HowTo NNTP

by Randy Charles Morin

Some of the simplest, yet very rich communication protocols were born on the Internet. This is the third article in a series where I will write on these simple communication protocols. This article will focus on the NNTP (Network News Transport Protocol) protocol. This protocol is the most often used protocol for sending and receiving USENET news over the Internet.

In our two previous articles we wrote a class named Socket that we will reuse. This class provides all the basic socket level code that we require to connect, send, receive and disconnect from NetNews servers. In the previous articles, I created two classes named Smtp and Pop3 that inherited from this common Socket class. In this article, I will create a new classed named Nntp that will also inherit from the Socket class.

The NNTP protocol is again very similar to the SMTP and POP3 protocols that I introduced in previous articles. The protocol works by establishing a connection with server, presenting commands in the form of verbs and attributes and reading status and data responses. I'll limit my discussion of NNTP to the following verbs; QUIT, LIST, GROUP, ARTICLE and POST. A complete NNTP client would normally explore a larger range of NNTP verbs in its implementation.

Our Nntp class declaration is presented below.

```
class Nntp : public Socket
{
public:
        Nntp(const std::string & strServer);
        virtual ~Nntp();

        std::vector<std::string> GetNewsgroups();
        std::vector<std::string> GetNews(const std::string & newsgroup);
        void Post(const std::string & newsgroup,
                const std::string & subject,
                const std::string & from,
                const std::string & content);
};
```

The TCP/IP connection is implemented in the constructor of our class. The disconnect and QUIT verbs are implemented in the destructor of our class. The LIST, GROUP, ARTICLE and POST verbs are implemented in the methods in our Nntp class. The GetNewsgroups method implements the LIST verb. The GetNews method implements the GROUP and ARTICLE verbs. The Post method implements the POST verb.

The constructor and connection algorithm for our NNTP class requires one pieces of information, the remote POP3 server name. When I establish the NNTP connection, the server will respond with a status message. A positive response will begin with the 200 status code, otherwise we raise an exception to indicate that the connection failed.

```
inline Nntp::Nntp(const std::string & strServer)
{
        Connect(strServer, "nntp");

        std::string response = Response();
        int status;
        std::stringstream ss;
        ss << response;
```

Copyright 2001-2002 Randy Charles Morin

```
        ss >> status;
        if (status != 200)
        {
                throw SocketException(response);
        }

}
```

The first step after establishing a new connection with a server is to download the list of available newsgroups. I connected to the news.microsoft.com news server as it is public and the list of available newsgroups is small enough that downloading them requires only a few minutes. Some servers have enough newsgroups that the initial download of newsgroups can take a very long time.

You can download the complete list of newsgroups using the LIST verb. The server will respond with one status line having the status code 215. Any other status code would indicate an incorrect response and an exception raised. Following the status line, the server send one line for each newsgroup on the server. Each line is a space delimited record with the name of the newsgroup, the last message id, the oldest message id and an flag indicating whether posting to the newsgroup is allowed. Finally, after all the records are sent, a terminating line is sent. The termination line has but one character, a period, following by a newline.

```
inline std::vector Nntp::GetNewsgroups()
{
        {
                std::stringstream ss;
                ss << "LIST\r\n";
                Write(ss.str());
        }

        std::string response = Response();
        int status;
        std::stringstream ss;
        ss << response;
        ss >> status;
        if (status != 215)
        {
                throw SocketException(response);
        }

        std::vector retval;
        while(true)
        {
                response = Response();
                if (response == ".\r\n")
                {
                        break;
                }

                if (response == ".\n")
                {
                        break;
                }

                std::string name;
                name = response.substr(0, response.find(" "));
                retval.push_back(name);
        }

        return retval;
}
```

Copyright 2001-2002 Randy Charles Morin

Next you will want to download the news items contained within a newsgroup. I'll use the GROUP and ARTICLE verbs to indicate the newsgroup I want to download and the articles I want to download.

The GROUP verb takes one attribute, the name of the newsgroup. The server will respond to the GROUP verb with a status line containing a positive status code of 211, the number of articles on the server, the oldest article id on the server and the last article id on the server.

You can now use the ARTICLE verb to download one article at a time use the article range returned from the GROUP verb response. The ARTICLE verb takes one parameter, the article id. The response from the server will be one or more lines. If the server responds with a status code of 423, then the article id does not exist. If the server responds with a status code of 220, then the article exist and each line of the article is sent. The last line of the article is following by the termination line, a period followed by a newline.

```
inline std::vector Nntp::GetNews(const std::string & newsgroup)
{
	{
		std::stringstream ss;
		ss << "GROUP " << newsgroup << "\r\n";
		Write(ss.str());
	}

	std::string response = Response();

	std::stringstream ss;
	ss << response;
	int status, articles, start, end;
	ss >> status;
	if (status != 211)
	{
		throw SocketException(response);
	}

	ss >> articles >> start >> end;

	if (start+100 < end && end > 100)
	{
		start = end-100;
	}

	std::vector retval;
	for(int i=start;i<=end;i++)
	{
		std::stringstream ss;
		ss << "ARTICLE " << i << "\r\n";
		Write(ss.str());

		response = Response();
		int status;

		{
			std::stringstream ss;
			ss << response;
			ss >> status;
		}

		if (status == 423)
		{
			continue;
		}

		if (status != 220)
```

Copyright 2001-2002 Randy Charles Morin

```
                {
                        throw SocketException(response);
                }

                std::stringstream article;
                while (true)
                {
                        response = Response();
                        if (response == ".\r\n")
                        {
                                break;
                        }

                        if (response == ".\n")
                        {
                                break;
                        }

                        if (article.str().length() < 1024)
                        {
                                article << response;
                        };

                }
                retval.push_back(article.str());
        }

        return retval;
};
```

You'll also want to post new articles to newsgroups. This is done using the POST verb. The POST verb does not take any attributes. The server will respond to the POST verb with a 340 status code, indicating that it is ready to receive the header and body of your message.

The minimal header will contain the news item subject, sender and newsgroup name. These are specified in single line records with a key value pair on each line. The key and value are separated by a semi colon and space. An example header is shown. The sender must be a valid email address and the newsgroup a valid newsgroup or list of newsgroups.

```
From: me@kbcafe.com
Newsgroups: microsoft.test
Subject: My test subject
```

The headers are separated from the body by one blank line. After the first blank line, the remainder of the message is the message body. Finally, after the message body, a single termination line indicates to the server that the message is complete. The server will respond to the header and body with a status line. The status line will have status code 240 to indicate the message was posted correctly. The format of the message follows the USENET news format as described in RFC 850.

```
inline void Nntp::Post(const std::string & newsgroup,
                const std::string & subject,
                const std::string & from,
                const std::string & content)
{
        {
                std::stringstream ss;
                ss << "POST\r\n";
                Write(ss.str());
        }

        std::string response = Response();
```

Copyright 2001-2002 Randy Charles Morin

```
        int status;

        {
                std::stringstream ss;
                ss << response;
                ss >> status;
                if (status != 340)
                {
                        throw SocketException(response);
                }
        }

        {
                std::stringstream ss;
                ss << "From: " << from << "\r\n"
                        << "Newsgroups: " << newsgroup << "\r\n"
                        << "Subject: " << subject << "\r\n\r\n"
                        << content << "\r\n.\r\n";
                Write(ss.str());
        }

        response = Response();
        {
                std::stringstream ss;
                ss << response;
                ss >> status;
                if (status != 240)
                {
                        throw SocketException(response);
                }
        }

};
```

Finally, once you've completed retrieving newsgroups, news and posting new news, you'll want to disconnect from the server. You should send a QUIT verb command to the server before terminating the socket connection.

```
inline Nntp::~Nntp()
{
        {
                std::stringstream ss;
                ss << "QUIT\r\n";
                Write(ss.str());
        }

        std::string response = Response();
        int status;
        std::stringstream ss;
        ss << response;
        ss >> status;
        if (status != 205)
        {
                throw SocketException(response);
        }

};
```

I tested this code using the following test console application. It was tested using Borland C++ Builder 5. I have not tested it with Visual C++ yet, but I don't see why it would not work. The previous Smtp, Pop3 and Socket classes were all developed in Visual C++.

```
#include
#include "nntp.h"

int main(int argc, char* argv[])
{
        try
        {
```

Copyright 2001-2002 Randy Charles Morin

```
                kbcafe::Nntp nntp("news.microsoft.com");
                std::vector list;
                list = nntp.GetNewsgroups();
                for (std::vector::iterator i = list.begin();
                        i != list.end(); i++)
                {
                        std::cout << *i << std::endl;
                }

                nntp.Post("microsoft.test", "Test Subject", "rmorin@kbcafe.com",
                       "Test Body\nEnd...");

                std::vector news = nntp.GetNews("microsoft.test");
                for (std::vector::iterator i = news.begin();
                        i != news.end(); i++)
                {
                        std::cout << *i << std::endl;
                }
        }
        catch(kbcafe::SocketException & e)
        {
                std::cerr << e.what();
        }
        return 0;
}
```

You'll see in my examples that I use the "microsoft.test" newsgroup for all my testing. It is very important to limit your testing to this and similar newsgroups, like "alt.test". If you test on other public newsgroups, you'll find you will develop some enemies who do not take kind to uncontrolled testing of NNTP servers. A benefit of testing using the "microsoft.test" and similar newsgroups are that servers are setup to automatically respond to each post in order to help the developer validate that his posting is being read on servers all over the world.

The next article in this series, I will introduce you to the FTP protocol. This is the protocol that has been used for years to transmit files over the Internet.

For a more complete understanding of POP3, I suggest you consult the RFCs (Requests For Comments). RFCs are Internet standards that have been adopted by the IETF (Internet Engineering Task Force). The RFC for NNTP is RFC 977 and can be found at the following URL

http://www.ietf.org/rfc/rfc0977.txt?number=977

You should also review the RFC for the USENET news message format as presented in RFC 850.

http://www.ietf.org/rfc/rfc0850.txt?number=850

I found it very helpful to read through the scenarios provided in the RFC. They give a better indication of the complete functionality provided for by this mail protocol.

Copyright 2001-2002 Randy Charles Morin