

CHAPTER**14**

Network News Transfer Protocol (NNTP)

The Network News Transfer Protocol (NNTP) is specified as a proposed standard in RFC 977, with the NNTP message format specified in RFC 1036. RFC 977 was published February 1986, and RFC 1036 was published December 1987. It should come as no surprise that these protocols are also undergoing revision and updating by IETF working groups: The NNTP Extensions (NNTPext) workgroup addresses the NNTP specification revision and adds to that specification some of the existing commonly used extensions to NNTP. The Usenet Article Standard Update (usefor) workgroup addresses the revision of the NNTP message format specification.

Netnews is the Internet application that uses NNTP and related protocols to distribute messages to a global audience. Whereas SMTP and other email protocols support the transmission of messages between specified individuals, NNTP supports the transmission of messages from individuals to newsgroups. A newsgroup comprises a body of messages, often grouped in threads—that is, messages relating to the same topic with each message being a reply to an earlier message in the thread.

NNTP is a many-to-many protocol for the widespread distribution of news articles. Not only must the protocol support the transmission of messages from individuals using NNTP clients to NNTP servers, but also the

228 Essential Email Standards: RFCs and Protocols Made Practical

movement of copies of those messages from NNTP server to NNTP server. Netnews distribution enables individuals to post their messages to a local NNTP server and have those messages propagated across the Internet and made available to other users around the world from their own local news servers. The netnews distribution protocol efficiently floods the network of cooperating netnews servers with new messages.

SMTP and NNTP share many characteristics. For example, the NNTP message format is explicitly based on the RFC 822/822bis specification, and the protocol commands and interactions also bear some resemblance to SMTP. However, in many ways NNTP is more complex than SMTP. NNTP uses servers to disseminate messages not just to clients but also to other servers, while SMTP simply enables the transport of messages from a source to a destination. With NNTP, each message originates at a single source but that single message is delivered to an unknown number of destination servers for distribution—potentially, to millions of individual clients. Further complicating NNTP is that it's of a hierarchical newsgroup namespace. While SMTP messages are delivered to mailboxes, each NNTP message is delivered to one or more newsgroups.

This chapter begins with an overview of NNTP as a protocol: what it does, how it is structured, how it works for end users, and how clients and servers interact. The next section examines the NNTP message format, with special attention to the revisions expected to be published with RFC USEFOR, as well as discussion of how it compares to the RFC 822/822bis message format. Finally, the details of NNTP protocol commands and interactions are discussed in the last section. This section examines the basic protocol elements defined in RFC 977 and discusses additions and changes expected from RFC 977bis.

NNTP Protocol Overview

Email may be an enduring all-time favorite Internet application, but it is not always appropriate. Email works well only when the sender can name and has an email address for everyone he or she wishes to receive a particular message. There is no mechanism for sending an email message to everyone who might be interested—even though email implementations usually provide some mechanism for copying a message to everyone in an organization, department, or workgroup. The problem with these mechanisms is that they gobble bandwidth. For one thing, they cause untold numbers of messages to be transmitted, stored, viewed, and deleted by recipients who either got the message by mistake when the sender mistakenly chose to send a message to everyone in the company instead of everyone in the workgroup. These messages take up network bandwidth, email server resources, and local system resources.

For another thing, even though everyone in the organization might potentially be interested in reading the message, usually only a few people actually *are* interested in hearing about someone's vacation plans or offer of kittens. Whether they are interested or not, however, they must at least read the message description and delete it.

In real life, individuals use postal mail to communicate with specific entities about specific topics. When they want to make a general announcement to a community, whether that community includes people who live in your neighborhood at home or who work in your neighborhood at work, individuals often use a bulletin board. NNTP defines a mechanism by which individuals can post messages to a digital bulletin board and a mechanism by which individuals can choose to subscribe to those digital bulletin boards and retrieve the messages that appear to be of interest. This approach saves bandwidth and creates a forum for sharing messages for like-minded individuals. It also offers a solution for when groups need to communicate among themselves, but not everyone needs or wants to receive a copy of every message at their desktops.

History of Netnews

Before chat and the Web, the Internet and its predecessor networks provided plenty of exciting interaction through netnews. Although email offered plenty of opportunities to interact with others, they were almost always others whom you knew or had some contact with, even if it was only by email. Netnews, on the other hand, allowed anyone with access to the Internet and a news server to interact with anyone else who has similar access. The person sending a message could never be certain who would or would not see the message.

Starting around 1980, a network of cooperating news servers called Usenet arose. Usenet persists with many thousands of newsgroups available today. ISPs and other connected networks receive newsgroup articles and provide access to their users through news servers. Initially, a simple news article format called A News was defined for Usenet news. RFC 850, published in 1983, established a news article format that was similar to the current format for Internet email messages (it differed by adding headers specific to news articles and by having some more restrictions on the basic headers).

RFC 1036, "Standard for Interchange of USENET Messages," was published in 1987 and incorporated relatively minor changes in the format. In 1994, Henry Spencer wrote a more comprehensive update to the standard and published it as an Internet-Draft. This document is sometimes referred to as "Son of [RFC] 1036." Although the draft expired long ago, implementers have still been using it along with RFC 1036 as a de facto standard. Later in the 1990s, the USEFOR workgroup used it as the basis of their update of RFC 1036 (an effort sometimes referred to as "Grandson of 1036").

230 Essential Email Standards: RFCs and Protocols Made Practical

The objective of this revision, in addition to clarifications and corrections, is to formally specify extensions to the news message format. In particular, the revision effort expects to specify standards for the following:

- Using digital signatures with news articles
- Using non-ASCII (8-bit) characters in news message headers and bodies
- News message bodies and use of MIME with news
- Standardizing third-party control messages

Before the workgroup is finished, it is expected that other issues will be identified and addressed.

The news message format is not the only part of netnews that required updating. The NNTP specification in RFC 977, "Network News Transfer Protocol, A Proposed Standard for the Stream-Based Transmission of News," was published in 1986. Since then, the protocol has been shown to be limited. The original netnews application used the Unix to Unix Copy Protocol (UUCP), which allowed news servers to batch articles together for transfer from server to server. UUCP is defined in RFC 976, "UUCP Mail Interchange Format Standard." This approach enabled a more efficient use of bandwidth, but batching meant that individual servers could not verify that they actually needed or did not already have a copy of the messages they were getting. As the number of netnews messages increased, the bandwidth required to transfer the messages increased, with a resulting increase in overall Internet bandwidth used by netnews.

NNTP eliminated redundant message transfers by adding a mechanism for a receiving server to check the sending server's database to see what messages it has and which of those messages it needs. In this way, the receiving server downloads only those messages that it doesn't already have. Unfortunately, the way NNTP specifies this mechanism means that the sender and recipient must negotiate a separate protocol transaction for every netnews message. Though this does add network bandwidth, the greater impact is in the loss of protocol efficiency as messages cannot be batched and transmitted in a stream. Doing so would enable NNTP to take advantage of TCP's flow control features for finding the highest rate of transmission of a large amount of data.

What's more, implementers have been extending NNTP since the 1980s, but the standards specifications have not kept up. The NNTPEXT workgroup was tasked with defining a mechanism for adding extensions to NNTP as well as with making the usual clarifications and corrections in the existing protocol. In addition, the workgroup took on the job of defining existing extensions and reviewing extensions for inclusion in the NNTP standard.

News Architecture

As with Internet email, the netnews architecture consists of several discrete entities:

- A standard format for messages
- A protocol that defines how messages get from server to server
- A protocol that defines how clients interact with servers, both for reading messages and for posting messages
- A network of clients and servers to distribute the messages
- A mechanism for distributing the messages to all servers in the news network

Figure 14.1 illustrates the last item in this list. Netnews originally used a flooding algorithm, propagating copies of every message to every server in the news network. As mentioned earlier, this can use a lot of bandwidth, but at the same time the benefit is a reliable and rapid mechanism for distributing messages. By contrast, NNTP modifies the flooding mechanism to reduce bandwidth.

Using flooding or some modified flooding mechanism also eliminates the problems related to maintaining a huge centralized server providing access to all interested users. Instead, local servers can offer news services to local users and can be installed with appropriate hardware capable of supporting the

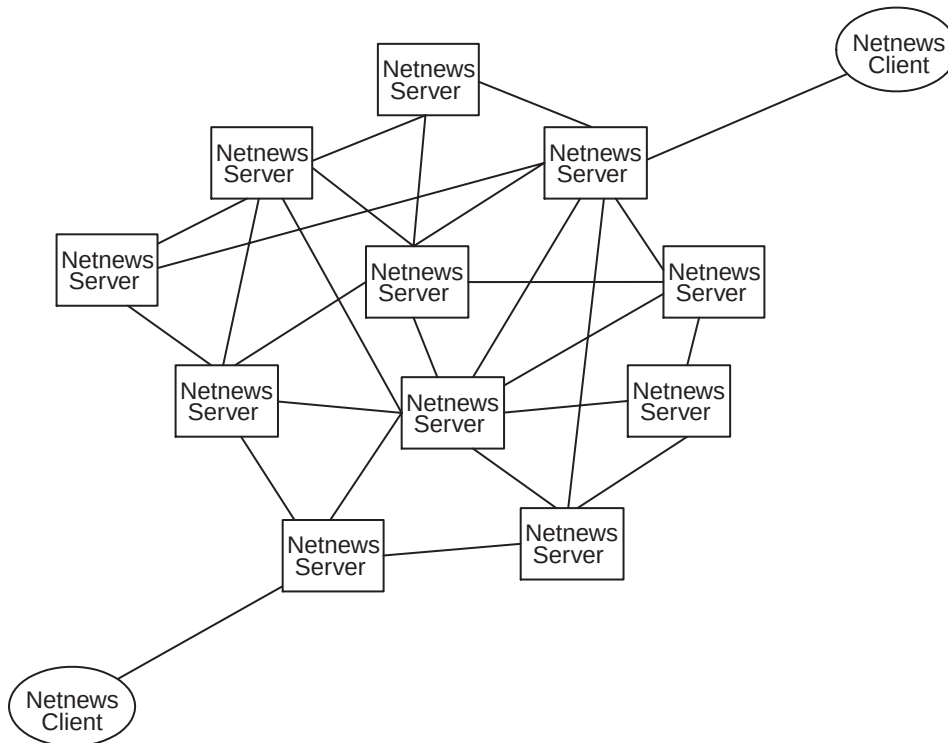


Figure 14.1 The netnews flooding distribution mechanism.

volume of requests as well as storing messages for as long as appropriate. The longer a server archives messages, the more storage it must have; terabyte (approximately 1,000 gigabytes) class storage is necessary to maintain any significant archive of Usenet postings.

The rest of this section details some of the other aspects of the netnews architecture, starting with a series of definitions taken from the USEFOR work-group's work in progress.

Netnews Terminology

So far in this chapter, we have not used any new language to describe the things that netnews does, depending largely on terms like message and server already familiar from discussions of Internet mail. However, despite similarities with mail, netnews has its own vocabulary describing all the pieces of its architecture. The terms in Table 14.1 are taken from the latest draft of the USEFOR News Article Format Internet-Draft.

Browsing through the terms listed in Table 14.1, one might assume that the netnews architecture is complex, full of interlocking and interconnecting systems and agents. In practice, however, most of the agents defined here reside

Table 14.1 Netnews Terms Defined

TERM	DEFINITION
Article	The atomic unit of netnews. It is comparable to an email message; it is created by an individual with the intention of passing it along to the netnews system.
Poster	The individual or entity that creates articles. The poster submits articles to an injecting agent (see below) for transmission into the netnews system.
Posting agent	The software used by a poster to create an article. The posting agent formats the article and adds all required and otherwise appropriate headers. It verifies that the article follows the message format standard. The posting agent also injects the article into the news stream (see below) by submitting it to an injecting agent (see below) and notifies the poster if the injecting agent rejects it for any reason.
News stream	The set of articles being transferred from one netnews server to another.
Injecting agent	This entity accepts articles from a posting agent. If it is well-formed and there is no reason not to relay it, the injecting agent passes the articles to the relaying agent for distribution. The posting agent often uses the NNTP POST command to submit articles to the injecting agent.
Relaying agent	This entity receives articles from injecting agents and/or from other relaying agents. The relaying agent may also transfer copies of the articles to other relaying agents and serving agents (see below).

Table 14.1 (Continued)

TERM	DEFINITION
Serving agent	This entity accepts articles from relaying agents and stores them in a news database. This includes offering reading agents (see below) an interface for accessing articles.
Reader	The entity accessing and reading news articles. This may be a person or a piece of software. A software-based reader might be a program that scans news articles; it is not the software that presents articles to a reader (see reading agent, below).
Reading agent	The software that is used by the reader to read articles.
Newsgroup	A news forum. Each newsgroup is intended to cover a specific topic, acting as a virtual bulletin board. Newsgroups are defined, in effect, by the articles that are posted to them. Newsgroups are given names that should indicate their topic. Articles can be posted to more than one newsgroup in two ways: by crossposting (see below) or by sending the text of the article separately to multiple newsgroups. Newsgroups may be moderated: All articles are first submitted to some person (or other entity) for review before posting. By default, newsgroups are unmoderated, but moderation may improve the general quality of postings while reducing their volume.
Crossposting	Occurs when an article is posted to multiple newsgroups. The article is sent once for all newsgroups, with headers indicating it should be distributed to all indicated newsgroups. This saves bandwidth, avoiding resending the entire article for each listed newsgroup.
Followup	A followup article contains a response to an earlier article. The earlier article is sometimes referred to as a precursor. A followup article is comparable to a message sent in reply to an earlier email message.
Followup agent	The combination of reading and posting agents that is used by the poster to create and submit a followup article.
Reply agent	The combination of reading agent and message user agent used by an individual to create and submit an email reply to an article. In other words, the reply agent makes it possible for a person to make a one-to-one response to a news article, outside the netnews system.
Message ID	The unique identifier associated with an article. This value is usually created by the posting agent. The message ID is globally unique: All articles with the same message ID are treated as if they are identical copies.
Gateway	A system that converts netnews articles to some other application format (for example, converting articles into email messages) or from some other application format into netnews.
Control message	An article containing newsgroup or article control information. This may include information about article cancellation, for example.
Reply address	The address at which the poster can be reached by email.

234 Essential Email Standards: RFCs and Protocols Made Practical

in the same systems. For example, the posting agent, reading agent, followup agent, and reply agent usually all reside in the client software used to read and write Internet email and news articles such as Netscape's Communicator or Microsoft's Outlook clients. Likewise, NNTP servers often incorporate all the functions of the injecting agent, relaying agent, and the serving agent.

The Newsgroup Name Space

The netnews architecture uses a pseudo-hierarchical name space for newsgroups—the entities within which news articles are categorized and organized. Newsgroups are organized by general categories, which may be based on very broad interests, such as sci (science), rec (recreational), or alt (alternative). They may use regional headings such as ne (New England), nyc (New York City), or us (United States). They may refer to more specific interests, corporate names, products, hobbies, or virtually anything else. Although at one time the hierarchy sprang from a relatively few top-level domains, now there are hundreds of top-level domains to choose from, with the total number of available newsgroups well into the tens of thousands.

A newsgroup name consists of at least two words separated by a period (".") and pronounced "dot." Thus, newsgroups have names like alt.folklore, comp.network.protocol.tcpip, and alt.barney.dinosaur.die.die.die. The last newsgroup name does not necessarily imply that there are other newsgroups under alt.barney.dinosaur.die.die.*, although there are other newsgroups under alt.barney.*. Thus, the hierarchy is not strict.

The commands used for creating new newsgroups are discussed later in this chapter, although it should be noted that there are policy issues related to newsgroup creation not discussed here.

News Message Format

To take advantage of existing messaging tools, RFC 1036 defined the message format for netnews articles in terms of the existing RFC 822 standard. In fact, that specification states "all USENET news messages must be formatted as valid Internet mail messages, according to the Internet standard RFC-822." Given that netnews needs more information than is necessarily provided in a minimal RFC 822 message, the news message format defines extensions for netnews articles. As we see later when discussing the impending update to the netnews article format, even more differences are apparent.

NNTP Commands and Transactions

As with SMTP and IMAP, NNTP defines different states, transactions, and response codes. NNTP provides mechanisms for clients to query servers about and request information about specific articles, including article headers and

bodies. NNTP also provides mechanisms for dealing with newsgroups as well as for requesting new news articles from servers. Clients may post new articles, and mechanisms also exist for creating, renaming, and deleting newsgroups.

RFC 977 defines NNTP commands and response codes, but is not as comprehensive as more modern Internet specifications. The latest versions of the NNTPEXT workgroup Internet-Drafts provide additional guidance to current standards for NNTP and more details about NNTP implementations and extensions.

Netnews Propagation Algorithms

Way back in the 1980s and earlier, netnews was propagated through the network by a simple flooding mechanism. To get on the Usenet netnews network, a site needed to get a news feed—a link to a server already on Usenet that would feed it new articles. Every news server forwarded any new articles it had to the news servers it fed. This worked relatively well: Most Usenet sites had relatively little bandwidth to play with, so a single interaction with each feed site kept protocol interactions to a minimum and allowed a single daily transmission in the off-peak hours. The feed data could be compressed to further conserve bandwidth. If the receiving server subscribed to more than one newsfeed, it could simply discard duplicate articles after downloading its news feed.

As volumes of Usenet news increased in the 1980s, the need for a more sophisticated distribution mechanism became apparent. Daily news feeds were not sufficient to keep up with user demands for timeliness. More sites joined Usenet, and with more news, more news servers, and more news feeds, it quickly became apparent that a simple flooding mechanism would not scale well as Usenet grew.

NNTP provides mechanisms for interaction between servers exchanging new news articles, so a server can determine whether it needs an article being offered by another server for download. With NNTP, servers seeking new articles open a TCP connection to a neighbor NNTP server. The server initiating the connection can check for the presence of new newsgroups, request a list of new news articles available from the remote server, download those articles it does not already have, and offer new articles that the initiating server has to the remote server.

The result is a distribution system that is more economical in terms of bandwidth than UUCP or other alternatives. NNTP's lack of a bulk transfer option results in a protocol that is less than completely efficient. NNTPbis documents the extensions that have been added to NNTP to remedy this deficiency, as we see later in this chapter.

News Transport Environments

Netnews is an application, and RFC 1036 defines the format for data exchanged within the netnews application. It happens that NNTP is the current protocol of

236 Essential Email Standards: RFCs and Protocols Made Practical

choice for transferring netnews articles, but it is not the only transport protocol available. UUCP may still be used to move articles from server to server, as can FTP, tape archives, or even physical delivery of news articles on magnetic or optical media.

RFC 1036/1036bis articles can be transmitted via any appropriate application protocol. Likewise, NNTP can use any transport layer protocol for its operation, even though TCP is the transport layer protocol specified for use in IP networks.

News, Mail, and MIME

The standard for news articles is effectively a subset of the Internet message standard, RFC 822bis. This means that any news article should be a well-formed message as defined in RFC 822bis, but that a well-formed message as defined in RFC 822bis will not necessarily be a valid news article. By sharing a common format, news-to-mail and mail-to-news gateways are relatively straightforward to implement, as is client software that supports both mail and news.

The MIME standards (see Chapter 9, “Multipurpose Internet Mail Extensions (MIME)”) can be applied to news articles, allowing the attachment of a wide range of application and other types of data. Just as a MIME enclosure can be included with an email message, so too can a MIME enclosure usually be included with a news article.

Usenet Message Format

Rather than repeat the discussion of the RFC 822 and RFC 822bis message format specifications, we focus instead on how the Usenet message format specification differs. We start by examining the Usenet message format, followed by a discussion of how it differs from the straight RFC 822/822bis specification, and why these differences are necessary. The second part of this section discusses what kinds of changes will likely be made in the update to the Usenet message format specification.

Usenet Message Format, RFC 822, and RFC 822bis

The schedule for the news message format revision (USEFOR) trailed the RFC 822 revision. As a result, early drafts of USEFOR had to reference a standard that itself was still an Internet-Draft.

Table 14.2 lists required and optional header fields for news articles, based on RFC 1036. The most important difference between the Internet mail message format and the Usenet article format is that Usenet articles may contain 8-bit data, while email messages are expected to be 7-bit ASCII. Neither RFC 1036 nor RFC 822 mentions character sets. However, RFC 822bis does specify that message bodies and headers must all be US-ASCII text.

Usenet Article Headers

As can be seen in Table 14.2, the Usenet article format requires one header field not specified for RFC 822 Internet mail messages, the Newsgroups: header.

Table 14.2 Required and Optional Headers for News Articles (from RFC 1036)

FIELD	REQUIRED	DESCRIPTION
From:	Yes	The email address of the person posting the article.
Date:	Yes	The date and time the article was posted.
Newsgroups:	Yes	The name or names of existing newsgroups to which the article is being posted. Multiple newsgroups are separated by a comma.
Subject:	Yes	A descriptive title for the article.
Message-ID:	Yes	A unique message identifier. This should remain unique for at least two years, preferably forever.
Path:	Yes	This header contains the names of all hosts through which the article has been passed. As a host forwards an article, it adds its name to the Path: header.
Followup-To:	No	Same format as Newsgroups: header; contains newsgroup(s) to which followup articles should be posted. If the keyword "poster" is present, newsgroup followups are not permitted, only email to the poster.
Expires:	No	A recommended expiration date for the article. This should normally be left out to allow local hosts to determine when to expire messages.
Reply-To:	No	Preferred address for replies, same format as From: header.
Sender:	No	Same format as From:, indicates actual poster of the article.
References:	No	Permitted only for followup articles. Contains message IDs of any precursor articles.
Control:	No	Indicates that the contents of the header are to be treated as a control message only.
Distribution:	No	Allows restriction on the scope of distribution of the article.
Keywords:	No	One or more words intended to indicate message contents. This header is primarily for use by people to choose articles to read.
Summary:	No	A brief summary of the message, intended for use with followup messages.

Continues

238 Essential Email Standards: RFCs and Protocols Made Practical**Table 14.2** Required and Optional Headers for News Articles (from RFC 1036) (*Continued*)

FIELD	REQUIRED	DESCRIPTION
Approved:	No	For moderated newsgroups, this header is required and consists of the moderator's email address. Also used with some control messages.
Lines:	No	Contains a number indicating the article's length, in lines.
Xref:	No	Contains the host name (not fully qualified domain name) of the news server along with newsgroups on which the article is posted and the message numbers associated with each.
Organization:	No	Intended to contain the name of the poster's organization, for the purpose of identifying the poster to people reading the article.

Further, additional optional news article headers include the Followup-To: and Expires: header fields. Usenet articles can have any other headers, as long as they conform to the RFC 822/822bis specification, so you will see headers such as NNTP-Posting-Host: as well as Content-Type: and many others in articles posted from modern news clients.

RFC 1036 provides relatively little in terms of formal ABNR protocol definitions, as with most early RFCs. Instead, it lists all the required and optional headers and discusses how they are expected to work and what they are expected to look like. The rest of this section discusses in more detail some of the headers described in Table 14.2 and in RFC 1036, as well as some of the directions in which the USEFOR workgroup will take the revision to the specification in RFC 1036bis and related specifications.

Usenet Article Character Sets

The need to support 8-bit data adds a new requirement to netnews systems: They must be 8-bit clean, meaning that they can not alter the data by stripping off the eighth bit or by removing it. Systems that modify inbound or outbound data in these ways are not able to handle netnews articles. The standard is expected to specify that all Usenet articles use the character set UTF-8. (UTF-8 is an ISO standard that supports multilingual text in a single encoding; UTF stands for "UCS Transformation Format," and UCS stands for "Universal Character Set.")

While the use of UTF-8 might appear to directly contradict the RFC 822/822bis format specification that calls for US ASCII characters only, in practice it does not necessarily break the specification. UTF-8 defines a general encoding for character sets beyond the basic US ASCII 7-bit set. However, all octets with a value of less than 128—by definition all 7-bit characters—can be interpreted directly as ASCII characters. UTF-8 is discussed in RFC 2279,

“UTF-8, a Transformation Format of ISO 10646” and RFC 2277, “IETF Policy on Character Sets and Languages,” (BCP 18) specifies that all protocols must support UTF-8. Why is RFC 822bis not supporting UTF-8? Because it is strictly a revision effort, and the workgroup is not permitted to add any new functionality, such as would be implied by updating the specification to support internationalization via UTF-8.

The Universal Multiple-Octet Coded Character Set (UCS) specification from ISO (ISO 10646) defines a mechanism by which international character sets can be encoded in one, two, or four octets. This allows non-ASCII characters from non-English Latin languages as well as characters from Cyrillic, Greek, Arabic, Chinese, and Japanese to be represented in octets whose values are 128 or higher, while leaving the characters represented by octets valued 0-127 with the same values as standard 7-bit US ASCII.

While 8-bit characters are permitted in article bodies as well as in the data stored in many article header fields, the header names themselves must still be expressed in 7-bit characters. Likewise for the rest of the article header field data: message IDs, date/time, path, and address data must all be in 7-bit characters.

Message IDs

RFC 1036 mandates the message-ID to contain a value that uniquely identifies the article across the entire netnews system. This means the message ID cannot use any value that was previously used by a different message during the original message’s lifetime. The RFC recommends that no message-ID be reused for at least two years after its first use. The format conforms to RFC 822, which requires a unique string separated from the fully qualified domain name of the originating host by the @ symbol; the ABNF representation looks like this:

```
message-id = "<" local-part "@" domain ">"
```

Beyond warnings to implementers not to assume too much about message IDs and a suggestion that the unique portion of the message ID could be a sequence number assigned by the posting agent, RFC 1036 has little more to say about ensuring that message IDs are globally unique. However, in practice getting the message IDs right—that is, unique—is not a trivial problem. One of the work items of the USEFOR group is an Internet-Draft titled “Recommendations for Generating Message IDs,” which details some mechanisms for creating message IDs that are truly globally unique and will never be reused.

Combining a sequence or process number with the fully qualified domain name of the host that originates a message or article is a reasonable first pass at the problem of creating a globally unique message ID. However, it assumes a great deal about how hosts are associated with fully qualified domain names as well as with how the local, unique, part of the message ID is generated.

The fully qualified domain name is not always available on the local host. In the past, this has meant using an IP address instead of domain name. This

240 Essential Email Standards: RFCs and Protocols Made Practical

approach can result in significant lack of uniqueness if the host is assigned its IP address through DHCP (the Dynamic Host Configuration Protocol). It means that the IP address may only be temporarily bound to the originating host. Instead, USEFOR workgroup work in progress suggests that the domain portion of the poster's return address should be used instead.

As for generating the local, unique, portion of the message ID, the most popular current method is to use the date and time that the message is posted, include a process number (or some other value) to differentiate between postings made at precisely the same time, and make sure that the values are expressed as alphanumeric data. At first glance, this should always produce a unique value; however, there are ways to improve the odds. First, let's look at some of the different strategies that have been used to guarantee uniqueness.

Simply using a sequence number (an approach not often used any more but once popular for server implementations) fails miserably to create unique message IDs the very first time an article was posted from a host whose software was reinstalled or from a computer replacing an existing system using the same host name. Likewise, using sequence numbers or even process numbers alone results in collisions in message ID when users posted from systems where the domain name was unavailable and an IP address is used instead.

Another approach is to use a pseudo-random number generator to generate some value to append to the message ID. Using eight bytes of randomness provides about 18 billion billion (that is, 2^{64} or 18,446,744,073,709,551,616) different values, making overlap a relatively rare occurrence. However, pseudo-random number generators offer no way to eliminate the (admittedly low) probability of generating the same number twice in a row. Furthermore, pseudo-random number generators must be seeded with some initializing randomness from which to begin generating pseudo-random values, which is something that has not always proven easy to implement. Improperly seeded, the generator can not be depended upon to generate random-seeming values.

Implementers can also add uniqueness to the message ID by using a cryptographically secure hashing function on the article. The function accepts as input the entire article body and generates a short (8- or 16-byte) hash value that can be appended to the unique part of the message ID.

Thus, a unique message ID can be generated by combining some or all of these mechanisms. Using the system date and time eliminates all collisions other than with messages and articles generated at the same time. Using a process number associated with the program doing the posting eliminates all collisions other than with messages generated at the same time and by the same software. Adding a pseudo-random number and a secure hash to the date/time and process number effectively reduces the probability of a collision to zero.

When the resulting value is combined with a permanent host domain name, the message ID is effectively guaranteed to be unique within all messages generated from that domain name. Even when the domain name being used is

taken from the user's email address, the unique portion of the message ID will in all likelihood be globally unique by itself.

Paths

SMTP servers relaying mail on its way to its final destination each add a Received: header to email messages. As news articles are relayed from NNTP server to server, a different mechanism provides a similar service. In both cases, the purpose is to provide a trail that can be used for troubleshooting mail or net-news routing and to track down where articles and messages came from and what systems might be responsible for problems. Unlike in SMTP, where each server adds a separate Received: header, NNTP servers each modify a single instance of the Path: header as they pass articles along. Further, the NNTP Path: header helps servers avoid sending articles to servers that have already received them. In particular, the Path: header helps prevent servers from sending copies of articles back to the server from which they were originally received.

The specification documented in RFC 1036 lacks an ABNF format for the Path: header, but it does state that each system that forwards the article adds its own entry in the Path: header. The first entity in the Path: header is placed there by the originating system. This entity stays to the right of the header, and new entries are added to the left of the first entry. The specification allows any punctuation character or characters to be used as delimiters, but virtually all news implementations now use the "!" character as delimiter.

As each host receives an article, it adds its own name to the left of the path. Consider the path:

```
nnntp.internet-standard.com!bingo.town-hall.com
```

This indicates that the article was first posted to the host bingo.town-hall.com and then sent to nnntp.internet-standard.com. When the host news.loschin.com gets a copy of this article, it adds a delimiter character ("!") and prepends its own name to the path:

```
news.loschin.com!nnntp.internet-standard.com!bingo.town-hall.com
```

RFC 1036 is not explicit about what kinds of hostnames are permitted, beyond saying that "the name each host uses to identify itself should be the same as the name by which its neighbors know it." In practice, that means using either a fully qualified domain name, IP address, or a UUCP host name. RFC 1036 also indicates that the rightmost entry can also be the sender's name, followed by the originating system's name. This option was largely included for historical reasons, but the update to RFC 1036 will almost certainly clarify its use.

Figure 14.2 shows ABNF code taken from a work in progress of the USEFOR working group, describing the Path: header format. This proposal represents some significant differences from the standard defined in RFC 1036. One

242 Essential Email Standards: RFCs and Protocols Made Practical

```

path-content      =      old-path / new-path

old-id            =      1*( ALPHA / digit / "-" | "." | "_" )

old-path          =      old-id *(punctuation old-id)

punctuation       =      LWS / %x21-2f / %x3a-40 / %x5b-60 / %x7b-7f
                        ; These are ! " # $ % & ' ( ) *
                        ; + , - . / : ; < = > ? @ [ \
                        ; ] ^ _ ` { | } ~ DEL

new-delims        =      [FWS] ("@" / "/" / "," ) [FWS]

new-path          =      post-injection "%" pre-injection

delim-plus-id     =      [FWS] "!" [FWS] old-id
                        / new-delims site-id

post-injection    =      *(site-id 1*new-delims) site-id

pre-injection     =      site-id *delim-plus-id

site-id           =      ALPHA word      ; UUCP name
                        / ALPHA          ; for "x" tail entry
                        / "." word       ; other registered name
                        / <FQDN>         ; as per RFC 1034
                        / <dotted-quad>  ; numeric IP address rep
                        ; specified in rfc820 etc.
                        / "[" dotted-quad "]"
                        / "[" <ipv6-numeric> "]" ; per RFC1884

word              =      1*(ALPHA / digit / "-" / "_")

```

Figure 14.2 Proposed ABNF encoding for the Path: header.

important step to cleaning up the specification will almost certainly be to limit the characters that may be used to delimit entries in the Path: header; another will be to allow more liberal use of folding white space to clarify article headers in general. The specification will also be clearer about how entries identify hosts in the Path: header, probably by explicitly limiting entries to UUCP host names, fully qualified domain names, and IP addresses.

It is also expected that the update to RFC 1036 will require that systems verify Path: header entries themselves, probably by using DNS to determine that the last host indicated on the Path: header matches the host from which the article was received.

Another clarification expected in this update relates to the tail entry of the Path: header—the rightmost entry in the header that was originally specified

to contain a user name. In practice, this entry rarely, if ever, points to an individual. Even if it did, it may not be used as an email address or to contact the user who posted the article. The current proposal suggests using the tail entry for authentication information about the source machine of the article and also to have the injecting agent prepend the symbol “%” to this tail entry to signal its presence and prevent it from being interpreted as a system.

Distribution

In RFC 1036, the Distribution: header field is described as a mechanism by which the scope of distribution for the article can be limited (in contrast to the Newsgroups: header, which expands the distribution scope of an article by having it listed in more than one newsgroup). For example, an article describing a car for sale in Boston but submitted to a national newsgroup might include a Distribution: line containing information that would keep the article from being distributed outside New England. Any followup articles retain the Distribution: line of the precursor article, so that a response to such an article would not be propagated outside the original area of interest. If the Distribution: header is not present, the default distribution is “world” meaning that the article should be distributed without any limitation.

The Distribution: header was specified in good faith to conserve bandwidth and keep newsgroups relevant to their users, but in practice it did not work out so well. When a news server providing a news feed to another news server got ready to send out articles, it checked a list of distributions configured for the target news server. The list contained the distributions that the target site should get. In theory, that list should have contained a list of distribution codes. The feed server would run down the list and if an article’s Distribution: header contained a distribution that matched the list, the article would be sent.

This is not how news feeds behave in the real world. Rather than configure each news feed with a list of the distributions that the remote server wants, most feeds are configured to deliver everything *except* the newsgroups local to the feed server. In other words, all of the newsgroups with an *internal* distribution would be excluded, while everything else gets fed to the remote server.

This is not so bad, overall. A person physically in the United Kingdom could be interested in what goes on in San Francisco. For example, the person might be from San Francisco and only in the UK on business, or the person could be a Briton getting ready to visit California or might have friends, relatives, or business contacts in the Bay area. If ISPs in the United Kingdom get the feed for San Francisco (and New York, and New England, and Florida, and so on), that’s a good thing as it increases the amount of content they can offer their customers. Of course, this defeats the purpose of the Distribution: header, which is to limit distribution to areas that someone thinks are relevant and keep the articles away from areas where the users (presumably) have no interest in them.

244 Essential Email Standards: RFCs and Protocols Made Practical

On the other hand, internal distributions can sometimes leak to the news feeds, causing internal corporate newsgroups to be propagated across the global Internet. This may not be so good, at least from the viewpoint of the corporation that does not want those newsgroups to be distributed externally as well as from the viewpoint of the people paying to globally distribute articles that are of limited interest.

The successor to RFC 1036 will provide a much more explicit mechanism for using the Distribution: header, probably by recommending that sites maintain lists of distributions to which they want to belong rather than allowing the current practice to continue.

Approved

The Approved: header field contains the email address and possibly also the full name of a newsgroup's moderator or of a newsgroup or news host administrator. This field is required of any article posted to a moderated newsgroup, as well as of any article containing a control message.

Control Messages

RFC 1036 describes the Control: header line as well as control messages in general. Any message that contains a Control: line is a control message. The audience for control messages consists of netnews host systems rather than people reading netnews, though control messages are passed from host to host in the same way as regular news articles. The Control: header field contains a control message to be interpreted by netnews hosts. Table 14.3 summarizes control messages defined in RFC 1036 as well as those referenced by the latest USEFOR draft. Though the USEFOR draft adds only one new control command, it does indicate several other commands that will likely be deemed obsolete in the update to RFC 1036.

Control messages are transferred in the same way as regular netnews articles, which means that they must be addressed and posted to a newsgroup using the preferred netnews transport protocol. These articles may be posted to an administrative newsgroup, with the form *.admin or *.announce (where the * represents the target newsgroup parent name). Having the Control: header identifies them to the receiving host as control messages to be interpreted as commands.

Control message commands ihave and sendme are posted to newsgroups that take the form to.hostname.domain. Thus, an ihave command sent to news.loshin.com would be addressed to the to.news.loshin.com newsgroup, and a sendme command sent to nntp.Internet-Standard.com would be addressed to the to.nntp.Internet-Standard.com newsgroup.

Table 14.3 Netnews Control Message Commands

COMMAND	SOURCE	STATUS AND DESCRIPTION
cancel	RFC 1036	Used to cancel earlier news articles. This command must include a message-ID to identify the article to be canceled. If the article is present on the receiving host, it is canceled (removed from distribution). Only the original sender or the local news administrator are permitted to issue cancel commands.
ihave	RFC 1036	Optional. The first command in the ihave/sendme protocol exchange. The command includes the term ihave, followed by a list of message-IDs and, optionally, the name of the sending system. Indicates that the sending host has received the articles indicated and that the host is willing to transmit those articles.
sendme	RFC 1036	Optional. The second command in the ihave/sendme exchange. The command includes the term sendme, followed by a list of message-IDs and, optionally, the name of the system sending the command message (the host that wants to receive the articles). When this command is received, the ihave host transmits the articles.
newgroup	RFC 1036	Allows creation of a new newsgroup. Command includes the term newgroup, a groupname, and an optional flag ("moderated") to indicate that the new group will be moderated. Requires an Approved: header line, otherwise it is ignored.
rmgroup	RFC 1036	Allows removal of a specified newsgroup. Command includes the term rmgroup and a newsgroup name. Requires an Approved: header line, otherwise it is ignored.
sendsys	RFC 1036	Obsolete. Requests that a copy of the sys file, containing a list of all the receiving host's neighbors to which that host sends netnews and the newsgroups each neighbor receives. A potential security threat, as it reveals information about network topology as well as relationships that could be exploited by an attacker.
version	RFC 1036	Obsolete. Requests the name and version of the software running on the host to which the command is sent. Again, a potential security threat as it reveals information about server software that could be exploited by an attacker.

Continues

246 Essential Email Standards: RFCs and Protocols Made Practical**Table 14.3** Netnews Control Message Commands (*Continued*)

COMMAND	SOURCE	STATUS AND DESCRIPTION
checkgroups	RFC 1036	This command is used along with a list of local "official" newsgroups with one-line descriptions in the body of the command article. The receiving host sends back, by email to the user "usenet" at that host, a list of any new newsgroups not on the list as well as a list of newsgroups that were on the list but are now obsolete.
mvgroup	USEFOR	A proposed command to replace the combination of rmgroup and newgroup when used to change a newsgroup's name.
whogets	"Son of 1036"	Obsolete. A proposed alternative to the command sendsys, used to request that a list of news feeds to other relayers be mailed to the article's reply address.
senduuname	RFC 850	Obsolete. Solicits the receiving host to run a program that lists all uucp neighbors of the local site.

Updating the Usenet Message Format

Significant modifications will be made to the standard for Usenet messages when the USEFOR workgroup has completed its task. Much of its work involves creating a formal and complete specification. As with any effort at updating an older Internet standard, clarifications are required for ambiguous language as well as generating the formal ABNF syntax for protocol entities such as messages, articles, commands, and responses. For example, Figure 14.3 shows a proposed ABNF syntax, adapted from current USEFOR work in progress, for Usenet messages.

Some of the changes that can be expected in the updated Usenet message format specification have already been referred to earlier in this chapter. This list includes some of the more likely areas in the specification that will see modification:

- Support for 8-bit data in netnews, and support of UTF-8 as the character set for headers.
- News article headers' support of some features allowed in the RFC 822/822bis specification, such as support for comments within headers and folding white space in headers.
- More clearly defined syntax for the Path: header to improve its usefulness.

```

article      = 1*header separator body
header       = header-name ":" SP header-content CRLF
header-name  = 1*name-character *( "-" 1*name-character )
name-character = ALPHA / DIGIT
header-content = usenet-header-content / unstructured
usenet-header-content
    = <a header-content specifically defined in this standard>
separator    = CRLF
body         = *( *998text CRLF )
nonblank-text = 1*( [FWS] nbtext )
nbtext       = qtext /           ; all of <text> except
              "\" / <">         ; SP and HTAB

              ; the next section shows support for 8-bit data

text         = %d1-9 /           ; all octets except
              %d11-12 /         ; US-ASCII NUL, CR and LF
              %d14-255
ctext        = NO-WS-CTL /       ; all of <text> except
              %d33-39 /         ; SP, HTAB, "(", ")"
              %d42-91 /         ; and "\"
              %d93-255
qtext        = NO-WS-CTL /       ; all of <text> except
              %d33 /           ; SP, HTAB, "\" and <">
              %d35-91 /
              %d93-255
ftext        = %d33-57 /         ; all octets except
              %d59-126 /        ; CTL, SP and ":"
              %d128-255
token        = 1*<any ftext except tspecials>
tspecials    = "(" / ")" / "<" / ">" / "@"
              "," / ";" / ":" / "
              "/" / "[" / "]" / "?" / "="

```

Figure 14.3 Proposed ABNF for updated Usenet message format.

- Formal acknowledgment of the importance of MIME to netnews, in particular for inclusion of application and multimedia content with articles.
- Addition of the control command mvgroup, for renaming newsgroups without deleting and recreating them.
- Addition of new article headers as well as other enhancement and specification clarifications.

Interested readers should check out the resources at the end of this chapter for more current information about the status of these specifications.

Network News Transfer Protocol

As with the Usenet message format, the Network News Transfer Protocol (NNTP) is undergoing revision: The specification needs to be updated, corrected, and clarified, widely used features need to be documented, and a mechanism for extending the protocol also needs to be specified. The revision effort is well along, but new RFCs are unlikely before 2000. This section focuses on NNTP as it is specified and deployed, though where appropriate, mention will be made of likely changes that can be expected in the specifications when they come out. The last part of this section summarizes the most likely areas in which NNTP will be changed when the new standards are approved and published.

NNTP Functions and Features

Much of the functionality of netnews is defined through the Usenet message format. The newsgroup hierarchies, the netnews commands, propagation of articles, article expiration, and other functions of netnews are byproducts of the message format. Articles must still be distributed across actual networks, which is the job for which NNTP was designed. NNTP defines how news articles are transmitted from host to host. This includes the transmission of articles from a server to a client as well as from a server to another server. In the Internet and most other IP networks, NNTP depends on the Transport Control Protocol (TCP) for reliable full-duplex virtual circuits between communicating hosts.

As with most of the other protocols discussed in this book, NNTP uses a stateful transaction protocol model. A host acting as a client initiates an NNTP connection with a host acting as a server by opening a TCP virtual circuit on the server's port 119. The server greets the client, netnews articles and data about articles can be exchanged, and then the session is terminated. The client sends commands, including parameters, to the server. The server responds with three-digit codes to indicate a success level as well as any requested data if the command was successful. When the client is finished transacting all the exchanges it wanted, it terminates the session.

Any client can specify a newsgroup, download new articles, upload new articles or relay existing articles to the server, query the server about specific articles or ranges of articles, and check on the existence of new articles in existing newsgroups. The client can select a newsgroup and move a virtual pointer to different articles within that newsgroup.

Though not specified in RFC 977, implementers have defined a number of extension commands, some of which are widely deployed. Many of these commands are useful for when the client is a news reading client, that is, used by a person to actually read news articles rather than a server acting as a client to retrieve or transmit news articles to another server. These commands

provide additional search function to the news reading client and are discussed later in this section.

NNTP States

RFC 977 is far from explicit about the existence of different states for server and client; neither is the most recent draft of the update to RFC 977—certainly not to the same degree that the SMTP and IMAP specifications are. However, NNTP servers and clients do cycle through various states of a connection whether they are explicitly documented in the specification or not. Three interactions are required for every NNTP session:

Connection. The connection occurs when the client opens the TCP virtual circuit to the server's port 119. It may occur only once during any given session.

Greeting. The greeting occurs when the server responds to the TCP open from the client. The greeting can indicate that the client is authorized to access articles through the server, is authorized to post new articles, is specifically not permitted any access to the server, or that the server is temporarily down or otherwise not available.

Disconnection. The session is disconnected after the client sends the quit command or when the connection times out or is simply dropped. It may occur only once during any given session.

Other types of interaction are not required but are more or less important if there is to be an exchange of news articles between hosts:

Capabilities Discovery. This is not part of the current NNTP specification, but it is an important part of the update to NNTP. At present, there is no formal protocol for determining what extensions a server supports, with server and client discovering each other's capabilities by issuing extension commands and interpreting the responses (the command is probably not supported when the server issues a "unknown command" or similar response).

News Exchange. In this phase, the client determines which articles it needs, and the server transmits them. The client may also indicate to the server that it has new articles, and they are transmitted to the server.

Conclusion. This is the initiation of a graceful exit by the client, occurring when the QUIT command is issued. It may occur only once during any given session.

The lack of explicit discussion of state in NNTP specifications does not mean it is not there. As we see in the next section, many NNTP commands must follow strict precedence, and response codes to most commands dictate what comes next.

NNTP Commands and Responses

NNTP was specified as a proposed standard early in 1986. Since then, it has been implemented on many different platforms and with many different “enhancements” and “extensions.” Some of these extensions have been recognized as “good things” and will likely be incorporated into the revised stan-

Table 14.4 NNTP Commands—Official and Unofficial

COMMAND	SOURCE	DESCRIPTION AND STATUS
ARTICLE	RFC 977	Display the header, a blank line, then the body (text) of an article specified by message ID or article number. When specified by article number, the internal current article pointer is set to that article.
BODY	RFC 977	Similar to ARTICLE command except returns only the text body of the article.
GROUP	RFC 977	Selects the named newsgroup; the server responds with estimated number of articles in group and article numbers of first and last articles.
HEAD	RFC 977	Similar to ARTICLE command except returns only the header lines of the article.
HELP	RFC 977	Returns brief summary of commands valid on the server.
IHAVE	RFC 977	Notifies the server that the client has an article available for upload. The server’s response indicates whether to send the article and header.
LAST	RFC 977	Sets article pointer to the previous article in current newsgroup; if already pointed to the first article, an error message is returned and the pointer remains selected.
LIST	RFC 977	Returns valid newsgroups with first/last article numbers posting status.
NEWGROUPS	RFC 977	Returns all newsgroups created since specified date/time in LIST format.
NEWNEWS	RFC 977	Returns a list of message-ids of articles posted or received to the specified newsgroup since the specified date.
NEXT	RFC 977	Moves the article pointer to the next article in the current newsgroup. If no more articles remain in the current group, an error message is returned and the current article remains selected.
POST	RFC 977	A two-step command, a 340 response code indicates the article should be posted.QUIT

Table 14.4 (Continued)

COMMAND	SOURCE	DESCRIPTION AND STATUS
QUIT	RFC 977	Client request that the server terminate the session and close the connection.
SLAVE	RFC 977	Obsolete. Meant to distinguish news readers from other news servers, but ambiguously defined and variously implemented.
STAT	RFC 977	Similar to ARTICLE command except no text is returned; when selecting by message number within a group, sets the current article pointer without sending text.
MODE STREAM	nntptext	Requests that the server go into STREAM mode, allowing the client to send commands in streams without waiting for a response before sending another command. Must be used prior to CHECK or TAKETHIS.
CHECK	nntptext	Issued with a message ID. This command asks whether the article should be sent using the TAKETHIS command when in streaming mode.
TAKETHIS	nntptext	Identifies an article by message ID and indicates that article will be sent immediately following the TAKETHIS command is sent.
LIST <ext>	nntptext	The original LIST command took no arguments, but since 1987, implementers have been adding functionality by adding arguments to this command.
LISTGROUP	nntptext	Lists all article numbers in a specified newsgroup.
MODE READER	nntptext	Requests the server go into READER mode, allowing the server to optimize itself for news client response.
XHDR	nntptext	Allows retrieval of specific headers from specific articles.
XOVER	nntptext	Requests information from an overview database about the article or articles specified in the argument.
XPAT	nntptext	Used to retrieve specific headers from specific articles containing headers that match a pattern included as an argument to the command.
XPATH	nntptext	Requests the path and filename of the file containing the specified article. Not recommended for use by client implementations, this command requires knowledge of the server's file system to be useful.

Continues

252 Essential Email Standards: RFCs and Protocols Made Practical**Table 14.4** NNTP Commands—Official and Unofficial (*Continued*)

COMMAND	SOURCE	DESCRIPTION AND STATUS
XROVER	nntptext	Requests reference information from the overview database about the article or articles specified in the argument.
AUTHINFO	nntptext	Usually implemented as a reply/response command pair, the server requests user ID and password from client; exchanged in plaintext.

dard; others are deemed historic or obsolete. Both these extended commands and the commands specified in RFC 977 are listed in Table 14.4. The NNTPEXT workgroup is working on a document to eventually be published as an RFC that lists all existing extensions and indicates which are widely deployed and used and which are obsolete (or should be). In Table 14.4, these commands are

Table 14.5 NNTP Response Codes

CODE	MEANING	RESPONSE TO COMMAND
100	Help text	ANY
190 through 199	Debug output	ANY
200	Server ready, posting allowed	ANY
201	Server ready, no posting allowed	ANY
400	Service discontinued; or, not accepting articles	ANY
480	Transfer permission denied	transfer commands
500	Command not recognized	ANY
501	Command syntaxerror	ANY
502	Access restriction or permission denied	ANY
503	Program fault, command not performed	ANY
220 n <a>	Article retrieved, head and body follow (n = article number, <a> = message-id)	ARTICLE
221 n <a>	Article retrieved, head follows	ARTICLE
222 n <a>	Article retrieved, body follows	ARTICLE
223 n <a>	Article retrieved, request text separately	ARTICLE

Table 14.5 (Continued)

CODE	MEANING	RESPONSE TO COMMAND
412	No newsgroup has been selected	ARTICLE
420	No current article has been selected	ARTICLE
423	No such article number in this group	ARTICLE
430	No such article found	ARTICLE
211 n f l s	Group selected (n = estimated number of articles in group, f = first article number in the group, l = last article number in the group, s = name of the group)	GROUP
411	No such news group	GROUP
100	Help text follows	HELP
235	Article transferred ok	IHAVE
335	Send article to be transferred	IHAVE
435	Article not wanted, do not send it	IHAVE
436	Transfer failed, try again later	IHAVE
437	Article rejected, do not try again	IHAVE
223 n a	Article retrieved, request text separately (n = article number, a = unique article id)	LAST
412	No newsgroup selected	LAST
420	No current article has been selected	LAST
422	No previous article in this group	LAST
215	List of newsgroups follows	LIST
231	List of new newsgroups follows	NEWGROUPS
230	List of new articles by message-id follows	NEWNEWS
223 n a	Article retrieved, request text separately (n = article number r, a = unique article id)	NEXT

Continues

254 Essential Email Standards: RFCs and Protocols Made Practical**Table 14.5** NNTP Response Codes (*Continued*)

CODE	MEANING	RESPONSE TO COMMAND
412	No newsgroup selected	NEXT
420	No current article has been selected	NEXT
421	No next article in this group	NEXT
240	Article posted OK	POST
340	Send article to be posted. End with <CR-LF>.<CR-LF>	POST
440	Posting not allowed	POST
441	Posting failed	POST
205	Closing connection, goodbye!	QUIT

listed as coming from NNTPEXT; information in this table is taken from RFC 977 as well as from NNTPEXT work in progress.

A close reading of the commands and response codes, listed in Table 14.5, reveals the bulk of the protocol. For example, the IHAVE command is used by a client (in this case, an NNTP server acting as a client) to notify the server that the client has an article for upload. The client may have received the article from another server and wants to offer it to its feed server.

In a typical IHAVE exchange, the client issues the IHAVE command along with an article message ID (uniquely identifying the article throughout the net-news system). If the server does not already have that article, it responds with the reply code 335, indicating “send article to be transferred.” Note that the reply codes are structured similar to most application protocol response codes; the first digit of 3 indicates that the command is successful but not yet complete.

Upon receiving the 335 code reply, the client can begin sending the article headers and body to the server. When the terminating line (a CRLF followed by a period followed by CRLF) is received, the server can send a 235 reply code, indicating “article transferred ok.” Alternatively, if the server already had the article and did not want the client to send it, the server would send a 435 code, meaning “article not wanted, do not send it.”

News Reader Client Commands

The RFC 977 commands are very general and can be used for both types of NNTP clients: the news reader client being used by a person who wants to read news articles as well as the news server getting new articles and relaying its own new articles. However, over the years implementers have added extensions to the LIST command that add to its functionality for news readers. Table 14.6 lists the common extensions to the LIST command. These are not the only

Table 14.6 Extensions to the NNTP LIST Command

COMMAND	DESCRIPTION
LIST ACTIVE [wildmat]	Lists all active newsgroups when command is issued with no parameter; when a parameter is present, lists only the active newsgroups that match that pattern.
LIST ACTIVE.TIMES	Some servers maintain information about the date/time newsgroups were initiated on the host; this command lists all newsgroups. Each listing includes newsgroup name, time created (measured as the number of seconds since January 1, 1970 at the time of creation), and an email address of the entity responsible for creating the newsgroup.
LIST DISTRIBUTIONS	Some servers maintain a distributions file containing valid values for the Distribution: header and definitions of those values. This command returns that information.
LIST DISTRIB.PATS	Some servers maintain a file (distrib.pat) that contains default values for the Distribution: header field for particular newsgroups. This command returns that information.
LIST NEWSGROUPS [wildmat]	Some servers maintain a file (newsgroups) containing the name and a short description of each active newsgroup on the server. This command returns that information. The wildmat parameter returns only those newsgroups matching the specified pattern.
LIST OVERVIEW.FMT	Some servers maintain a file (overview.fmt) that specifies the order in which header information is stored in the overview databases for each newsgroup. This command returns the header fields in the order in which they are stored in the database.
LIST SUBSCRIPTIONS	Returns a default subscription list for new users of the server receiving the command.

additions to the protocol that help the user, but because they extend an existing command rather than add a new command, they are more protocol friendly.

WILDMAT Matching in NNTP

For many years, NNTP servers have supported a pattern matching format called WILDMAT. Based on the format for pattern matching used in the Unix find command, WILDMAT was first developed by Rich Salz and implemented in the INN news server. WILDMAT is likely to be incorporated into the revised specification for NNTP, to be used with the LIST command. Simply listing all

256 Essential Email Standards: RFCs and Protocols Made Practical**Table 14.7** WILDMAT Pattern-Matching Operations

OPERATOR	DESCRIPTION	EXAMPLES
*	Matches any sequence of zero or more UTF-8 characters	alt.losin.* returns all newsgroups under the alt.losin hierarchy
?	Matches any single UTF-8 character	alt.losin.??? returns all direct children in the alt.losin hierarchy with three-characters in the last element of the newsgroup name
[abc] or [a-c]	Matches specific characters or ranges of characters (specific characters and ranges may be combined)	alt.losin.[a-o]rangutan returns any existing newsgroups that look like alt.losin.arangutan, alt.losin.brangutan, and so on
[^abc] or [^a-c]	Matches any character other than those specified in the list or range	alt.losin.[o]rangutan returns any newsgroup that matches alt.losin.?rangutan EXCEPT for alt.losin.organgutan
!x	Excludes x, where x is any other valid expression using	!alt.losin.* excludes all newsgroups the operators listed above in the alt.losin hierarchy
\x	Interprets x as a regular character; x can be one of [, !, *, ?, or \.	alt.losin.*star returns the newsgroup alt.losin.*star (the asterisk is part of the newsgroup name)

newsgroups on a server can take a considerable amount of time considering that there are tens of thousands of newsgroups available; using some mechanism to limit the number of newsgroups listed can be extremely helpful.

WILDMAT defines six pattern-matching operations, listed in Table 14.7 along with examples and explanations.

Extending and Updating NNTP

The NNTPEXT workgroup of the IETF is spearheading a two-pronged effort. First, it is taking an Internet protocol specification published in 1986 and updating it by making corrections, clarifications, and formalizing its content. Second, it is adding new features and functionality to a protocol that has served the Internet well for many years.

Foremost among the new features and functions will be the addition of a protocol extension mechanism and explicit support for UTF-8 characters in newsgroup names. New commands as well as command extensions (such as the

extensions to the LIST command) are likely to wind up in the revised specification. More important, a mechanism by which hosts can do extension discovery will also be incorporated into the specification. This mechanism will likely resemble those incorporated into SMTP and other protocols discussed in this book.

Reading List

Table 14.8 lists RFCs relevant to the contents of this chapter. As NNTP and the Usenet message format standards are still undergoing significant updates, the reader is directed to the IETF workgroup Web pages for more information about the status of these specifications:

<http://www.ietf.org/html.charters/nntpext-charter.html>
<http://www.ietf.org/html.charters/usefor-charter.html>

Table 14.8 Suggested RFCs for Further Study

RFC	TITLE
RFC 822bis	Internet Message Format Standard
RFC 977	Network News Transfer Protocol
RFC 977bis	Network News Transfer Protocol
RFC 1036	Standard for Interchange of USENET Messages
RFC 1036bis	News Article Format
RFC 2076	Common Internet Message Headers

