's world.) `nntpsend.sh` will clean up after an `nntpxmit` connection finishes, and will requeue any unsent articles for the next session. Thus, even if there is a network problem, typically nothing is lost and all pending articles are transmitted next time.

Thus, pushing feeds out *via* may mean setting up `nntpsend.sh` properly, and then invoking it periodically from `cron`. If your Usenet server connects to the Internet only intermittently, then the process which sets up the Internet connection should be extended or modified to fire `nntpsend.sh` whenever the Internet link is established. For instance, if you are using the Linux `pppd`, you can add statements to the `/etc/ppp/ip−up` script to change user to `news` and run `nntpsend.sh`

5. Setting up INN

5.1. Getting the source

INN is maintained and archived by the ISC (Internet Software Consortium, `www.isc.org`) since 1996, and the INN homepage is at `http://www.isc.org/products/INN/`. The latest release of INN as of the time of this writing is INN v2.3.3, released 7 May 2002. The full sources can be downloaded from `ftp://ftp.isc.org/isc/inn/inn-2.3.3.tar.gz`

5.2. Compiling and installing

TO BE EXTENDED LATER.

5.3. Configuring the system

TO BE ADDED LATER.

5.4. Setting up `pgpverify`

TO BE ADDED LATER.

5.5. Feeding off an upstream neighbour

TO BE ADDED LATER.

5.6. Setting up outgoing feeds

TO BE ADDED LATER.

5.7. Efficiency issues and advantages

TO BE ADDED LATER.

6. Connecting email with Usenet news

Usenet news and mailing lists constantly remind us of each other. And the parallels are so strong that many mailing lists are gatewayed two−way with corresponding Usenet newsgroups, in the `bit` hierarchy which maps onto the old BITNET, and elsewhere.

There are probably ten different situations where a mailing list is better, and ten others where the newsgroup approach works better. The point to recognise is that the system administrator needs a choice of gatewaying one with the other, whenever tradeoffs justify it. Instead of getting into the tradeoffs themselves, this chapter will then focus on the mechanisms of gatewaying the two worlds.

One clear and recurring use we find for this gatewaying is for mailing lists which are of general use to many employees in a corporate network. For instance, in stockbroking company, many employees may like to subscribe to a business news mailing list. If each employee had to subscribe to the mailing list independently, it would waste mail spool area and perhaps bandwidth. In such situations, we receive the mailing list into an internal newsgroup, so that individual mailboxes are not overloaded. Everyone can then read the newsgroup, and messages are also archived till expired.

6.1. Feeding Usenet news to email

In CNews, this is trivially done by adding one line to the `sys` file, defining a new outgoing feed listing all the relevant groups and distributions, and specifying the commandline to be executed which is supposed to send out the outgoing message to that ``feed." This command, in our case, should be a mail−sending program, *e.g.* `/bin/mail user@somewhere.com`. This is often adequate to get the job done. We are sure almost every Usenet news software system will have an equally easy way of piping the feed of a newsgroup to an email address.

6.2. Feeding email to news: the `mail2news gateway`

With our Usenet software sources has been integrated a set of scripts which we have been using for at least five years internally. This set of scripts is called `mail2news`. It contains one shellscript called `mail2news`, which takes an email message from `stdin`, processes it, and feeds the processed version to `inews`, the `stdin`−based news article injection utility of C−News. The `inews` utility accepts a new article post in its `stdin` and queues it for digestion by `newsrun` whenever it runs next.

To use `mail2news`, we assume you are using Sendmail to process incoming email. Our instructions can easily be modified to adapt to any Mail Transport Agent (MTA) of your choice. You will have to configure Sendmail or any other MTA to redirect incoming mails for the gateway to a program called `m2nmailer`, a Perlscript which accepts the incoming message in its standard input and a list of newsgroup names, space separated, on its command line. Sendmail can be easily configured to trigger `m2nmailer` this way by defining a new mailer in `sendmail.cf`, and directing all incoming emails meant for the Usenet news system to this mailer. Once you set up the appropriate rulesets for Sendmail, it automatically triggers `m2nmailer` each time an incoming email comes for the `mail2news` gateway.

The precise configuration changes to Sendmail have already been specified in the chapter titled ``Setting up C−News + NNTPd."

6.3. Using GNU Mailman as an email−NNTP gateway

TO BE ADDED LATER

6.3.1. GNU's all−singing all−dancing MLM

TO BE ADDED LATER

6.3.2. Features of GNU Mailman

TO BE ADDED LATER

6.3.3. Gateway features connecting NNTP and email

TO BE ADDED LATER

7. Security issues

It almost seems strange that we are discussing security issues in the context of Usenet news servers. Usenet news has been one of the most open and free−for−all network services traditionally. However, with the exponential growth of the Internet, all services are becoming aware of potential threats. The community of Internet intruders too has acquired new profiles: a lot of Internet intrusion attempts are program−driven, and exploit a set of ``well known'' vulnerabilities, *i.e.* vulnerabilities which have been identified by the computer security and intrusion community and published in their reports and advisories. Thus, the question of ``Why will someone attack my harmless Usenet server?'' is no longer valid. It will be attacked if it can be attacked, merely because its IP address falls in a range of addresses being targeted, perhaps.

Security issues for Usenet news servers fall into two categories. First come vulnerabilities which will allow an attacker to bring down your server or run code of his choice on it. Second come vulnerabilities which can distort or corrupt your Usenet article hierarchy, either by junk postings, unsolicited commercial messages, or forged control messages. The second category of threats is specific to Usenet news and needs Usenet−specific protection mechanisms, some of which require tapping into defence mechanisms designed by the Usenet administrator community.

7.1. Intrusion threats

Here we discuss the vulnerabilities which will allow an intruder to ``gain control'' of your Usenet server, or ``bring it down,'' either of which may be irritating, embarassing, or downright disastrous for your business or occupation.

7.1.1. Generic server vulnerabilities

Foremost among these vulnerabilities are those which render *any* server vulnerable to intrusion attempts. Most of these vulnerabilities are unrelated to Usenet news itself. For instance, if you have the Telnet service active on a server exposed to the Internet, then it is likely that systematic attempts by intruders to acquire usernames and passwords will bear fruit, using methods we will best leave to specialised texts on the subject. Once this is done, the intruder will merely ``walk into'' your server by Telnetting into it.

We will not discuss this class of vulnerabilities here any further; they belong in documents dedicated to general security issues. For further reading, check the ``Security HOWTO'', the ``Security Quickstart HOWTO'', the ``User Authentication HOWTO'', the ``VPN HOWTO'', and the ``VPN Masquerade HOWTO'' ... and that's just from the Linux HOWTO collection. As one can see, there is, if anything, a surfeit of material on this and related subjects.

There are vulnerabilities which allow an intruder to mount the so−called DoS attacks, which make your service inaccessible to legitimate users, even though it does not let the intruder in. The most publicised of these attacks were the SYNFlood and the Ping of Death attacks, both quite old and well−understood by now. A Linux server running a recent version of the kernel and properly configured, should be immune to both these attack methods. But network protocols being what they are, there are always new DoS methods being thought up, which can temporarily overload or slow down a server. Once again, the texts discussing generic security issues are the best place to study these vulnerabilities.

7.1.2. Vulnerabilities in Usenet software

Then come server vulnerabilities, if any, which are caused specifically by Usenet news software. For instance, if it was possible for an intruder to issue some string of bytes to your server's NNTP server and cause it to execute a command of the intruder's choice, then this vulnerability would be in this category.

Any server which accepts a text string as input from a client is open to the buffer overrun class of attacks, if the `gets()` C library function has been used in its code instead of the `fgets()` with a buffer size limit. This was a vulnerability made famous by the 1988 Morris Internet Worm, discussions on which can be found elsewhere. (Go Google for it if you're keen.) As far as we know, the INN NNTP server and the `nntpd` which forms part of the NNTP Reference Implementation both have no known buffer overrun vulnerabilities. This class of vulnerabilities is less significant in the case of NNTPd or INN because these daemons do not run as `root`. In fact, they would begin to cause malfunctioning of the underlying Usenet software if they ran as `root`. Therefore, even if an intrepid intruder could find some way of gaining control of these daemons, she would only be able to get into the server as user `news`, which means that she can play havoc with the Usenet installation, but no further. A daemon which runs as `root`, if compromised, can allow an intruder to take control of the operating system itself.

UUCP is generally believed to be insecure. We believe a careful configuration of Taylor UUCP plugs a lot of these vulnerabilities. One vulnerability with UUCP over TCP is that the username and password travel in plaintext form in TCP data streams, much like with Telnet or FTP. We therefore do not advise using UUCP over TCP in this manner if security is a concern at all. We recommend the use of UUCP through a SSH tunnel, with the SSH setup working only with a pre–installed public key. This way, there is no need for usernames and passwords for the SSH tunnel setup, and passwords cannot be leaked even intentionally. And the UUCP username and password then passes through this encrypted tunnel and is therefore totally superfluous for security; the preceding SSH tunnel provides a much stronger connection authentication than the UUCP username and password. And since we set up our SSH tunnels to demand key–based authentication only, it rejects any attempt to connect using usernames and passwords when the tunnel is being set up.

A third possible vulnerability is related to the back–end software which processes incoming Usenet articles. It is conceivable that an NNTP server will receive an incoming `POST` command, receive an article, and queue it for processing on the local spool; the NNTP server often does not perform any real–time processing on the incoming post. The post–processing software which periodically processes the incoming spool (the `in.coming` directory in C–News) will read this article and somehow be forced to run a command of the intruder's choice, either by buffer overrun vulnerabilities or any other means.

While this possibility exists, it appears that neither the C–News `newsrun` and family nor INN are vulnerable to this class of attempts. We base our comment on the solid evidence that both these systems have been around in an intrusion–prone world of public Usenet servers for more than a decade. INN, the newer of the two, completed one decade of life on 20 August 2002. And both these software systems had their source freely available to all, including intruders. We can be fairly certain that if vulnerabilities of this class have not been seen, it not for want of intrusion attempts.

7.2. Vulnerabilities unique to the Usenet service

There are certain security precautions that a Usenet server administrator has to take to ensure that her servers are not swamped by irritating junk or configured out of shape by spurious control messages. These vulnerabilities do not allow an intruder to run her software on your servers, but allows her to mess up your server, causing you to lose a precious weekend (or week) straightening out the mess.

7.2.1. Unsolicited commercial messages

Unsolicited commercial messages are called SPAM. There is a war against SPAM being fought in the Internet community. The biggest battlefront is in the world of email. Second to that is Usenet newsgroups.

There are many tools that Usenet administrators use in their battle against SPAM. The most important of these is the NoCeM suite. See `http://www.cm.org/` for details of NoCeM, and the newsgroup `alt.nocem.misc` for the SPAM cancel messages which NoCeM reads to identify which articles to discard. Your server will need a feed of `alt.nocem.misc` to use the NoCeM facility. These special messages are signed by NoCeM volunteers whose job is to identify SPAM articles, list their message−IDs, and then issue these deletion instruction, digitally signed with special private keys, which tell all Usenet servers to delete the SPAM messages. Your server's NoCeM software will need public key software (typically PGP) and a keyring with the public key of each NoCeM volunteer you want to accept instructions from.

Other anti−spam tools for Usenet services are listed in the Anti−SPAM Software Web page (`http://www.exit109.com/~jeremy/news/antispam.html`). The `Cleanfeed` software will clean out articles identified as SPAM. There are many others.

SPAM is such a nuisance and a drain on organisational expense pockets (by wasting bandwidth you pay for) that it is almost imperative today that every Usenet server protects itself against it. We will integrate some selected anti−SPAM measures into our integrated source distribution soon.

7.2.2. Spurious control messages

Control messages, discussed in detail earlier in Section 2.4>, instruct a Usenet server to take certain actions, like delete a message or create a newsgroup. If this facility is ``open to the public'', anyone with half a brain can forge control messages to create twenty new newsgroups, and then post thousands of articles into those groups. In the mid−nineties, we were hit by a storm of over 2,000 (two thousand) `newgroup` control messages, which rapidly taught us the danger of unprotected control messages and the protection against them.

The standard protection mechanism against this vulnerability is `pgpverify`, which can be downloaded from multiple Websites and FTP mirror sites by searching for `pgpverify` (the program) or `pgpcontrol` (the total software package). We have integrated this into our source distribution, so that our C−News works in a tightly coupled manner with `pgpverify`.

`pgpverify` works using public key cryptography, much like NoCeM, and all the official maintainers of respective Usenet group hierarchies sign control messages using their private keys. Your server will carry their public keys, and `pgpverify` will check the sign on each control message to ensure that it's from the official maintainer of the hierarchy. It will then act upon legit control messages and discard the spurious ones.

In today's nuisance−ridden Usenet environment, no sane Usenet server administrator receiving a feed of ``public'' hierarchies and control messages will even dream of running her server without `pgpverify` protection.

---

8. Access control in NNTPd

The original NNTPd had host−based authentication which allowed clients connecting from a particular IP address to read only certain newsgroups. This was very clearly inadequate for enterprise deployment on an Intranet, where each desktop computer has a different IP address, often DHCP−assigned, and the mapping between person and desktop is not static.

What was needed was a user−based authentication, where a username and password could be used to authenticate the user. Even this was provided as an extension to NNTPd, but more was needed. The corporate IS manager needs to ensure that certain Usenet discussion groups remain visible only to certain people. This authorisation layer was not available in NNTPd. Once authenticated, all users could read all newsgroups.

We have extended the user−based authentication facility in NNTPd in some (we hope!) useful ways, and we have added an entire authorisation layer which lets the administrator specify which newsgroups each user can read. With this infrastructure, we feel NNTPd is fit for enterprise deployment and can be used to handle corporate document repositories, messages, and discussion archives. Details are given below.

8.1. Host−based access control

TO BE ADDED LATER

8.2. User authentication and authorisation

8.2.1. The NNTPd password file

TO BE ADDED LATER

8.2.2. Mapping users to newsgroups

TO BE ADDED LATER

8.2.3. The `X−Authenticated−Author` article header

TO BE ADDED LATER

8.2.4. Other article header additions

TO BE ADDED LATER

9. Components of a running system

This chapter reviews the components of a running CNews+NNTPd server. Analogous components will be found in an INN–based system too. We invite additions from readers familiar with INN to add their pieces to this chapter.

9.1. `/var/lib/news`: the CNews control area

This directory is more popularly known as `$NEWSCTL`. It contains configuration, log and status files. There are no articles or binaries kept here. Let's see what some of the files are meant for. Control files are dealt in slightly greater detail in "Section 4.3>"

- `sys`: One line per system/NDN listing all the newsgroup hierarchies each system subscribes to. Each line is prefixed with the system name and the one beginning with ME: indicates what we are going to receive. Look up manpage of `newssys`.
- `explist`: This file has entries indicating articles of which newsgroup expire and when and if they have to be archived. The order in which the newsgroups are listed is important. See manpage of `expire` for file format.
- `batchparms`: Details of how to feed other sites/NDN, like the size of batches, the mode of transmission (UUCP/NNTP) are specified here. manpage to refer: `newsbatch`.
- `controlperm`: If you wish to authenticate a control message before any action is taken on it, you must enter authentication–related information here. The `controlperm` manpage will list all the fields in detail.
- `mailpaths`: It features the e–mail address of the moderator for each newsgroup who is responsible for approving/disapproving articles posted to moderated newsgroups. The sample `mailpaths` file in the `tar` will give you an idea of how entries are made.
- `nntp_access/user_access`: These files contain entries of servernames and usernames on whom restrictions will apply when accessing newsgroups. Again, the sample file in the tarball shall explain the format of the file.
- `log, errlog`: These are log files that keep growing large with each batch that is received. The `log` file has one entry per article telling you if it has been accepted by your news server or rejected. To understand the format of this file, refer to Chapter 2.2 of the `CNews` guide. Errors, if any, while digesting the articles are logged in `errlog`. These log files have to be rolled as the files hog a lot of disk space.
- `nntplog`: This file logs information of the `nntpd` giving details of when a connection was established/broken and what commands were issued. This file needs to be configured in `syslog` `syslogd` should be running.
- `active`: This file has one line per newsgroup to be found in your news server. Besides other things, it tells you how many articles are currently present in each newsgroup. It is updated when each batch is digested or when articles are expired. The `active` manpage will furnish more details about other paramaters.
- `history`: This file, again, contains one line per article, mapping `message-id` to newsgroup name and also giving its associated article number in that newsgroup. It is updated each time a feed is digested and when `doexpire` is run. Plays a key role in loop–detection and serves as an article database. Read manpage of `newsdb`, `doexpire` for the file format
- `newsgroups`: It has a one–line description for each newsgroup explaining what kind of posts go into each of them. Ideally speaking, it should cover all the newsgroups found in the `active` file.
- *Miscellaneous files*: Files like `mailname`, `organisation`, `whoami` contain information required for forming some of the headers of an article. The contents of `mailname` form the `From:` header

and that of `organisation` form the `Organisation:` header. `whoami` contains the name of the news system. Refer to chapter 2.1 of `guide.ps` for a detailed list of files in the `$NEWSCTL` area. Read `RFC 1036` for description of article headers .

9.2. `/var/spool/news`: the article repository

This is also known as the `$NEWSARTS` or `$NEWSSPOOL` directory. This is where the articles reside on your disk. No binaries or control files should belong here. Enough space should be allocated to this directory as the number of articles keep increasing with each batch that is digested. An explanation of the following sub−directories will give you an overview of this directory:

- `in.coming`: Feeds/batches/articles from NDNs on their arrival and before being processed reside in this directory. After processing, they appear in `$NEWSARTS` or in its `bad` sub−directory if there were errors.
- `out.going`: This directory contains batches/feeds to be sent to your NDNs *i.e.* feeds to be pushed to your neighbouring sites reside here before they are transmitted. It contains one sub−directory per NDN mentioned in the `sys` file. These sub−directories contain files called `togo` which contain information about the article like the `message-id` or the article number that is queued for transmission.
- >*newsgroup directories*: For each newsgroup hierarchy that the news server has subscribed to, a directory is created under `$NEWSARTS`. Further sub−directories are created under the parent to hold articles of specific newsgroups. For instance, for a newsgroup like `comp.music.compose`, the parent directory `comp` will appear in `$NEWSARTS` and a sub−directory called `music` will be created under `comp`. The `music` sub−directory shall contain a further sub−directory called `compose` and all articles of `comp.music.compose` shall reside here. In effect, article 242 of newsgroup `comp.music.compose` shall map to file `$NEWSARTS/comp/music/compose/242`.
- `control`: The control directory houses only the control messages that have been received by this site. The control messages could be any of the following: `newgroup, rmgroup, checkgroup` and `cancel` appearing in the subject line of the article. More information to be found in "Section 2.4>"
- `junk`: The `junk` directory contains all articles that the news server has received and has decided, after processing, that it does not belong to any of the hierarchies it has subscribed to. The news server transfers/passes all articles in this directory to NDNs that have subscribed to the `junk` hierarchy.

9.3. `/usr/lib/newsbin`: the executables

TO BE ADDED LATER

9.4. `crontab and cron jobs`

The heart of the Usenet news server is the various scripts that run at regular intervals processing articles, digesting/rejecting them and transmitting them to NDNs. I shall try to enumerate the ones that are important enough to be cronned. :)

- `newsrun`: The key script. This script picks the batches in the `in.coming` directory, uncompresses them if necessary and feeds it to `relaynews` which then processes each article digesting and batching them and logging any errors. This script needs to run through `cron` as frequently as you want the feeds to be digested. Every half hour should suffice for a non−critical requirement.

- sendbatches: This script is run to transmit the togo files formed in the out.going directory to your NDNs. It reads the batchparms file to know exactly how and to whom the batches need to be transmitted. The frequency, again, can be set according to your requirements. Once an hour should be sufficient.
- newsdaily: This script does maintenance chores like rolling logs and saving them, reporting errors/anomalies and doing cleanup jobs. It should typically run once a day.
- newswatch: This looks for news problems at a more detailed level than newsdaily like looking for persistent lock files or unattended batches, determining space shortage issues, and the likes. This should typically run once every hour. For more on this and the above, read the newsmaint manpage.
- doexpire: This script expires old articles as determined by the control file explist and updates the active file. This is necessary if you do not want unnecessary/unwanted articels hogging up your disk space. Run it once a day. Manpage: expire
- newsrunning off/on: This script shuts/starts off the news server for you. You could choose to add this in your cron job if you think the news server takes up lots of CPU time during peak hours and you wish to keep a check on it.

9.5. newsrun and relaynews: digesting received articles

The heart and soul of the Usenet News system, newsrun just picks up the batches/ articles in the in.coming directory of $NEWSARTS and uncompresses them (if required) and calls relaynews. It should run from cron.

relaynews picks up each article one by one through stdin, determines if it belongs to a subscribed group by looking up sys file, looks in the history file to determine that it does not already exist locally, digests it updating the active and history file and batches it for neighbouring sites. Logs errors on encountering problems while processing the article and takes appropriate action if it happens to be a control message. More info in manpage of relaynews.

9.6. doexpire and expire: removing old articles

A good way to get rid of unwanted/old articles from the $NEWSARTS area is to run doexpire once a day. It reads the explist file from the $NEWSCTL directory to determine what articles expire today. It can archive the said article if so configured. It then updates the active and the history file accordingly. If you wish to retain the article entry in the history file to avoid re–digesting it as a new article after having expired it, add a special */expired/;* line in the control file. More on the options and functioning in the expire manpage.

9.7. nntpd and msgidd: managing the NNTP interface

As has already been discussed in the chapter on setting up the software, nntpd is a TCP–based server daemon which runs under inetd. It is fired by inetd whenever there's an incoming connection on the NNTP port, and it takes over the dialogue from there. It reads the C–News configuration and data files in $NEWSCTL, article files from $NEWSARTS>, and receives incoming posts and transfers. These it dutifully queues in $NEWSARTS/in.coming, either as batch files or single article files.

It is important that inetd be configured to fire nntpd as user news, not as root like it does for other daemons like telnetd or ftpd. If this is not done correctly, a lot of problems can be caused in the functioning of the C–News system later.

nntpd is fired each time a new NNTP connection is received, and dies once the NNTP client closes its connection. Thus, if one nntpd receives a few articles by an incoming batch feed (not a POST but an XFER), then another nntpd will not know about the receipt of these articles till the batches are digested. This will hamper duplicate newsfeed detection if there are multiple upstream NDNs feeding our server with the same set of articles over NNTP. To fix this, nntpd uses an ally: msgidd, the message ID daemon. This daemon is fired once at server bootup time through newsboot, and keeps running quietly in the background, listening on a named Unix socket in the $NEWSCTL area. It keeps in its memory a list of all message IDs which various incarnations of nntpd have asked it to remember.

Thus, when one copy of nntpd receives an incoming feed of news articles, it updates msgidd with the message IDs of these messages through the Unix socket. When another copy of nntpd is fired later and the NNTP client tries to feed it some more articles, the nntpd checks each message ID against msgidd. Since msgidd stores all these IDs in memory, the lookup is very fast, and duplicate articles are blocked at the NNTP interface itself.

On a running system, expect to see one instance of nntpd for each active NNTP connection, and just one instance of msgidd running quietly in the background, hardly consuming any CPU resources. Our nntpd is configured to die if the NNTP connection is more than a few minutes idle, thus conserving server resources. This does not inconvenience the user because modern NNTP clients simply re–connect. If an nntpd instance is found to be running for days, it is either hung due to a network error, or is receiving a very long incoming NNTP feed from your upstream server. We used to receive our primary incoming feed from our service provider through NNTP sessions lasting 18 to 20 hours without a break, every day.

9.8. nov, the News Overview system

NOV, the News Overview System is a recent augmentation to the C–News and NNTP systems and to the NNTP protocol. This subsystem maintains a file for each active newsgroup, in which it maintains one line per current article. This line of text contains some key meta–data about the article, *e.g.* the contents of the From, Subject, Date and the article size and message ID. This speeds up NNTP response enormously. The nov library has been integrated into the nntpd code, and into key binaries of C–News, thus providing seamless maintenance of the News Overview database when articles are added or deleted from the repository.

When newsrun adds an article into starcom.test, it also updates $NEWSARTS/starcom/test/.overview and adds a line with the relevant data, tab–separated, into it. When nntpd comes to life with an NNTP client, and it sees the XOVER NNTP command, it reads this .overview file, and returns the relevant lines to the NNTP client. When expire deletes an article, it also removes the corresponding line from the .overview file. Thus, the maintenance of the NOV database is seamless.

9.9. Batching feeds with UUCP and NNTP

Some information about batching feeds has been provided in earlier sections. More will be added later here in this document.

10. Monitoring and administration

Once the Usenet News system is in place and running, the news administrator is then aided in monitoring the system by various reports generated by it. Also, he needs to make regular checks in specific directories and files to ascertain the smooth working of the system.

10.1. The `newsdaily` report

This report is generated by the script `newsdaily` which is typically run through `cron`. I shall enumerate some of the problems that are reported by it, based on my observations .

- *bad input batches*: This gives a list of articles that have been declared bad after processing and hence have not been digested. The reason for this is not given. You are expected to check each article and determine the cause.
- *leading unknown newsgroups by articles*: Newsgroup names that do not appear in the `active` file but their hierarchy has been subscribed to, would find their names mentioned under this heading. Choose to add the name in the active file if you think it is important. For *e.g.*, you would see this happen if you have subscribed to the hierarchy `comp` but the `active` does not contain the newsgroup name `comp.lang.java.3d`. You could deny subscription to this particular newsgroup by specifying so in the `sys` file.
- *leading unsubscribed newsgroups*: If the news server receives maximum articles of a particular newsgroup hierarchy to which you haven't subscribed, it will appear under this heading. You really cannot do much about this except to subscribe to them if they are required.
- *leading sites sending bad headers*: This will list your NDNs who are sending articles with malformed/insufficient headers.
- *leading sites sending stale/future/misdated news*: This will list your NDNs who are sending you articles that are older than the date you have specified for accepting feeds.
- Some of the reports generated by us: We have modified the newsdaily script to include some more statistics.
    - *disk usage*: This reports the size in bytes of the `$NEWSARTS` area. If you are receiving feeds regularly, you should see this figure increasing.
    - *incoming feed statistics*: This reports the number of articles and total bytes recevied from each of your NDNs.
    - *NNTP traffic report*: The output of nestor has also been included in this report which gives details of each `nntp` connection and the overall performance of the network connection read from the `newslog` file. To understand the format, read the manpage of `nestor`.
- *Error reporting from the `errorlog` file*: Reports errors logged in the `errorlog` file. Usually these are file ownership or file missing problems which can be easily handled.

10.2. Crisis reports from `newswatch`

Most of the problems reported to me are those with either space shortage or persistent locks. There are instances when the scripts have created locks files and have aborted/terminated without removing them. Sometimes they are innocuous enough to be deleted but this should be determined after a careful analysis. They could be an indication of some part of the system not working correctly. For *e.g.* I would receive this error message when sendbatches would abnormally terminate trying to transmit huge togo files. I had to determine why sendbatches was failing this often.

The space shortage issue has to be addressed immediately. You could delete unwanted articles by running `doexpire` or add more disk space at the OS level.

10.3. Disk space

The `$NEWSBIN` area occupies space that is fixed. Since the binaries do not grow once installed, you do not have to worry about disk shortage here. The areas that take up more space as feeds come in are `$NEWSCTL` and `$NEWSARTS`. The `$NEWSCTL` has log files that keep growing with each feed. As the articles are digested in huge numbers, the `$NEWSARTS` area continues to grow. Also, you will need space if you have chosen to archive articles on expiry. Allocate a few GB of disk space for `$NEWSARTS` depending on the number of hierarchies you are subscribing and the feeds that come in everyday. `$NEWSCTL` grows to a lesser proportion as compared to `$NEWSARTS`. Allocate space for this accordingly.

10.4. CPU load and RAM usage

With modern C–News and NNTPd, there is very little usage of these system resources for processing news article flow. Key components like `newsrun` or `sendbatches` do not load the system much, except for cases where you have a very heavy flow of compressed outgoing batches and the compression utility is run by `sendbatches` frequently. `newsrun` is amazingly efficient in the current C–News release. Even when it takes half an hour to digest a large consignment of batches, it hardly loads the CPU of a slow Pentium 200 MHz CPU or consumes much RAM in a 64 MB system.

One thing which does slow down a system is a large bunch of users connecting using NNTP to browse newsgroups. We do not have heuristic based figures off–hand to provide a guidance figure for resource consumption for this, but we have found that the load on the CPU and RAM for a certain number of active users invoking `nntpd` is more than with an equal number of users connecting to the POP3 port of the same system for pulling out mailboxes. A few hundred active NNTP users can really slow down a dual–P–III Intel Linux server, for instance. This loading has no bearing on whether you are using INN or `nntpd`; both have practically identical implementations for NNTP *reading* and differ only in their handling of feeds.

Another situation which will slow down your Usenet news server is when downstream servers connect to you for pulling out NNTP feeds using the pull method. This has been mentioned before. This can really load your server's I/O system and CPU.

10.5. The `in.coming/bad` directory

The `in.coming` directory is where the batches/articles reside when you have received feeds from your NDN and before processing happens. Checking this directory regularly to see if there are batches is a good way of determining that feeds are coming in. The batches and articles have different nomenclature. Batches, typically, have names like *nntp.GxhsDj* and individual articles are named beginning with digits like *0.10022643380.t*

The `bad` sub–directory under `in.coming` holds batches/articles that have encountered errors when they were being processed by `relaynews`. You will have to look at the individual files in this directory to determine the cause . Ideally speaking, this directory should be empty.

10.6. Long pending queues in `out.going`

TO BE ADDED.

10.7. Problems with `nntpxmit` and `nntpsend`

TO BE ADDED.

10.8. The `junk` and `control` groups

Control messages are those that have a `newgroup/rmgroup/cancel/checkgroup` in their subject line. Such messages result in `relaynews` calling the appropriate script and on execution a message is mailed to the admin about the action taken. These control messages are stored in the `control` directory of `$NEWSARTS`. For the propogation of such messages, one must subscribe to the `control` hierarchy.

When your news system determines that a certain article has not been subscribed by you, it is "junked" i.e. such articles appear in the junk directory. This directory plays a key role in transferring articles to your NDNs as they would subscribe to the junk hierarchy to receive feeds. If you are a leaf node, there is no reason why articles should pile here. Keep deleting them on a daily basis.

11. Usenet news clients

This HOWTO was written to allow a Linux system administrator provide the Usenet news service to readers of those articles. The rest of this HOWTO focuses on the server−end software and systems, but one chapter dedicated to the clients does not seem disproportionate, considering that the *raison d'etre* of Usenet news servers is to serve these clients.

The overwhelming majority of clients are software programs which access the article database, either by reading `/var/spool/news` on a Unix system or over NNTP, and allow their human users to read and post articles. We can therefore probably term this class of programs UUA, for Usenet User Agents, along the lines of MUA for Mail User Agents.

There are other special−purpose clients, which either pull out articles to copy or transfer somewhere else, or for analysis, *e.g.* a search engine which allows you to search a Usenet article archive, like Google (`www.google.com`) does.

This chapter will discuss issues in UUA software design, and bring out essential features and efficiency and management issues. What this chapter will certainly *never* attempt to do is catalogue all the different UUA programs available in the world −−− that is best left to specialised catalogues on the Internet.

This chapter will also briefly cover special−purpose clients which transfer articles or do other special−purpose things with them.

11.1. Usenet User Agents

11.1.1. Accessing articles: NNTP or spool area?

TO BE ADDED LATER

11.1.2. Threading

TO BE ADDED LATER

11.1.3. Quick reading features

TO BE ADDED LATER

11.2. Clients that transfer articles

We will discuss Suck and `nntpxfer` from the NNTP server distribution here. Suck has already discussed earlier. We will be happy to take contributed additions that discuss other client software.

11.3. Special clients

11.3.1. NNTPCache

NNTPCache is an interesting transparent cacheing proxy for news articles. News articles are read−only by definition, *i.e.* they do not change once they are posted; they can only be deleted. NNTPCache uses this feature to build a local cache of news articles.

You set up NNTPCache to listen on the NNTP port of your local Unix server, and act like an NNTP daemon. You configure it to connect back−to−back to another NNTP daemon, further away, which has all the interesting stuff the users want to read. When a user connects to the local NNTPCache, it connects to the remote NNTP server and acts as a relay for the NNTP connection, ferrying commands and responses back and forth. What the user sees therefore comes from the remote server, the first time. However, all news articles fetched by NNTPCache are also stored in a local cache, thus allowing the next user to browse the same set of articles faster. Like all demand−driven caches, the advantage here is that the local NNTPCache does not need (much) administering, and will automatically delete all articles from its cache once they've been lying unread long enough.

We list it here as an NNTP client because every proxy server is a server on one side and a client on the other.

12. Our perspective

This chapter has been added to allow us to share our perspective on certain technical choices. Certain issues which are more a matter of opinion than detail, are discussed here.

12.1. Efficiency issues of NNTP

To understand why NNTP is often an inappropriate choice for newsfeeds, we need to understand TCP's sliding window protocol and the nature of NNTP. NNTP is an apalling waste of bandwidth for most bulk article transfer situations, because of the following simple reasons:

- *No compression*: articles are transferred in plain text.
- *No article transmission restart*: if a connection breaks halfway through an article, the next round will have to start with the beginning of the article.
- *Ping−pong protocol*: NNTP is unsuitable for bulk streaming data transfer because the TCP sliding window feature is unusable with NNTP.

What is a ping−pong protocol? TCP uses a sliding window mechanism to pump out data in one direction very rapidly, and can achieve near wire speeds under most circumstances. However, this only works if the application layer protocol can aggregate a large amount of data and pump it out without having to stop every so often, waiting for an ack or a response from the other end's application layer. This is precisely why sending one file of 100 Mbytes by FTP takes so much less clock time than 10,000 files of 10 Kbytes each, all other parameters remaining unchanged. The trick is to keep the sliding window sliding smoothly over the outgoing data, blasting packets out as fast as the wire will carry it, without ever allowing the window to empty out while you wait for an ack. Protocols which require short bursts of data from either end constantly, *e.g.* in the case of remote procedure calls, are called ``ping pong protocols'' because they remind you of a table−tennis ball.

With NNTP, this is precisely the problem. The average size of Usenet news messages, including header and body, is 3 Kbytes. When thousands of such articles are sent out by NNTP, the sending server has to send the message ID of the first article, then wait for the receiving server to respond with a ``yes'' or ``no.'' Once the sending server gets the ``yes'', it sends out that article, and waits for an ``ok'' from the receiving server. Then it sends out the message ID of the second article, and waits for another ``yes'' or ``no.'' And so on. The TCP sliding window never gets to do its job.

This sub−optimal use of TCP's data pumping ability, coupled with the absence of compression, make for a protocol which is great for synchronous connectivity, *e.g.* for news reading or real−time updates, but very poor for batched transfer of data which can be delayed and pumped out. All these are precisely reversed in the case of UUCP over TCP.

To decide which protocol, UUCP over TCP or NNTP, is appropriate for your server, you must address two questions:

1. How much time can your server afford to wait from the time your upstream server receives an article to the time it passes it on to you?
2. Are you receiving the same set of hierarchies from multiple next−door neighbour servers, *i.e.* is your newsfeed flow pattern a mesh instead of a tree?

If your answers to the two questions above are ``messages cannot wait'' and ``we operate in a mesh'', then

NNTP is the correct protocol for your server to receive its primary feed(s).

In most cases, carrier−class servers operated by major service providers do not want to accept even a minute's delay from the time they receive an article to the time they retransmit it out. They also operate in a mesh with other servers operated by their own organisations (*e.g.* for redundancy) or others. They usually sit very close to the Internet backbone, *i.e.* with Tier 1 ISPs, and have extremely fast Internet links, usually more than 10 Mbits/sec. The amount of data that flows out of such servers in outgoing feeds is more than the amount that comes in, because each incoming article is retained, not for local consumption, but for retransmission to others lower down in the flow. And these servers boast of a retransmission latency of less than 30 seconds, *i.e.* I will retransmit an article to you within 30 seconds of my having received it.

However, if your server is used by a company for making Usenet news available for its employees, or by an institute to make the service available for its students and teachers, then you are not operating your server in a mesh pattern, nor do you mind it if messages take a few hours to reach you from your upstream neighbour.

In that case, you have enormous bandwidth to conserve by moving to UUCP. Even if, in this Internet−dominated era, you have no one to supply you with a newsfeed using dialup point−to−point links, you can pick up a compressed batched newsfeed using UUCP over TCP, over the Internet.

In this context, we want to mention Taylor UUCP, an excellent UUCP implementation available under GNU GPL. We use this UUCP implementation in preference to the bundled UUCP systems offered by commercial Unix vendors even for dialup connections, because it is far more stable, high performance, and always supports file transfer restart. Over TCP/IP, Taylor is the only one we have tried, and we have no wish to try any others.

Apart from its robustness, Taylor UUCP has one invaluable feature critical to large Usenet batch transfers: file transfer restart. If it is transferring a 10 MB batch, and the connection breaks after 8 MB, it will restart precisely where it left off last time. Therefore, no bytes of bandwidth are wasted, and queues never get stuck forever.

Over NNTP, since there is no batching, transfers happen one article at a time. Considering the (relatively) small size of an article compared to multi−megabyte UUCP batches, one would expect that an article would never pose a major problem while being transported; if it can't be pushed across in one attempt, it'll surely be copied the next time. However, we have experienced entire NNTP feeds getting stuck for days on end because of one article, with logs showing the same article breaking the connection over and over again while being transferred [1]. Some rare articles can be more than a megabyte in size, particularly in `comp.binaries`. In each such incident, we have had to manually edit the queue file on the transmitting server and remove the offending article from the head of the queue. Taylor UUCP, on the other hand, has never given us a single hiccup with blocked queues.

We feel that the overwhelming majority of servers offering the Usenet news service are at the leaf nodes of the Usenet news flow, not at the heart. These servers are usually connected in a tree, with each server having one upstream ``parent node'', and multiple downstream ``child nodes.'' These servers receive their bulk incoming feed from their upstream server, and their users can tolerate a delay of a few hours for articles to move in and out. If your server is in this class, we feel you should consider using UUCP over TCP and transfer compressed batches. This will minimise bandwidth usage, and if you operate using dialup Internet connections, it will directly reduce your expenses.

A word about the link between mesh−patterned newsfeed flow and the need to use NNTP. If your server is receiving primary −−− as against trickle −−− feeds from multiple next−door neighbours, then you have to use NNTP to receive these feeds. The reason lies in the way UUCP batches are accepted. UUCP batches are received in their entirety into your server, and then they are uncompressed and processed. When the sending server is giving you the batch, it is not getting a chance to go through the batch article by article and ask your server whether you have or don't have each article. This way, if multiple servers give you large feeds for the same hierarchies, then you will be bound to receive multiple copies of each article if you go the UUCP way. All the gains of compressed batches will then be neutralised. NNTP's `IHAVE` and `SENDME` dialogue in effect permits precisely this double−check for each article, and thus you don't receive even a single article twice.

For Usenet servers which connect to the Internet periodically using dialup connections to fetch news, the UUCP option is especially important. Their primary incoming newsfeed cannot be pushed into them using queued NNTP feeds for reasons described in the above paragraph These hapless servers are usually forced to pull out their articles using a pull NNTP feed, which is often very slow. This may lead to long connect times, repeat attempts after every line break, and high Internet connection charges.

On the other hand, we have been using UUCP over TCP and `gzip`'d batches for more than five years now in a variety of sites. Even today, a full feed of all eight standard hierarchies, plus the full `microsoft`, `gnu` and `netscape` hierarchies, minus `alt` and `comp.binaries`, can comfortably be handled in just a few hours of connect time every night, dialing up to the Internet at 33.6 or 56 Kbits/sec. We believe that the proverbial `full feed' with all hierarchies including `alt` can be handled comfortably with a 24−hour link at 56 Kbits/sec, provided you forget about NNTP feeds. We usually get compression ratios of 4:1 using `gzip -9` on our news batches, incidentally.

12.2. C−News+NNTPd or INN?

INN and CNews are the two most popular free software implementations of Usenet news. Of these two, we prefer CNews, primarily because we have been using it across a very large range of Unixen for more than one decade, starting from its earliest release −−− the so−called ``Shellscript release'' −−− and we have yet to see a need to change.[2]

We have seen INN, and we are not comfortable with a software implementation which puts in so much of functionality inside one executable. This reminds us of Windows NT, Netscape Communicator, and other complex and monolithic systems, which make us uncomfortable with their opaqueness. We feel that CNews' architecture, which comprises many small programs, intuitively fits into the Unix approach of building large and complex systems, where each piece can be understood, debugged, and if needed, replaced, individually.

Secondly, we seem to see the move towards INN accompanied by a move towards NNTP as a primary newsfeed mechanism. This is no fault of INN; we suspect it is a sort of cultural difference between INN users and CNews users. We find the issue of UUCP versus NNTP for batched newsfeeds a far more serious issue than the choice of CNews versus INN. We simply cannot agree with the idea that NNTP is an appropriate protocol for bulk Usenet feeds for most sites. Unfortunately, we seem to find that most sites which are more comfortable using INN seem to also prefer NNTP over UUCP, for reasons not clear to us.

Our comments should not be taken as expressing any reservation about INN's quality or robustness. Its popularity is testimony to its quality; it most certainly ``gets the job done'' as well as anything else. In addition, there are a large number of commercial Usenet news server implementations which have started with the INN code; we do not know of any which have started with the CNews code. The Netwinsite DNews system and the Cyclone Typhoon, we suspect, both are INN−spired.

We will recommend CNews and NNTPd over INN, because we are more comfortable with the CNews architecture for reasons given above, and we do not run carrier−class sites. We will continue to support, maintain and extend this software base, at least for Linux. And we see no reason for the overwhelming majority of Usenet sites to be forced to use anything else. Your viewpoints welcome.

Had we been setting up and managing carrier–class sites with their near–real–time throughput requirements, we would probably not have chosen CNews. And for those situations, our opinion of NNTP versus compressed UUCP has been discussed in Section 12.1>

Suck and Leafnode have their place in the range of options, where they appear to be attractive for novices who are intimidated by the ``full blown'' appearance of CNews+NNTPd or INN. However, we run CNews + NNTPd even on Linux laptops. We suspect INN can be used this way too. We do not find these ``full blown'' implementations any more resource hungry than their simpler cousins. Therefore, other than administration and configuration familiarity, we don't see any other reason why even a solitary end–user will choose Leafnode or Suck over CNews+NNTPd. As always, contrary opinions invited.

13. Usenet software: a historical perspective

This section comprises excerpts from a well–known Usenet Periodic Posting document which was last changed in Feb 1998. Our copy of that old document was picked up from

`ftp://rtfm.mit.edu/pub/usenet-by-hierarchy/news/software/b/Usenet_Software:_History_and_Sources`

We suspect other copies will also be found elsewhere. The physical file on the FTP server appears to have been touched last on 29 Dec 1999. The first few lines of the archived file provide information about the origin of this document and its authors:

```
Date: Tue, 28 Dec 1999 09:00:19 GMT
Supersedes: <FMMECL.58s@tac.nyc.ny.us>
Expires: Fri, 28 Jan 2000 09:00:19 GMT
Message-ID: <FnG10J.HAo@tac.nyc.ny.us>
From: netannounce@deshaw.com (Mark Moraes)
Subject: Usenet Software: History and Sources
Newsgroups: news.admin.misc,news.announce.newusers,news.software.readers,news.software.b,news.answers
Followup-To: news.newusers.questions
Approved: netannounce@deshaw.com (Mark Moraes)

Archive-name: usenet/software/part1
Original-from: spaf@cs.purdue.edu (Gene Spafford)
Comment: edited until 5/93 by spaf@cs.purdue.edu (Gene Spafford)
Last-change: 9 Feb 1998 by netannounce@deshaw.com (Mark Moraes)
Changes-posted-to: news.admin.misc,news.misc,news.software.readers,news.software.b,news.answers
```

We have been seeing this document as a periodic posting in `news.announce.newusers` since the early nineties, and it has always been our final reference on the history of Usenet server software. We reproduce excerpts below, retaining the portions which discuss server software, and removing discussions of client software, newsreaders, software for non–Unix operating systems, *etc.* All quoted portions are reproduced unedited other than changing FTP file paths to the modern URL format. We have added our comments emphasised, in separate paragraphs. We feel the information captured here is essential reading for anyone interested in Usenet server software.

If anyone can point us to a fresher version of this document, in case it is still maintained, we will be happy to refer to that version instead of this one, though we suspect the reader will not suffer due to the four–year gap; most of the information reproduced below is historical anyway.

13.1. The quoted excerpts

Currently, Usenet readers interact with the news using a number of software packages and programs. This article mentions the important ones and a little of their history, gives pointers where you can look for more information and ends with some special notes about ``foreign'' and ``obsolete'' software. At the very end is a list of sites from which current versions of the Usenet software may be obtained.

...

13.1.1. History

Usenet came into being in late 1979, shortly after the release of V7 Unix with UUCP. Two Duke University grad students in North Carolina, Tom Truscott and Jim Ellis, thought of hooking computers together to exchange information with the Unix community. Steve Bellovin, a grad student at the University of North Carolina, put together the first version of the news software using shell scripts and installed it on the first two sites: `unc` and `duke`. At the beginning of 1980 the network consisted of those two sites and `phs` (another machine at Duke), and was described at the January Usenix conference. Steve Bellovin later rewrote the scripts into C programs, but they were never released beyond `unc` and `duke`. Shortly thereafter, Steve Daniel did another implementation in C for public distribution. Tom Truscott made further modifications, and this became the ``A'' news release.

In 1981 at U. C. Berkeley, grad student Mark Horton and high school student Matt Glickman rewrote the news software to add functionality and to cope with the ever increasing volume of news –– ``A'' News was intended for only a few articles per group per day. This rewrite was the ``B'' News version. The first public release was version 2.1 in 1982; the 1.* versions were all beta test. As the net grew, the news software was expanded and modified. The last version maintained and released primarily by Mark was 2.10.1.

Rick Adams, at the Center for Seismic Studies, took over coordination of the maintenance and enhancement of the B News software with the 2.10.2 release in 1984. By this time, the increasing volume of news was becoming a concern, and the mechanism for moderated groups was added to the software at 2.10.2. Moderated groups were inspired by ARPA mailing lists and experience with other bulletin board systems. In late 1986, version 2.11 of B News was released, including a number of changes to support a new naming structure for newsgroups, enhanced batching and compression, enhanced `ihave/sendme` control messages, and other features.

The final release of B News was 2.11, patchlevel 19. B News has been declared ``dead'' by a number of people, including Rick Adams, and is unlikely to be upgraded further; most Usenet sites are using C News or INN (see next paragraphs).

In March 1986 a package was released implementing news transmission, posting, and reading using the Network News Transfer Protocol (NNTP) (as specified in RFC 977). This protocol allows hosts to exchange articles via TCP/IP connections rather than using the traditional UUCP. It also permits users to read and post news (using a modified news user agent) from machines which cannot or choose not to install the Usenet news software. Reading and posting are done using TCP/IP messages to a server host which does run the Usenet software. Sites which have many workstations like the Sun and SGI, and HP products find this a convenient way to allow workstation users to read news without having to store articles on each system. Many of the Usenet hosts that are also on the Internet exchange news articles using NNTP because the load impact of NNTP is much lower than UUCP (and NNTP ensures much faster propagation).

Our comments: This remark about relative loadings of UUCP and NNTP is no longer applicable with faster machines and networks, and with hugely increased traffic volumes. Today's desktop computers, let alone servers, can all handle both NNTP and UUCP loads effortlessly, if traffic volumes can be restricted. This is partly due to performance enhancements to UUCP as embodied in Taylor UUCP, and partly due to vastly faster processors.

NNTP grew out of independent work in 1984–1985 by Brian Kantor at U. C. San Diego and Phil Lapsley at U. C. Berkeley. Primary development was done at U. C. Berkeley by Phil Lapsley with help from Erik Fair, Steven Grady, and Mike Meyer, among others. The NNTP package (now called the reference implementation) was distributed on the 4.3BSD release tape (although that was version 1.2a and out–of–date) and is also available on many major hosts by anonymous FTP. The current version is 1.5.12.2. It includes NOV (News Overview −− see below) support and runs on a wide variety of systems. It is available from `ftp.academ.com:/pub/nntp1.5/nntp.1.5.12.2.tar.gz`. For those with access to the World−Wide Web on the Internet, the WWW page `http://www.academ.com/academ/nntp.html` contains a description and news about NNTP. A different variant, called nntp−t5, implements many of the extensions provided by INN (including NOV support). It is available from `ftp.uu.net:/networking/news/nntp/nntp-t5.tar.gz`.

One widely−used version of news, known as C News, was developed at the University of Toronto by Geoff Collyer and Henry Spencer. This version is a rewrite of the lowest levels of news to increase article processing speed, decrease article expiration processing and improve the reliability of the news system through better locking, etc. The package was released to the net in the autumn of 1987. For more information, see the paper ``News Need Not Be Slow,'' published in The Winter 1987 Usenix Technical Conference proceedings. This paper is also available from `ftp://ftp.cs.toronto.edu/doc/programming/c-news.*,` and is recommended reading for all news software programmers. The most recent version of C News is the Sept 1994 ``Cleanup Release.'' C News can be obtained by anonymous ftp from its official archive site, `ftp.cs.toronto.edu:pub/c-news/c-news.tar.Z`.

Our comments: C News is no longer maintained by anyone that we know, other than ourselves. However, after fixing the remaining bugs in the source, we have not found the need for further maintenance. NNTPd from Brian Kantor and Phil Lapsley is in the same state, but we are working on enhancements to the source for access control and other functionality.

Another Usenet system, known as InterNetNews, or INN, was written by Rich Salz `(rsalz@uunet.uu.net)`. INN is designed to run on Unix hosts that have a socket interface. It is optimized for larger hosts where most traffic uses NNTP, but it does provide full UUCP support. INN is very fast, and since it integrates NNTP many people find it easier to administer only one package. The package was publicly released on August 20, 1992. For more information, see the paper ``InterNetNews: Usenet Transport for Internet Sites'' published in the June 1992 Usenix Technical Conference Proceedings. INN can be obtained from many places, including the 4.4BSD tape; its official archive site is `ftp.uu.net` in the directory `/networking/news/nntp/inn`. Rich's last official release was 1.4sec in Dec 1993.

Our comments: The original paper by Rich Salz about INN, where he proposed the design of an alternate Usenet server software, is a must−read for readers interested in Usenet server software. So is the paper by C News authors, cited before it. Most of the issues that Rich Salz had with C News, as stated in his paper, were very relevant at that time. Today, with the current version of NNTPd and the incorporation of the message ID daemon and NOV, these issues are no longer relevant, and the choice of C News+NNTPd versus INN is now based more on the level of maintenance of source code, familiarity and personal preferences than on core design factors.

In June 1995, David Barr began a series of unoffical releases of INN based on 1.4sec, integrating various bug−fixes, enhancements and security patches. His last release was 1.4unoff4, found in `ftp://ftp.math.psu.edu:/pub/INN`. This site is also the home of contributed software for INN and other news administration tools.

INN is now maintained by the Internet Software Consortium `(inn@isc.org)`. The official INN home is now `http://www.isc.org/isc/` and the latest version (1.7.2) can be obtained from `ftp://ftp.isc.org/isc/inn/`.

Our comments: The URL for the INN home page above is probably incorrect. Try http://www.isc.org/products/INN/.

Towards the end of 1992, Geoff Collyer implemented NOV (News Overview): a database that stores the important headers of all news articles as they arrive. This is intended for use by the implementors of news readers to provide fast article presentation by sorting and ``threading'' the article headers. (Before NOV, newsreaders like `trn`, `tin` and `nn` came with their own daemons and databases that used a nontrivial amount of system resources). NOV is fully supported by C News, INN and NNTP−t5. Most modern news readers use NOV to get information for their threading and article menu presentation; use of NOV by a newsreader is fairly easy, since NOV comes with sample client−side threading code.

...

Details on many other mail and news readers for MSDOS, Windows and OS/2 systems can be found in the FAQ posted to `comp.os.msdos.mail−news`.

```
<ftp://rtfm.mit.edu/pub/usenet/comp.os.msdos.mail-news/intro>
<ftp://rtfm.mit.edu/pub/usenet/comp.os.msdos.mail-news/software>
```

13.1.2. Newsfeed management software

Gup, the Group Update Program is a Unix mail−server program that lets a remote site change their newsgroups subscription on their news feed without requiring the intervention of the news administrator at the feed site. Gup operates with the INN (and likely the C News) batching mechanisms. The news administrators at the remote sites simply mail commands to gup to make changes to their own site's subscription list. The mail/interface is password protected. Gup checks the requests for valid newsgroup names, patterns that have no effect and so on. Gup's authors are Mark Delany (`markd@mira.net.au`) and Andrew Herbert (`andrew@mira.net.au`). Its official FTP location is `ftp.mira.net.au:/unix/news/gup−0.4.tar.gz`, but since that's not as well connected as UUNET, people are strongly advised to obtain it from a mirror site, *e.g.* `ftp.uu.net:/networking/news/misc/gup−0.4.tar.gz`.

`dynafeed` is a package from Looking Glass Software Limited that maintains a `.newsrc` for every remote site and generates the batches for them. Remote sites can use UUCP or run a program to change their `.newsrc` dynamically. It comes with a program that the remote site can run to monitor readership in newsgroups and dynamically update the feed list to match reader interest. The goal of this is to get a feed that sends only exactly the groups currently being read. `dynafeed` can be obtained from `ftp://ftp.clarinet.com/sources/dynafeed.tar.Z`.

13.1.3. News processing software

Software also exists to automatically archive Usenet newsgroups. The package `rkive`, written by Kent Landfield (`kent@sterling.com`) can be configured to archive news automatically based on different headers −− Archive−Name, Volume−Issue, Chronological, Subject and External−Command to name a few. It can be run in batch mode from the command line or from cron. It can also be installed in the `sys` or `newsfeeds` file to process articles as they are received. `rkive` supports local spool directories as well as NNTP based access. `rkive` is available via FTP from `ftp://ftp.sterling.com/rkive`.

Newsclip is a programming language for writing news filtering programs, from Looking Glass Software Limited, marketed by ClariNet Communications Corp. It is C−like, and translates to C, so a C compiler is required. It has data−types to represent the kinds of things found in article headers and bodies. It can maintain databases of users, message−ids, patterns, subjects, etc. These can be used to decide whether to ignore or select an article. Newsclip can either operate as a standalone program or as part of rn. It is free for non−commercial use and is available from `ftp://ftp.clarinet.com/sources/nc.tar.Z`. Contact `clari-info@clarinet.com` with a subject line of ``newsclip'' for more info.

13.1.4. Commercial software

DNEWS is a commercial product from NetWin. DNEWS licenses are provided free to educational institutions for non profit use. With DNEWS, the news is stored in a database so as not to overload the raw file system. DNEWS supports 'sucking' where only groups which users read are pulled over from the feeder site. DNEWS is currently known to run on VMS, Windows NT, Solaris, SunOS, Unixware, HP/UX. DNEWS binaries are available by anonymous FTP from `ftp://ftp.std.com/ftp/vendors/netwin/dnews` or from `http://world.std.com/~netwin/` DNEWS sources can be obtained on request, see the file `source.txt` in the FTP area for more information.

Our comments: The information on DNEWS may be dated. We have been seeing DNEWS on their own Website for quite a few years now. Check `www.netwinsite.com`. Moreover, there are other commercial Usenet server software systems available, including the one bundled with the Internet Information Server of Microsoft Windows NT and the ones from iPlanet. And for carrier class systems, there are many commercial Usenet routers available.

13.1.5. Special note on ``notes'' and old versions of news

...

``B'' news software is currently considered obsolete. Unix sites joining the Usenet should install C news or INN to ensure proper behavior and good performance. Most old B news software had compiled−in limits on the number of newsgroups and the number of articles per newsgroup; the increasing volume of news means that B news software cannot reliably cope with a moderately−full newsfeed.

14. Documentation, information and further reading

This section fills in gaps which were hard to classify under any of the previous chapters.

14.1. The manpages

The following manpages are installed automatically when our integrated software distribution is compiled and installed, listed here in no particular order:

- `badexpiry`: utility to look for articles with bad explicit Expiry headers
- `checkactive`: utility to perform some sanity checks on the `active` file
- `cnewsdo`: utility to perform some checks and then run C–News maintenance commands
- `controlperm`: configuration file for controlling responses to Usenet control messages
- `expire`: utility to expire old articles
- `explode`: internal utility to convert a master batch file to ordinary batch files
- `inews`: the program which forms the entry point for fresh postings to be injected into the Usenet system
- `mergeactive`: utility to merge one site's newsgroups to another site's `active` file
- `mkhistory`: utility to rebuild news `history` file
- `news(5)`: description of Usenet news article file and batch file formats
- `newsaux`: a collection of C–News utilities used by its own scripts and by the Usenet news administrator for various maintenance purposes
- `newsbatch`: covers all the utilities and programs which are part of the news batching system of C–News
- `newsctl`: describes the file formats and uses of all the files in `$NEWSCTL` other than the two key files, `sys` and `active`
- `newsdb`: describes the key files and directories for news articles, including the structure of `$NEWSARTS`, the `active` file, the `active.times` file, and the `history` file.
- `newsflag`: utility to change the flag or type column of a newsgroup in the `active` file
- `newsmail`: utility scripts used to send and receive newsfeeds by email. This is different from a mail–to–news gateway, since this is for communication between two Usenet news servers.
- `newsmaint`: utility scripts used by Usenet administrator to manage and maintain C–News system
- `newsoverview(5)`: file formats for the NOV database
- `newsoverview(8)`: library functions of the NOV library and the utilities which use them
- `newssys`: the important `sys` file of C–News
- `relaynews`: the `relaynews` program of C–News
- `report`: utility to generate and send email reports of errors and events from C–News scripts
- `rnews`: receive news batches and queue them for processing
- `nntpd`: The NNTP daemon
- `nntpxmit`: The NNTP batch transmit program for outgoing push feeds

14.2. Papers, documents, articles

There are certain documents and published conference papers which are a must–read for Usenet server administrators, both for their historical value and for the insight they give into Usenet server architecture in general. We list our chart–toppers here.

14.2.1. The Usenix paper on C News

This very interesting paper has been mentioned in the section titled "Usenet software: a historical perspective>". It is titled ``News Need Not Be Slow'', and is available from `ftp://ftp.cs.toronto.edu/doc/programming/c-news.*` or from our Website (`http://www.starcomsoftware.com/proj/usenet/doc/c-news.{ps,pdf}`).

It focuses on B News, analyses it for performance, and demonstrates how specific changes in design and implementation can speed things up. It is well−written, and is educative in many areas independent of Usenet news.

14.2.2. The Usenix paper on INN

This paper talks about the things that C News didn't address, and takes Usenet news processing into the world of pure Internet connectivity. Its author is Rich Salz, the author of INN, and the paper is titled ``InterNetNews: Usenet Transport for Internet Sites.'' This can be picked up from `ftp://ftp.uu.net/networking/news/nntp/inn/inn.usenix.ps.Z` or from our Website (`http://www.starcomsoftware.com/proj/usenet/doc/inn.usenix.{ps,pdf}`), uncompressed. Be warned: this PostScript file is probably missing some mandatory first−line tag like `%!PS-Adobe-1.0` and some PostScript processors can have problems with it. For instance, on our Linux boxes, `ghostview` can display it, but `kghostview` can't, which is very strange.

This paper analyses the world of Usenet servers with C News and NNTPd, in the presence of multiple parallel feeds, and proceeds to build a case for a powerful NNTP−optimised software architecture which will handle multiple parallel incoming NNTP feeds efficiently. What later INN users appear to miss sometimes when comparing C−News+NNTPd with INN, is that INN's strengths are *only* in situations which its author had specifically targeted, *i.e.* multiple parallel incoming NNTP feeds. There is no clear superiority of one system over the other in any other situation.

14.2.3. The C News guide

This document is part of the C−News source, and is available in the `c-news/doc` directory of the source tree. The `makefile` here uses `troff` and the source files to generate `guide.ps`. This C News Guide is a very well−written document and provides an introduction to the functioning of C News.

14.3. O'Reilly's books on Usenet news

O'Reilly and Associates had an excellent book that can form the foundations for understanding C−News and Usenet news in general, titled ``Managing UUCP and Usenet,'' dated 1992. This was considered a bit dated because it did not cover INN or the Internet protocols.

They have subsequently published a more recent book, titled ``Managing Usenet,'' written by Henry Spencer, the co−author of C−News, and David Lawrence, one of the most respected Usenet veterans and administrators today. The book was published in 1998 and includes both C−News and INN.

We have a distinct preference for books published by O'Reilly; we usually find them the best books on their subjects. We make no attempts to hide this bias. We recommend both books. In fact, we believe that there is very little of value in this HOWTO for someone who studies one of these books and then peruses information on the Internet.

14.4. Usenet−related RFCs

TO BE ADDED

14.5. The source code

TO BE ADDED

14.6. Usenet newsgroups

There are many discussion groups on the Usenet dedicated to the technical and non−technical issues in managing a Usenet server and service. These are:

- `news.admin.technical` Discusses technical issues about administering Usenet news
- `news.admin.policy` Discusses policy issues about Usenet news
- `news.software.b` Discusses C−News (no separate newsgroup was created after B−News gave way to C−News) source, configuration and bugs (if any)

MORE WILL BE ADDED LATER

14.7. We

We, at Starcom Software, offer the services of our Usenet news team to provide assistance to you by email, as a service to the Linux and Usenet administrator community, on a best effort basis.

We also offer you an integrated source distribution of C News, NNTPd, as discussed earlier in the section titled "Setting up C News + NNTPd>". This integrated source distribution fixes some bugs in the component packages it includes, and it comes pre−configured with ready made configuration files which allow all components to be compiled and installed on a Linux server in a manner by which they can work together (*e.g.* key directory paths are specified consistently across all components, *etc.*) This is available at `http://www.starcomsoftware.com/proj/usenet/src/`

The URL `http://www.starcomsoftware.com/proj/usenet/src/archives/` holds the original sources of some of the software components we base our distribution on. These include C News (`c−news.tar.Z`), NNTPd (`nntp.1.5.12.1.tar.Z`), and Nestor (`nestor.tar.Z`). Other components, like `pgpverify` are maintained by their current maintainers and can be obtained from their respective sites. Therefore, they are not included in our archives.

The URL `http://www.starcomsoftware.com/proj/usenet/doc/` carries copies of some of the important technical articles and Usenix papers on the subject of the Usenet.

We will endeavour to answer all queries sent to `usenet@starcomsoftware.com`, pertaining to the source distribution we have put together and its configuration and maintenance, and also pertaining to general technical issues related to running a Usenet news service off a Unix or Linux server.

We may not be in a position to assist with software components we are not familiar with, *e.g.* Leafnode, or platforms we do not have access to, *e.g.* SGI IRIX. Intel Linux will be supported as long as our group is alive;

our entire office runs on Linux servers and diskless Linux desktops.

You are not forced to be dependent on us, because neither do we have proprietary knowledge nor proprietary closed−source software. All the extensions we are currently involved in with C−News and NNTPd will immediately be made available to the Internet in freely redistributable source form.

15. Wrapping up

15.1. Acknowledgements

This HOWTO is a by−product of many years of experience setting up and managing Usenet news servers. We have learned a lot from those who have trod the path ahead of us. Some of them include the team of the ERNET Project (Educational and Research Network), which brought the Internet technology to India's academic institutions in the early nineties. We specially remember what we have learned from the *SIGSys* Group of the Department of Computer Science of the Indian Institute of Technology, Mumbai. We have also benefited enormously from the guidance we received from the Networking Group at the NCST (National Centre for Software Technology) in Mumbai, specially from Geetanjali Sampemane.

On a wider scale, our learning along the path of systems and networks started with Unix, without which our appreciation of computer systems would have remained very fragmented and superficial. Our insight into Unix came from our ``Village Elders'' in the Department of Computer Science of the IIT (Indian Institute of Technology) at Mumbai, specially from ``Hattu,'' ``Sathe,'' and ``Sapre,'' none of whom are with the IIT today, and from Professor D. B. Phatak and others, many of whom, luckily are still with the Institute.

Coming to Starcom, all the members of Starcom Software who have worked on various problems with networking, Linux, and Usenet news installations have helped the authors in understanding what works and what doesn't. Without their work, this HOWTO would have been a dry text book.

Hema Kariyappa co−authored the first couple of versions of this HOWTO, starting with v2.0.

15.2. Comments invited

Your comments and contributions are invited. We cannot possibly write all sections of this HOWTO based on our knowledge alone. Please contribute all you can, starting with minor corrections and bug fixes and going on to entire sections and chapters. Your contributions will be acknowledged in the HOWTO.

15.3. Copyright

3. Give due credit to previous authors and major contributors. If you are considering making a derived work other than a translation, it is requested that you discuss your plans with the current maintainer.

15.4. About Starcom Software Private Limited

starcom (Starcom Software Private Limited, `www.starcomsoftware.com`) has been building products and solutions using Linux and Web technology since 1996. Our entire office runs on Linux, and we have built mission−critical solutions for some of the top corporate entities in India and abroad. Our client list includes arguably the world's largest securities depository (The National Securities Depository Limited, India, `www.nsdl.com`), one of the world's top five stock exchanges in terms of trading volumes (The National Stock Exchange of India Limited, `www.nseindia.com`), and one of India's premier financial institutions listed on the NYSE. In all these cases, we have introduced them to Linux, and in many cases, we have built them their first mission−critical business applications on Linux. Contact the authors or check the Starcom Website for more information about the work we have done.

Notes

[1] This lack of a restart facility is something NNTP shares with its older cousin, SMTP, and we have often seen email messages getting stuck in a similar fashion over flaky data links. In many such networks which we manage for our clients, we have moved the inter−server mail transfer to Taylor UUCP, using UUCP over TCP.

[2] One of us did his first installation with with BNews, actually, at the IIT Mumbai. Then we rapidly moved from there to CNews Shellscript Release, then CNews Performance Release, CNews Cleanup Release, and our current release has fixed some bugs in the latest Cleanup Release.