A proposed technique for tracing origin of spam on the Usenet

by

Dirk Bertels, BComp

A dissertation submitted to the School of Computing
in partial fulfillment of the requirements for the degree of

Bachelor of Computing with Honours

University of Tasmania

June 2006

This thesis contains no material which has been accepted for the award of any other degree or diploma in any tertiary institution. To the candidate's knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

Signed

Dirk Bertels
Hobart, June 2006

Abstract

The Usenet, a *worldwide distributed decentralized conferencing system*, is widely targeted by spammers who use a variety of techniques in order to obscure their identity. One of these techniques is called *path preload*, in which the path header is spoofed by means of attaching a false section at the beginning of this path.
The process of detecting and confirming path preload is laborious and requires a thorough understanding of the Usenet. A technique which downloads a particular article from several servers, and compares their path headers is explored as to its usefulness regarding path preload detection.

This document begins with a general background on the Usenet, highlighting those aspects that are relevant to the research, especially the topics of Usenet headers and spam. This leads to a description of the proposed technique and the development of a tool capable of implementing this technique.

The tool essentially downloads a spam article from different servers, and analyses their headers. A map is constructed from the data gathered showing the servers that the articles traverse in order to reach their destiny. Since each article is downloaded from more than one location, some commonality may be found in these trajectories. If this commonality occurs at the beginning of the trajectory, the article is said to have a *common path*.

Once a common path is established, a more heuristic approach is taken in order to establish some preliminary conclusions as to whether this common path is likely to be path preload or not. This approach requires some knowledge about various aspects of the Usenet and involves additional anti-spam techniques alongside the common path technique.

Several sessions have been conducted and the results outlined, analysed, and discussed at the end of this document, followed by some thoughts on possible future advances and further improvements.

Acknowledgements

The author likes to thank the following people for their contribution:

Prof. Christopher Lueg	Supervisor
Dr. Vishv Malhotra	Co-supervisor

Index

1. Introduction

Having observed a variety of research presentations, one thing stands clear; all research should have some human aspect to it. This holds very much true in this research; the Usenet is first and foremost a human interaction tool and all its uses and misuses serve this purpose. The best way to get acquainted with the Usenet is to connect to it, read some articles on a particular topic, respond to a thread that interests you, and read the replies to it.

In order to get a more technical understanding on the topic, it is necessary to know how the articles propagate through the network; i.e. the protocol used and some networking issues. Then, in order to get a grasp on this particular line of research you need to look at how articles are constructed, especially the headers; which ones are mandatory, which ones are easy to forge, etc.

But first, what is the Usenet?

Usenet, also known as *Network News*, is a distributed electronic *bulletin board* mechanism that functions on a global scale. It came into existence in 1979 and has been the foremost online community ever since.
The great functional strength of the Usenet lies in the fact that posted messages can be accessed and replied to (in the form of new posted messages) by anyone. This is due to a large network of interconnected *news servers*. However, due to the free nature and wide spread reach of the Usenet, a large amount of Spam can be generated on the network.

An analysis of the Usenet messages, together with various accumulated data, such as data determined by retracing the various paths a message takes should reveal some mechanisms that enable one to determine where the article originated. Indeed the aim of this thesis is to explore a technique that is based on this idea.

In order to investigate this technique, which will be outlined soon, a tool has been build which effectively enables one to analyse the headers of the same spam article, downloaded from different locations (news servers). The *path* header is what interests us most. It lists the identities (address spaces) of the different servers the article passes through in its propagation to a receiver.

In the knowledge that the header of an article contains a sequenced list of all the servers that the article passes, it is easy for a person to trace back through this list and identify the server on which the article was posted. In the case where the article is spam, a conscientious person can notify the administrator of this server and make a complaint. The administrator in turn can *cancel* the article, effectively deleting it from the Usenet. He can also reprimand the spammer or, worse, deny the spammer further access to his server.

Clearly, a spammer will be tempted to alter this path in such a way that any effort to trace back to him will give false readings. One of the techniques used by spammers to hide their identity is called *path preloading*. This technique consists of appending fake server address spaces to the beginning of the list. This project will exploit this fact. Indeed, the technique proposed to me by one of my supervisors, Professor Christopher Lueg, states that:

It is possible to gather headers of the same spam article posted to different news servers. If the first few addresses in the path header of the suspect spam article are the same in all of these versions, then it could be a preload section of the path.

The word *could* is important here, because it may be possible that there are valid reasons for the path entries being the same. If no such reasons are forthcoming, the first server adjacent to the preload section of the path should identify the server the spammer originally posted to.

In a more practical sense, we are investigating two aspects:

1. Determining a method that enables mapping the paths that an article travels in its diffusion through the Usenet. This mapping facilitates the detection of a common path (possible preload). This stage of the research involves investigating the possible platforms to develop the tool on, the programming languages to use, the structure of the program, etc. Generally, the practical programming aspect.
2. Investigating factors that may lead one to conclude to a degree of certainty whether a common path, mapped by the tool, is in fact a result of path preload.

In short, the first part covers detecting and mapping the *common path*, while the second part handles the determination of *path preload*. These two aspects will be elaborated upon in this thesis when enough background has been covered. It is worth noting that the issue is not to determine whether an article is theoretically spam, but whether it contains a preload in its path header; though it can be argued that the presence of path preload almost certainly indicates that the article is spam (almost, because sometimes people use preload just to prove that it can be done).

What has been done already to detect path preload? Not a lot, there is much talk about preload on the Usenet, but to my knowledge, no preload analysing tools have been developed so far, for good reasons as will be concluded at the end of this document. This research is a first attempt to do this.

2. The Usenet – an overview

The Usenet can be defined in many ways: As a *global distributed conferencing system* (Emerson 1983), as a *conversational social cyberspace* (Turner, Smith, Fisher and Welser 2005), or more technically, as a *set of protocols for exchanging messages within a decentralized set of news servers* (Hewlett-Packard 2002).

The other great mail networking system, email, has two important limitations as far as group communication is concerned: Each message must be explicitly addressed to a recipient, and each recipient needs a separate copy of the message.

The Usenet on the other hand, is a network that consists of thousands of nodes (servers) dispersed worldwide and loosely connected. Each of these nodes holds a chosen range of topics (newsgroups). When users (clients) connect to any of these servers they may either read a message or post a message on any of the topics that the server holds. Once posted, the 'posted to' server feeds the message to its near servers (called *peers*) who in turn feed it to *their* peers, and so on. This means that virtually anyone connected to the Usenet can read your message.

2.1. Origin of the Usenet

The Usenet has its origins in 1979. Two people are accredited with its invention, Tom Truscott and Jim Ellis. Some accounts claim that Tom Truscott started the concept of the Usenet because he missed (emotionally) the UNIX system, in particular the UUCP protocol used to communicate between 2 UNIX machines, after spending some time at Bell laboratories (the birth place of Unix) (Lueg and Fisher 2003).
Another account claims that the birth of the Usenet was due to the 2 graduate students, Truscott and Ellis, devising the distributed news system after an upgrade of the operating system caused the existing bulletin board software to fail (page 15) (Ingvoldstad 2001).

Maybe the combination of both these issues inspired the new graduates to use this new promising operating system, UNIX, to build a new type of bulletin board system. Whatever the cause, at the time the Usenet came into existence many people thought it was just another bulletin board system, as bulletin boards were very popular then. However these were early days and the full grasp of Usenet wasn't yet realised, in fact, Truscott and Ellis envisioned a system of many ‘Usenets’.

To put things in a historical context (assuming 25 years can be regarded as history), Usenet predates the WWW, which started in 1990, by a decade. The Internet has been in existence since 1960, but the TCP/IP protocol, widely used on the Internet now, came into being around 1983. Because of this, the first years of the Usenet used the UUCP transport mechanism.

The word *Usenet* comes from *USENIX Association,* the professional and technical UNIX users group. The name UNIX is a pun on MULTICS (the major pre-UNIX operating system). The pun is that where in some areas MULTICS tried to do many things at once, UNIX tried to do one thing well. (Quarterman and Hoskins 1986).

2.2. Current state of the Usenet

Many people regard the Usenet as being somewhat outdated. Even more people have never heard of the Usenet. But figures regarding the current state of the Usenet are surprising. For example (Sit, Dabek and Robertson 2004) reveal that the size of Usenet postings per day doubles every 10 months. In 2004, users created 1.4 TB of Usenet data (about 4 million articles) per day. Note however that the growth is largely due to increased postings of binaries (estimated at 2 TB), while the volume of text postings have remained largely stable. It was suggested by Prof. Lueg that although the size of 1.4 TB may occur on particular high traffic days, 1 to 1.2 TB seems like a more realistic estimation.

This sudden explosion in binary files is curious since relatively few people know of the Usenet. (Bray 2005) blames it on piracy; although you can't directly post binary files on the Usenet, you can use special software to encode these binary files into a compatible format. He also states that in 1996, a day's worth of Usenet postings amassed to around 4.5 GB (this means that throughput has grown 250 times in 8 years).

Bray cites Microsoft as the big contributor to the Usenet, sponsoring about 2,800 professionals (MVP's, Most Valuable Professionals). A MVP may spend several hours a day on the Usenet answering questions. Some of these resources are now being shifted to Blogs and Web boards. But the Usenet postings have the advantage that all postings are preserved (unlike postings on Blogs or Web boards). The search engine, Google, maintains a database of all the Usenet postings (past and present) so that all the advice is preserved.

2.3. Usenet Addressing and Newsgroups.

Newsgroups are the addressing mechanism of the Usenet. An article can only be posted to a particular newsgroup. Each newsgroup represents a particular area of interest. A newsgroup does not reside at a particular location, it is a hierarchy that exists amongst all the news servers, a topic classification system that all news servers agree to. Any news server holds a certain range of newsgroups that people can access or post to (if permitted to do so).

There were originally about 143 widely distributed newsgroups, 8 of them forming the historical hierarchies of the Usenet (referred to as the *Big 8*):

```
comp, misc, news, rec, sci, soc, talk, hum
```

Examples of newsgroups are

```
comp.lang.perl.misc
alt.support.diabetes.kids
misc.kids.pregnancy
```

Today the volume of newsgroups has grown in the thousands; this is largely due to the fact that anyone can start a new newsgroup, even a poster.

A user connecting to a news server may post a message to this server only if this message is addressed to one or more existing newsgroups. Assuming that the server carries the newsgroup(s), the poster has the option to post one article to more than one newsgroup, a technique called *crossposting*. The following chart from Marc Smith clarifies this.

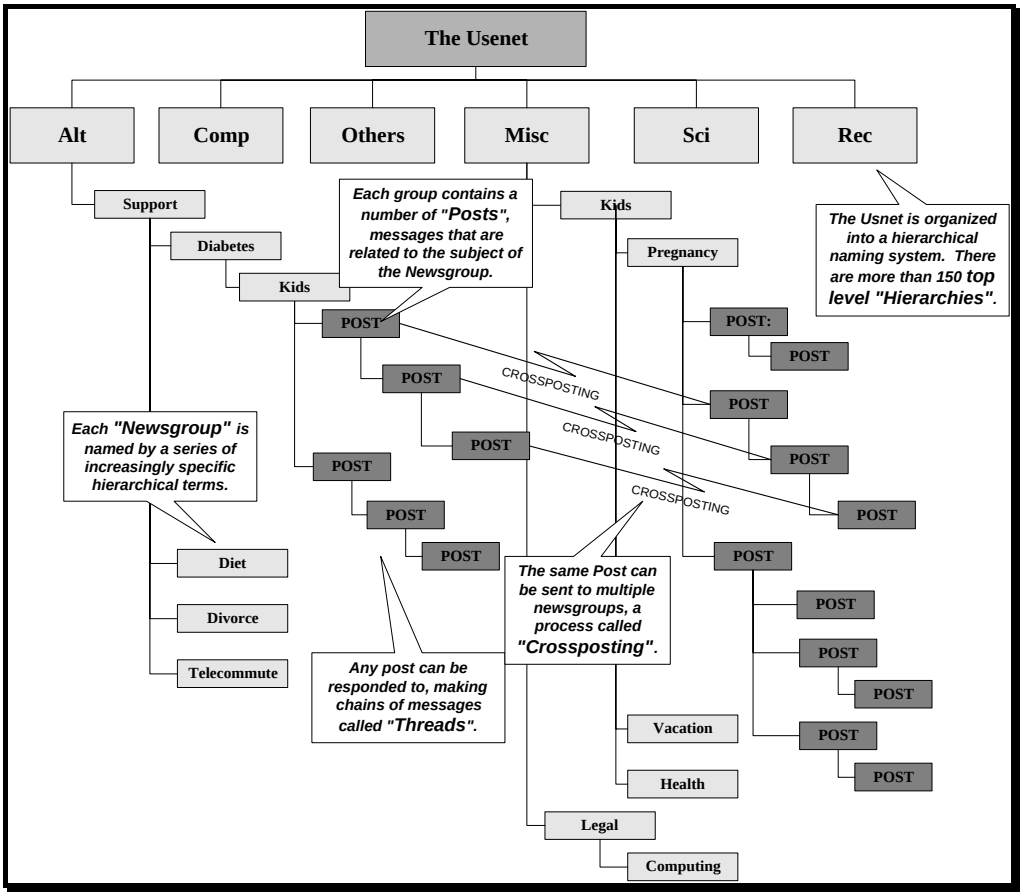


Figure 1: Schematic representation of the Usenet newsgroups (Smith 1999)

Crossposting

As stated above, *crossposting* is simply the act of posting a message to more than one newsgroup. Note that crossposting is essentially an efficient mechanism because

it avoids sending copies of the same message to different newsgroups, which would greatly overload the Usenet.

In one third of all newsgroups, 100% of the messages are crossposted to (or from) other groups. Using crossposting records it is possible to generate hierarchical maps showing the relationships between Usenet newsgroups (Smith 1999).

2.4. The Usenet Networking System

The UUCP network

Just as the concept of the Usenet has grown, so has the technology. In the early days of the Internet, the UUCP transport mechanism was used. UUCP is a transport service between adjacent systems. The initials stand for **U**NIX to **U**NIX **C**o**p**y. In UUCP there is no clear division between the transport layer and the network layer and there is also no resemblance of an internet protocol (Quarterman and Hoskins 1986), after all as previously stated, the TCP/IP protocol used today was introduced in 1983, after the Usenet came into being.

This means that there was no DNS (Domain Name Server), as is used on today's Internet to resolve IP addresses. Instead, to enable other hosts finding your host to communicate with, you needed to be registered in the *UUCP map*. This is a registry that is kept by volunteers known as the UUCP project (Summers-Horton 1985). The map is posted monthly in the newsgroup *comp.mail.maps*. Interestingly a program called *pathalias* can compute reasonable routes from the UUCP maps.

While NNTP (the Usenet protocol) today provides direct end-to-end mail delivery, UUCP uses *store and forward* protocols which moves mail towards its destination one hop at a time (Hunt 2002).

NNTP - The News Network Transport Protocol

With the advent of the TCP/IP Internet protocols, the transfer of Usenet articles can be handled using TCP.
Usenet now needed an application-level protocol to cover Usenet traffic over TCP/IP. The result is the NNTP, similar to the SMPT protocol which in turn borrowed from Telnet and FTP. NNTP was published as RFC977 in February 1986. NNTP handles 2 situations: The propagation of news articles between servers, and client article posting and access.

2.5. Request For Comments (RFC) standards

The two main protocols guiding the Usenet are the RFC 2822, which sets a standard for the interchange of Usenet messages, and the RFC 977 setting the standards for the NNTP.

RFC2822: Standard for Interchange of Usenet messages.

The RFC2822 standard (Lindsey 2006) updates and replaces RFC 1036 (Horton and Adams 1987), which in turn replaces RFC850. The RFC2822 standard is a recent update, from July 2005. RFC 1036 describes the message format and gives some standards regarding the transmission of news. The document handles four topics: The message format, valid control messages, valid transmission methods, and the overall news propagation algorithm.
Note that this protocol does not discuss message transport issues (Saiedian and Winslade 1992). It lists the mandatory headers as:

```
From, Date, Newsgroups, Subject, Message-ID, and Path.
```

While some optional headers are:

```
Followup-To, Expires, Reply-To, Sender, References,
Control, Distribution, Keywords, Summary, Approved,
Lines, Xref, and Organization.
```

The headers relevant to this project will be discussed in the *Header* section.

RFC 977: Network News Transfer Protocol

This is the Proposed Standard for the Stream-Based Transmission of News, using NNTP (Kantor and Lapsley 1986).

In its introduction, the document explains how this protocol came about to remedy the problems associated with distributing Internet news using mailing lists, where information is copied and send to each subscriber on the mailing list. This quickly becomes inefficient once the mailing list becomes large.
The Usenet news system avoids this problem by using central storage servers (news servers) that propagate the articles. Clients can then access any of these servers and download the required articles. Unlike SMPT, which only functions between servers, NNTP also allows server-client interaction. This enables the client to post articles to a news server using the same protocol.

Other topics handled by this protocol are news distribution, NNTP specifications, and the various commands used (*Article, Body, Head, Stat, Group, Ihave, Last, List, Newsgroups, Newnews, Next, Post, Quit, Slave*).

In closing, the article shows some sample conversations. For example, to post a new article, the conversation may go as follows:

```
server:   (listens at TCP port 119)
client:   (requests connection on TCP port 119)
server:   200 BANZAIVAX news server ready,
          posting allowed.
client:   POST
server:   340 Continue posting; Period on a line
          by itself to end
client:   (transmits news article in RFC850
          format)
server:   240 Article posted successfully.
client:   QUIT
server:   205 BANZAIVAX closing connection.
          Goodbye.
```

2.6. Usenet Structure and Propagation

The Usenet network consists of high speed ISP's and large network (campus) servers forming the Usenet backbone. From there, anarchy sets in and a mishmash of servers connect and interconnect in a web-like fashion. This hierarchy is not strict, redundancy occurs as many NNTP servers maintain connections to many other servers for quick propagation.
Transmission can be controlled by looking at message ID's to avoid duplication. Most Usenet servers only maintain a direct connection to their upstream neighbour, and to any downstream sites to which they provide service.

A server is said to receive a *news feed* from its upstream connection, it then provides a news feed to all the servers downstream from it. Users connected to any of these campus servers can read this article or post a new one.

A physical metaphor used to describe the propagation of Usenet articles is what happens when water is poured over an uneven surface (Palme 2000). *Flooding* or *flood-fill* is NNTP's most important function. It enables dispersion of vast amounts of data (Usenet articles) between servers.
The flood-fill algorithm works as follows: A server receives a new article (from a poster or from another server). The server then uses the *check* message to all the peers he knows that will be interested in this article. If the peer replies that the article is wanted, the server uses the *takethis* message to feed the new article (Sit, Dabek et al. 2004). The peers, in turn will go through the same algorithm, and soon the whole of the Usenet will be flooded with the article.
The following chart from Marc Smith illustrates the physical networking structure of the Usenet using NNTP servers.

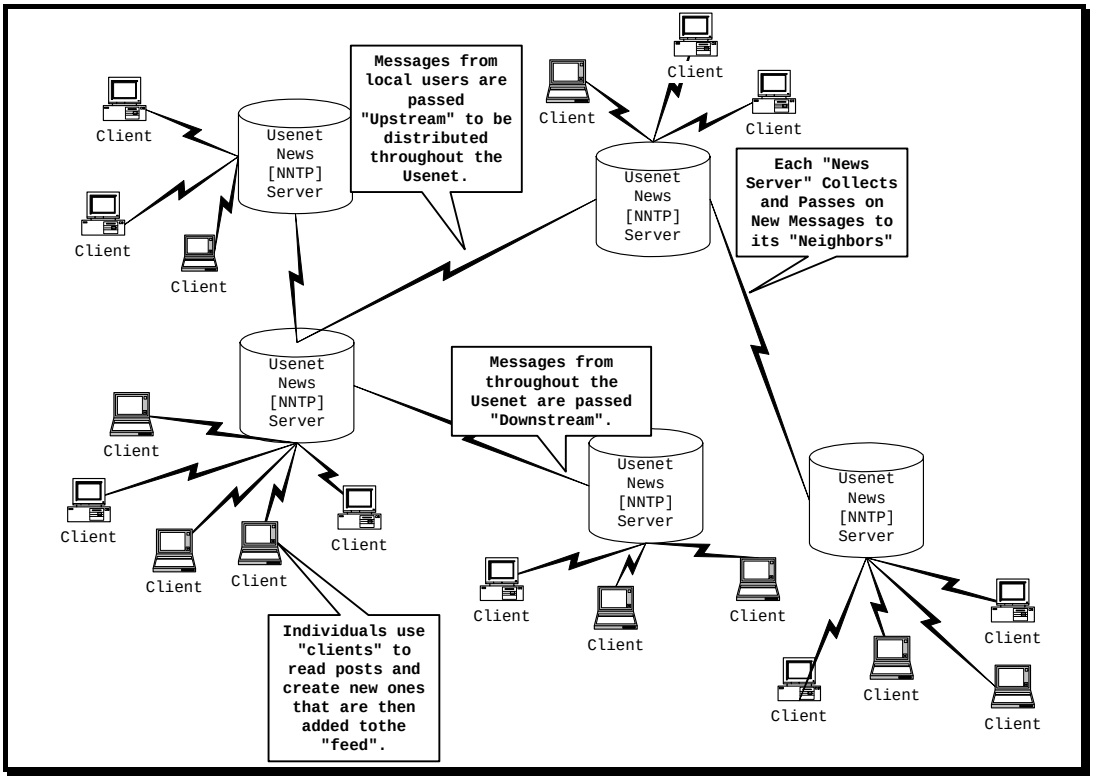


Figure 2: Technical structure of the Usenet news servers (Smith 1999)

2.7. Current handling of Usenet traffic[1]

During the last 5 years or so, the volume of news has been going growing fast, largely due to the exponential growth of binary postings. Previously, the news servers used the full NNTP protocol to exchange full feeds with peers and also service its readers.

To enable greater performance, today's servers split these two roles, which essentially amount to splitting the protocol: The peer *communication server*, also called ***transit server***, takes care of the flood-fill part of the protocol, i.e. *ihave*, *check*, and *takethis*. The news administrator just needs to find 4 or 5 administrators to peer with, and both sides then configure their servers to offer every article to all the people they are connected to.

[1] Sincere thanks to Josh Gagliardi from Highwind.com for clarifying this.

The second type of server, the ***reader server*** just takes a single copy of a complete news-feed, and worries about indexing this news-feed into a form suitable for the rest of the NNTP protocol: *list*, *xover*, *article*, and *post*.
They accept the *ihave*, *check*, and *takethis* commands, but typically don't issue them, other than to send articles which originated on this server via *post* to the rest of the world.

Transit servers are 'self-serving': all they care about is the *message-id*. Reader servers will typically insert, replace, or remove the *nntp-posting-host* header. If this header isn't checked when the article is posted, it's unlikely any other server has the knowledge to do useful enforcement.

2.8. Usenet Servers and Clients

The INN Server software

The Internet News Server is *any computer that manages news-feed services, including connections to external news-feed configurations and control of client access to newsgroups* (Hewlett-Packard 2002).

INN software was originally written by Rich Salz and is free software, supported by the Internet Systems Consortium.
The server may be configured either as a standalone server (for local bulletin boards such as private corporate news articles) or as a server that is connected to the Usenet. There are also two ways of connecting to the Usenet: As an internal node or as a leaf node. Internal nodes propagate news articles to other nodes. Leaf nodes receive a large volume of articles but do not propagate to other nodes (servers) on the Usenet. However, leaf nodes can still post articles to other nodes.

This means that there are three main configuration types possible for a news server:

1. As an internal node for propagating all news articles.
2. As a leaf node to receive articles and send out local postings.

3. As a standalone news server, managing a local bulletin board.

NNTP clients - Usenet readers

Just as email clients use email application software (such as Outlook) to read and send email, so does Usenet use clients named *news readers*. News readers are software programs that provide user access to the Usenet through NNTP. News readers also can post and read. Note that many email clients also function as NNTP clients.

Posting an article is simply done by the client establishing connection to the server and issuing a *post* command. The server then accepts and replies with a prompt for the client to send its article. Following that, the article is transmitted from client to server.

You can interact with NNTP by using Telnet to connect to it on port 119. The basic syntax of a NNTP command is

```
<command-code><parameters>
```

3. Visualisation - Mapping various aspects of the Usenet.

This project involves the development of a tool that maps out the various paths an article traverses to numerous news servers. The result is a graphic representation which facilitates final analysis. As a matter of fact, the Usenet is a favorite medium for certain types of projects requiring mapping and statistics, especially where it concerns social changes and network connectivity. This is mainly due to the vast volume of data available for analysis and the socially-based semantics that can be derived from this data. In this light it is useful to briefly describe some of these mapping projects. Examining these maps also provides a good overview of the evolution of the Usenet in general.

Spanning a 5 year period, the Netscan project created visual interfaces for data generated on the Usenet. During this period, about 1.2 billion Usenet messages were created by 48 million identities and sorted into 150,000 different newsgroups (Turner, Smith et al. 2005). This provides us with a vast database for research. Turner et al used two types of analysis on the data: overview analysis and selective analysis. Questions such as how the social interaction is changing over time can be visually represented. The following chart represents the changes over 2000 - 2004 of Usenet postings and answers,

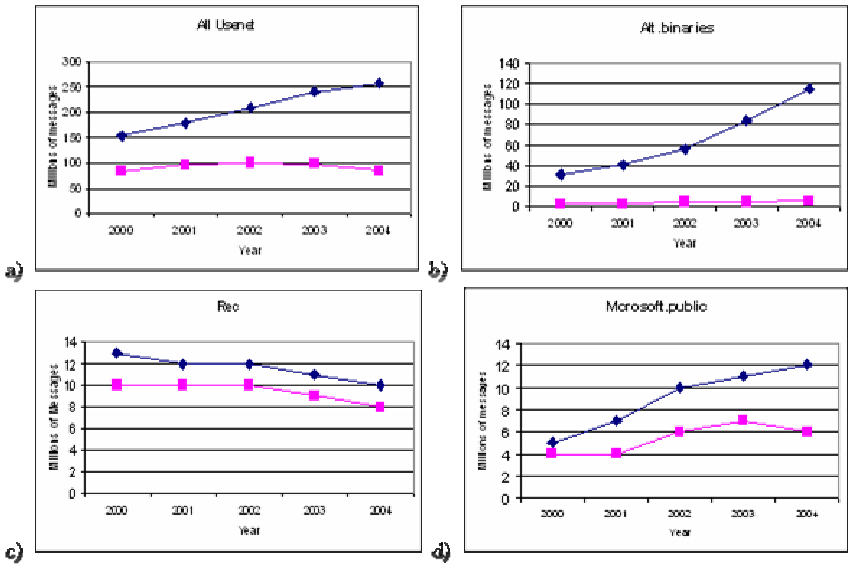


Figure 3: Number of messages (top) and number of replies (below) for all Usenet and three sub-hierarchies between 2000 and 2004 (Turner, Smith et al. 2005)

Tree maps

Tree maps are maps that show hierarchies. Following tree maps show the changes in hierarchy of newsgroups over the same period.

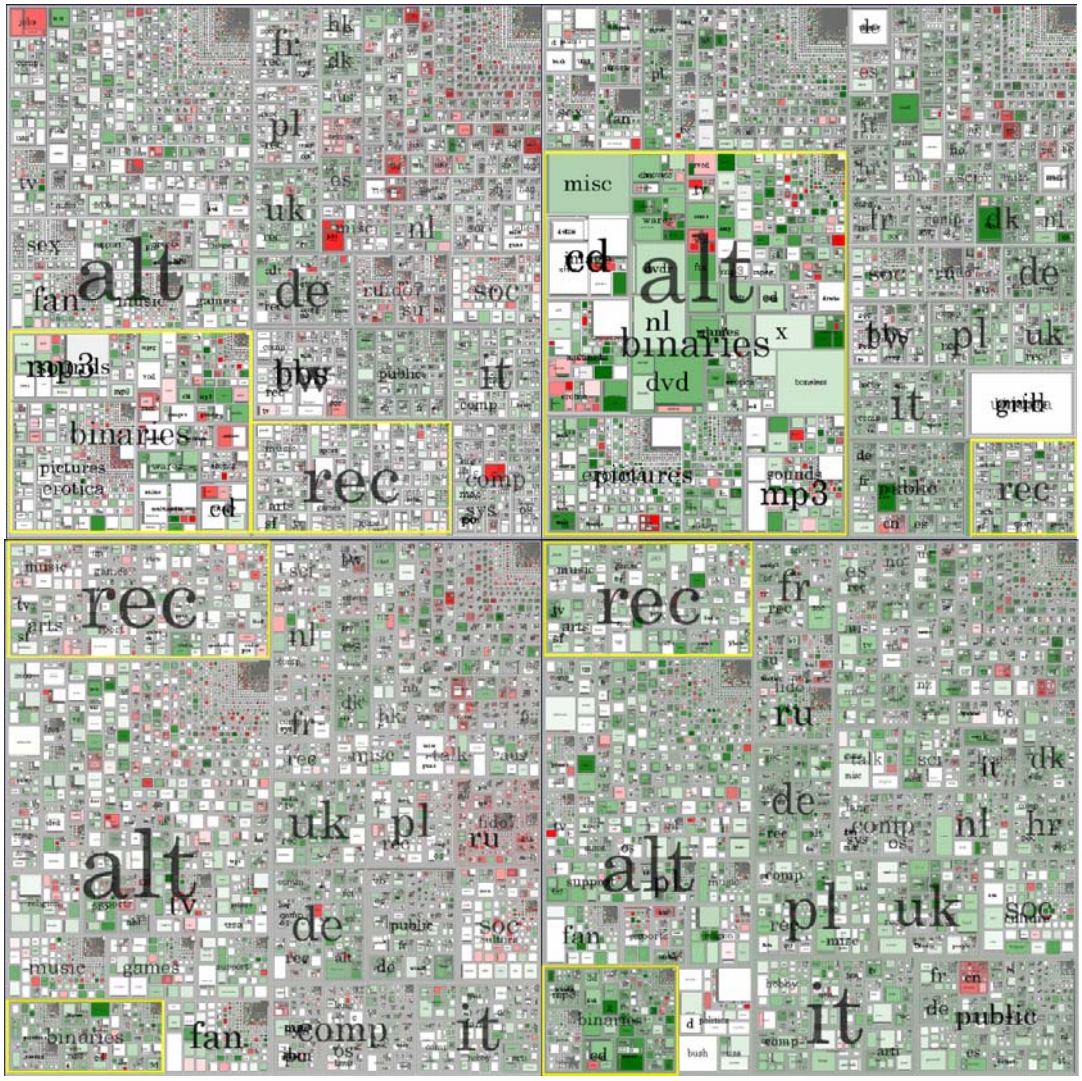


Figure 4 (Turner, Smith et al. 2005)

Top left : Posts to all of Usenet - Treemap for January 2000. The hierarchies alt.binaries and rec are highlighted.

Top right: Posts to all of Usenet - Treemap for January 2004. The hierarchies alt.binaries and rec are highlighted.

Bottom left: Replies to all of Usenet - Treemap for January 2000. alt.binaries and rec hierarchies are highlighted.

Bottom right: Replies to all of Usenet - Treemap for January 2004. alt.binaries and rec hierarchies are highlighted.

Note how the above maps illustrate that the newsgroup distribution hasn't changed much from the replier's perspective, while from the poster's side, great changes have taken place, largely due to the growth in binary postings.

Turner gives many more examples of charts illustrating various aspects of so-called *social spaces* on the Usenet.

Traffic flow maps

Another good source for network mapping is the magazine Mappa Mundi. One issue describes John Quarterman as the longest serving cartographer. Quaterman has a research consultancy Matrix Internet Derctory Services (MIDS) (Dodge 2001). His earliest contribution to Internet mapping is described in his book *The Matrix: Computer Networks and Conferencing Systems Worldwide* (Quaterman 1990). (Dodge 2001) discusses Brian Reid's Usenet Traffic flow maps. Reid has been producing detailed maps of the Usenet traffic from 1986 to 1995. His maps were the primary source at the time, they enabled people to determine which nodes were important.

The following figure show the 1993 Usenet world map (the cartographic tool used was Netmap):

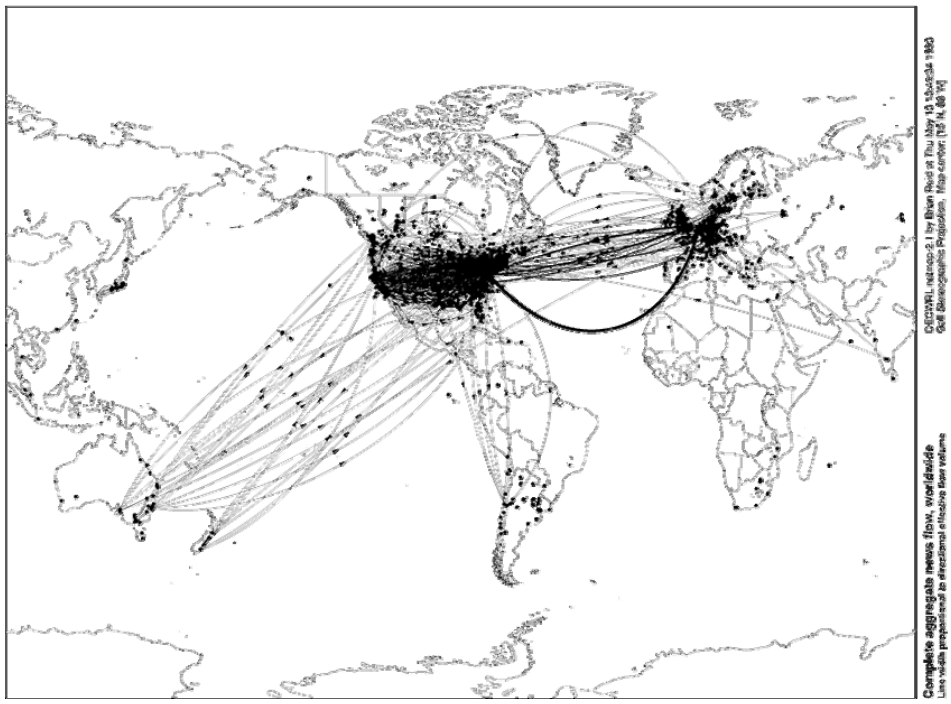


Figure 5: Map of complete aggregate news flow, worldwide, from 13 May 1993. (Dodge 2001)

Traffic flow maps may prove very useful for a future extension of this project as it would enable determining which paths are valid or likely and which paths are not likely. However so far, no recent maps have been published to my knowledge. The reason for this is probably the fact that propagation on the Usenet is a little more complicated these days, an issue I touched upon in chapter 2.7.

Conversation maps

Some browsers have the capability to analyse the text of newsgroup messages and produce a graphical interface that enables one to search and read messages. A set of newsgroup articles can be browsed depending on who is talking to whom. (Sack 2000) gives an example of this using the Conversation Map system to analyse about 1200 messages from a particular group. The top part of the map shows the social networks, discussion themes and a semantic network. The lower part shows the message threads. There is also interaction possible between all the sections.
An interesting section *Text analysis Procedure* gives a step by step outline of this procedure using parsing.

4. Spamming

In very general terms, *spamming* is the act of sending many copies of the same message to people who did not choose to receive it. The original meaning of the word spam refers to a kind of Usenet posting. Usenet spam is defined as a single message sent to 20 or more Usenet newsgroups.

4.1. History of spam

The first incidence on the Usenet labeled the term *spam*, is related in a Scientific American article Spam, spam, spam, lovely spam (Hayes 2003):

> In April of 1994 a message with the subject heading “Green Card Lottery- Final One?” was posted simultaneously to 6,000 Usenet newsgroups. The advertisement, signed by the Phoenix law firm of Canter & Siegel, offered information and legal services to immigrants.
> Thousands of Usenet regulars—incensed not only by the commercial nature of the message but also by the waste of bandwidth and the breach of “netiquette”—hounded Canter & Siegel by email and fax and telephone. The lawyers’ Internet access was cut off, and eventually the firm went out of business; Canter was disbarred. There have not been many such victories in the fight against spam.
>
> As it happens, the Canter & Siegel incident was the event that first popularized the term “spam.” The ultimate source was a 1970 skit on the British television show *Monty Python’s Flying Circus*, about a restaurant with a limited menu and a chorus of Vikings chanting “Spam, spam, spam, spam, lovely spam, lovely spam.” (Incidentally, Hormel Foods, who make SPAM rather than spam, attempted a defense of their brand name, then decided to have a sense of humor about it.)

Templeton, who is regarded as one of the authorities on the history of spam, notes that spam originally started on the Usenet and migrated to email (O'Brien 2003). O'Brien also lists the first spam Usenet post (though it wasn't called spam yet) in appendix 3 of the above referenced document. This post dates from 1988 and basically consists of a poor student's plea for financial assistance.

(Patterson, Anderson and Borrelli 2003) point to a six month study conducted by the center for Democracy and Technology claiming that 97 percent of the spam is received by email addresses posted on the public web (harvested by spammers with web crawling programs). The study claims that the amount of spam received from Usenet messages is much less (though still significant).

4.2. Usenet Spam

Definition of Usenet spam

Spam on the Usenet manifests itself in two ways: *Excessive Multi Posting* (EMP) or *Excessive Crossposting* (ECP). Originally only EMP was considered spam, but ECP proved to be an equal nuisance to the Usenet and its users. Often a subtle combination of both EMP and ECP is used.

Crossposting

As stated before, *crossposting* is sending a single message to more than one newsgroup. This is generally a good thing because it allows you to reach more groups with less impact on the net. A good practice is to ensure that the *followup-to* header is set to only one group. However, when the amount of crossposting is excessive, the end stops justifying the means and the excessive traffic becomes a burden to the network and a nuisance to the Usenet readers.

Excessive Crossposting - ECP

ECP means that a single message has been crossposted to too many newsgroups. The quantitative interpretation of ‘too many’ depends on other factors also, hence the introduction of the Breidbart Index, which is discussed below.

Excessive multiposting (EMP)

EMP means that too many separate copies of an identical or substantively identical article have been posted. Since each of these articles is posted separately, maybe to different servers, they will get different message ID’s and hence avoid ‘annihilation’ by the INN software.

A substantively identical article is an article that is sufficiently similar to the original one. Both the original article and the copy can be

1. identical from byte to byte.
2. slightly customized for each group it appears in.
3. articles that advertise the same service.
4. articles that have the same signature
5. articles that incorporate other user's postings, but are otherwise identical.

This fuzziness in definition can cause disagreements on whether an article is EMP or not. Take for example a binary posting in the *alt.binaries.** newsgroup. A binary posting constitutes of many, say 50 postings, each of which is an encoded section of one binary file (such as a DVD movie). These can be regarded as articles that bare the same 'signature' or 'advertise the same service'. Others would argue that the articles themselves are substantially different and therefore can not be regarded as EMP spam.

Breidbart Index

The Breidbart Index is a formula, invented by Seth Breidbart, which quantifies the degree of 'spamminess' (EMP or ECP) as a single number. Currently, various versions occur, all differing in their level of 'aggressiveness'. The higher the aggression, the lower the threshold is.

The Breidbart Index (BI) is defined as the sum of the square roots of n, where n is the number of newsgroups each copy was posted to.

Example: Two copies of an article are posted; one to 9 groups and one to 16 groups. The BI is

$$BI = \sqrt{9} + \sqrt{16} = 7$$

BI allows a maximum of 125 newsgroups.

The second version is the Breidbart Index Version 2, which is more aggressive than the original BI, allowing only a maximum of 35 newsgroups. Using the above example, the BIv2 is calculated as

$$BIv2 = \frac{\sqrt{9} + \sqrt{16} + 9 + 16}{2} = 16$$

The third version, called the Skirvin-Breidbart Index (SBI), is slightly less aggressive. It is calculated much the same as the BIv2, but sums the number of groups in the *followup-to* header, if available, rather than the newsgroups.

> *This [the followup-to header] is a moderated control field to make a copy of the posting in the specified newsgroup(Cotse.Net 2004).*

Adding to the example given above the fact that the number of groups in the *followup-to* headers was 4, then

$$BIv2 = \frac{\sqrt{9} + \sqrt{16} + 9 + 4}{2} = 10$$

4.3. Spam Cancel

A *cancel* message is generally issued by a news server, but can also be issued by the client. When an article is cancelled, it is effectively annihilated from all the Usenet servers. This is a very powerful action and can be used aggressively by people with ill-intend.

Generally, spam is cancelable if its BI is 20 or more over a 45 day period.

Note that a single posting would have to crosspost to 400 groups to reach a BI of 20. This is not possible given the limitations in the Usenet software.

Note also that the actual content of the spam is irrelevant in this calculation.

4.4. Spam and anti-spam Techniques

The Internet Watch Foundation gives a short rundown of some methods used by spammers to gain anonymity.

1. *Using bulk newsgroups posting software (Bulk-News 2002). This enables one to post to over 2000 newsgroups per hour using an anonymous proxy server.*
2. *Using anonymous remailers, normally used to forward mail. Spammers find an open mail relay server or open / public anonymous proxy server.*
3. *Using bulk posting / mailing packages (such as Stealthmail master) and direct via a remailer.*
4. *Using anonymous proxy servers. (IWF 2004)*

The IWF memo also provides various links regarding all 4 methods listed above.

(Baker 1997) lists about 12 anti-spam techniques, mostly to do with Spam headers and filtering. He handles these techniques with a particular news server (SBNews) but the techniques are also relevant in a broader sense.

Two anti-spam techniques involve the *xref* header. *Xref* headers specify the newsgroups an article is posted to and matches them with the server's article number (more on the *xref* header in chapter 6).

1. Spam is likely if too many newsgroups are listed in the *xref* header.
2. Use the listing of newsgroups to identify unwanted or offensive newsgroups.

Another technique involves limiting the download of message lines, a message with fewer than 10 lines is bound to be from a binary group or isn't really worth reading anyway. The next 2 techniques enable lockout. One locks out posters - spammers usually use bogus names frequently. The other one locks out messages with a particular subject.

Another interesting technique involves locking out the posting host. According to Scott Baker, Spam advertisers tend not to use a bogus posting host name.

Spam traps

Spam traps are set to facilitate tracing original spam postings. It usually involves posting with a unique address for each of the postings and for each of the header fields in those postings. (Zenger 2002) describes an elaborate spam trap he set for 2 years between 2000 and 2002.

4.5. Spam news servers statistics from 2006

It is necessary to know what main servers are out there, which ones tend to carry a lot of spam, and which ones are freely accessible.

According to (News.admin 2006), the frequency of domain appearances in the path is well below 50%. Only one server, newsguy.com, claims 100%, Giganews has 33%, while Highwinds has 20% occurrence.

The frequency of the domain being the originating point in the path is well below 5% in general (only Giganews and Eweka have 18%).

The Supplementary shows some statistics as regards the latest statistics on top spam newsgroups, news servers, and news servers carrying the most spam (spam hosts). One important conclusion to draw from these statistics is that the newsgroups most prone to spam are the *alt.binaries* groups, especially in the general DVD and erotica subgroups.

This led to the stage in the research where open servers where investigated that carry binary data. While general DVD postings adhere to the definition of spam, the intention of the spammer is not so much advertising or menacing, but in a sense altruistic, wanting to please others by distributing, generally illegal, copies of DVD’s.

4.6. Is the Spammer working from a server or a client?

It is obvious that a spammer that administrates a Usenet server has many tools to his disposal at a low level of implementation. However no spammer is totally untraceable and it seems that a spammer working from a server has one major disadvantage; the server can not change location easily, and being labelled as a spammer could undo a lot of work as regards finding suitable peers and the like. While it is true that a server can forge all the headers of an article, there are some aspects of the article transport that makes total anonymity impossible.
Say the server forges his identity by adding a path preload to avoid aggressive detectors; he still needs to relay his message to a transit server who by default needs to have the identity of the spammer's address. This address will be traceable in the article's header and needs to make sense to that transit server.
A spammer working from a client has many tools to his disposal also, such as a Control Client, covered in chapter 5. There is also the added advantage of mobility and anonymity, i.e. the spammer may be spamming from an Internet café.

4.7. Report spam

If the same article is spotted in several groups, each with different *message-id*s, the conscientious news reader can collect the complete headers of each article and check the newsgroup *news.admin.net-abuse.misc* if it has already been reported. If it hasn't been reported yet, the reader can start a thread in *news.admin.net-abuse.misc* or *news.admin.net-abuse.usenet* using the following format:

```
Subject: Spam (was Re: <original Subject>)
```

The same reader will have to ensure that all of the headers and as much of the body as deemed necessary are included.

5. Networking and Spamming Tools

Research in current and past Usenet spamming tools has been an integral part in the search for a path preload detection tool. While no such tool was discovered, there was a lot of information on Usenet tools in general, and about the approaches taken by the tool builder. Much information could also be found on programming languages and techniques.

(Saiedian and Winslade 1992) investigate the practicality of implementing news messages as active objects. Though written in 1992, it gives some examples in C++ on how to represent entities such as messages and header lines as objects.

The web page *http://spam.abuse.net* contains a link to a spam tracking page that tracks Usenet spam. The google group link can help determine distribution level of a Usenet spam (ITPro 2005).

Netscan is a software tool that gathers an ongoing stream of Usenet messages and maintains a database drawn from the header of each message. It then distils measures of activity and relationships in any collection of newsgroups selected for study (Smith 1999).

A project describing a system that gathered Usenet traffic, and consequently traced and analysed them is described in an educational set of 39 powerpoint slides (Saito 1998). The slides explain in detail how servers communicate with other servers or with clients. It then describes the setup used to gather data: First a news server is set up in which the Usenet can pump the full feed. A client-side trace is consequently set up on the same subnet as the Usenet to capture all the NNTP packets. The slides also run through some server structure configurations and anatomy of the INN. One conclusion stands out in this project: Most of the traffic was due to a small number of automated clients connecting to the server regularly to download large number of articles.

A full list of possible networking tools to fight spam are given and elaborated upon in (Zocholl 1996). It lists these techniques sequentially thereby giving the reader step by step instructions on how to fight spam (including email spam). The general run of investigation is

1. Inspect important headers
2. Use of networking commands: Use a web search engine (such as Google) to look for references to the domain names found.

It is possible to compromise any field in a post when working from a server; the file in the news spool can be edited as root, a post can be intercepted in a 'man-in-the-middle' attack between the server and the poster by spoofing the IP of the server to the client.

5.1. Usenet Control Clients

Some newsgroups, such as the *alt.admin.** newsgroups, carry heated discussions on various matters regarding spam. A frequent topic of discussion involves *NewsAgent*. NewsAgent is a *Control Client*, a multi-threaded Java application that is a powerful spamming or anti-spamming tool. The documentation is very useful for this project's purposes as it gives an insight into the capabilities spammers have available to them. Following are some excerpts from this documentation. The headers alluded to will be covered in the next chapter, chapter 6.

- *On today's Usenet, a control message is nothing more than a suggestion that servers take a particular action.*
- *Ihave, the other way to post … The beauty of posting by this method is that it allows you to set ALL of the headers, including the hateful NNTP-Posting-Host.*
- *With HITCH, you can cancel articles in moderated groups.*
- *Message-ID header … must contain some alphanumerics on both sides of the @-sign, and be less than 240 characters long.*
- *Preloaded Path headers is one of the best ways to mask the location of where an article originated. Before the advent of NNTP-Posting-Host header, inspection of the Path was fairly important. Nowadays it's still swell, but the*

swelling's gone down. People often read the Path, but it's usually misleading anyway.

- *The open-access nature of CyberCafe's renders the dreaded NNTP-Posting-Host header worthless. Tracebacks to the internet equivalent of a phone booth aren't all that helpful.*
- *Anonymity – using proxies to hide your IP address. To protect netizens from the Cabal's gang-bang complaint attacks, Newsagent can make connections through a variety of proxy protocols, each of which listens on a different port number.* [2]

5.2. Basic networking commands

The detection of preload can not be regarded as an isolated technique. To truly confirm preload a variety of tools are needed.

Network commands are generally executed on the command line, though working behind a firewall makes this difficult. The *Path Analyser*, the tool developed for this research, has links to web sites that will execute these commands from their servers. This has the added advantage that different results can be obtained using one command. The most interesting commands are those that establish routes, such as *traceroute* (see below).

The following networking commands are most useful:

dig and nslookup

The command *dig* queries domain name servers for information about certain host/domain names. The alternative is the *nslookup* command, which also resolves host/domain names but provides less information.

```
dig <ipaddress>

dig <hostname>
```

[2] HipCrimes's NewsAgent (v2.0) by the Phantom of Sympatico accessed April 2006 pp8

whois

The *whois* command returns the administrative and technical contacts for the hosts/domains.

traceroute

The *traceroute* command returns the route from the location where the command has been issued to the location supplied with the command. This essentially is a sequential list of all the locations that a message (issued by the traceroute command) traverses.

The Path Analyser uses this command to map trajectories from random web servers to locations that need investigating. An example follows:

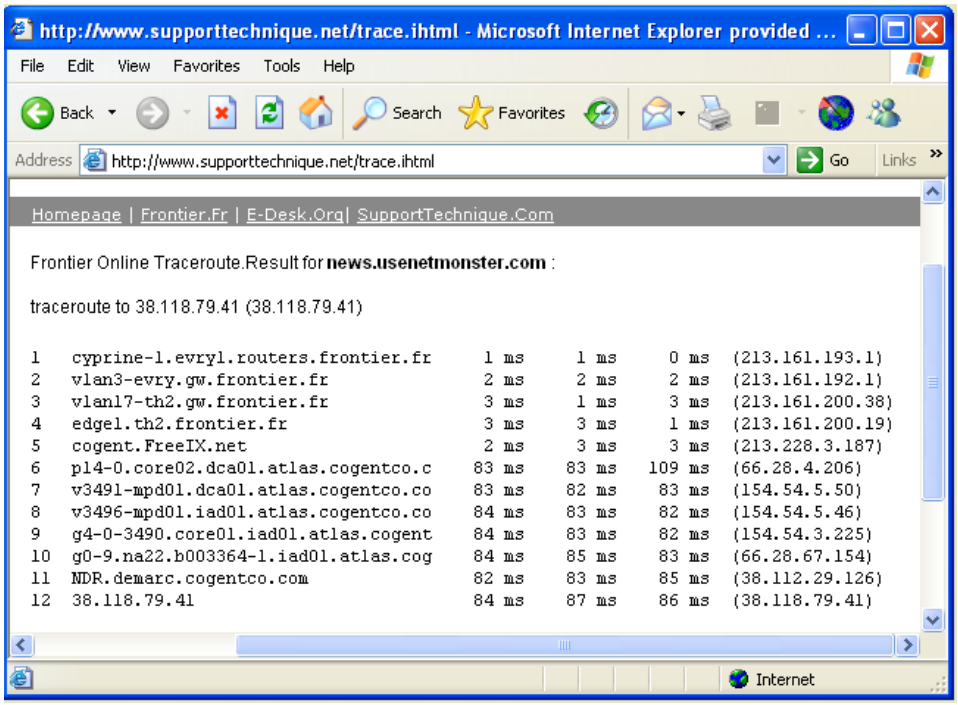


figure 6: trouceroute result

The traceroute tool on the web page *supporttechnique.net* was used to investigate the linking from their server, *cyprene-1.evryl.routers.frontier.fr*, to the *news.usenetmonster.com* server (which has an IP address of 38.118.79.41).

The command issued on this web page was

```
traceroute news.usenetmonster.com
```

5.3. Additional networking tools

Geolocation

Knowing the geographical location of servers is not necessarily an indication of their linkage on the IP level, i.e. servers that are geographically close to each other are not necessarily closely linked on the IP level, and vice versa, servers located at great distances from each other may well turn out to be peers.

For that reason, knowing the location may not be all that useful, however, if after determining the common path 2 servers are found to be connected and prove to be minor servers located a great distance from each other, there should be more cause for suspicion.

All IP addresses are unique, and every server is officially registered. However, IP addresses are handled by hundreds of institutions. Some companies keep databases of this IP address/location information. These databases are easy to integrate into any web application. The general stated accuracy of this information is from 80% to 97%, though it is probably safer to take this figure at the lower end of the scale.

Many web sites offer geological look up, some even draw maps of server connections on the IP level, though this procedure is often slow and many servers are not identified because they prefer to be anonymous.
Following is a result of a geological lookup for the IP address 204.153.244.170[3]:

```
Hostname        204.153.244.170
Country Code    US
Country Name    United States
Region          MN
Region Name     Minnesota
City            Minneapolis
Postal Code     55432
Latitude        45.0946
```

[3] Using the web tool at http://www.maxmind.com/app/lookup_city

```
Longitude       -93.2582
ISP             Mithral Communication & Design
Organization    Mithral Communication & Design
Metro Code           613
Area Code            763
```

Server networking tools

While this research uses techniques available to the client, it is of use to know what servers do to fight spam. (Baker 1997) lists various useful techniques utilized by the news server SBNews.

What	Where	Why
Maximum XRef Limit	Configuration:Preferences	Ignore messages posted to more than a specified number of newsgroups
Lockout XRef	Configure:Lockout:Xref	Ignore messages posted to specific groups
Preload XRef	Configure:Preferences	In combination with above, pre-loads the "Xref" information so that SBNews can ignore a message without having to start downloading it.
Minimum Message Lines	Configure:Preferences	Ignore messages with too few lines in them to hold meaningful data
Lockout Poster	Configure:Lockout:Poster	Ignore messages form a specified person (or any "From:" header line containing the specified pattern)
Lockout Subject	Configure:Lockout:Subject	Ignore messages with a specific subject (or any "Subject:" header line containing the specified pattern)
Add "free" to Lockout Subject	Configure:Lockout:Subject	Lots of 'Pay' services put 'Free' in the message subject.
Add "http://" to Lockout Subject	Configure:Lockout:Subject	Lots of 'Pay' services put their http:// address in the message subject
<Headers> button	<Headers> Button	Manually view message headers and reject/lockout the ones you don't want.
Lockout Any	Configure:Lockout:Any	Lockout any phrases which you know you don't want to appear in desirable messages.
Lockout Posting Host:	Configure:Lockout:Posting Host	Lockout a specific host (i.e. ISP or service provider) which is permitting SPAM
Preload host hdrs	Configure:Preferences	Download the posting host headers ahead of time so that SBNews can ignore a message with an invalid host before starting to download it. (used in combination with the lockout posting host feature)
Complain	n/a	Submit complaints to the the offender's ISP to stop them. (sometimes it does work, but not usually)

table 1

6. Headers

6.1. General discussion

Understanding how the Usenet headers are structured and used is one of the main issues in this project. There are quite a few examples to be found where an analysis is done on the Usenet message headers. A particular illustrative example is given by (Marjie 1999):

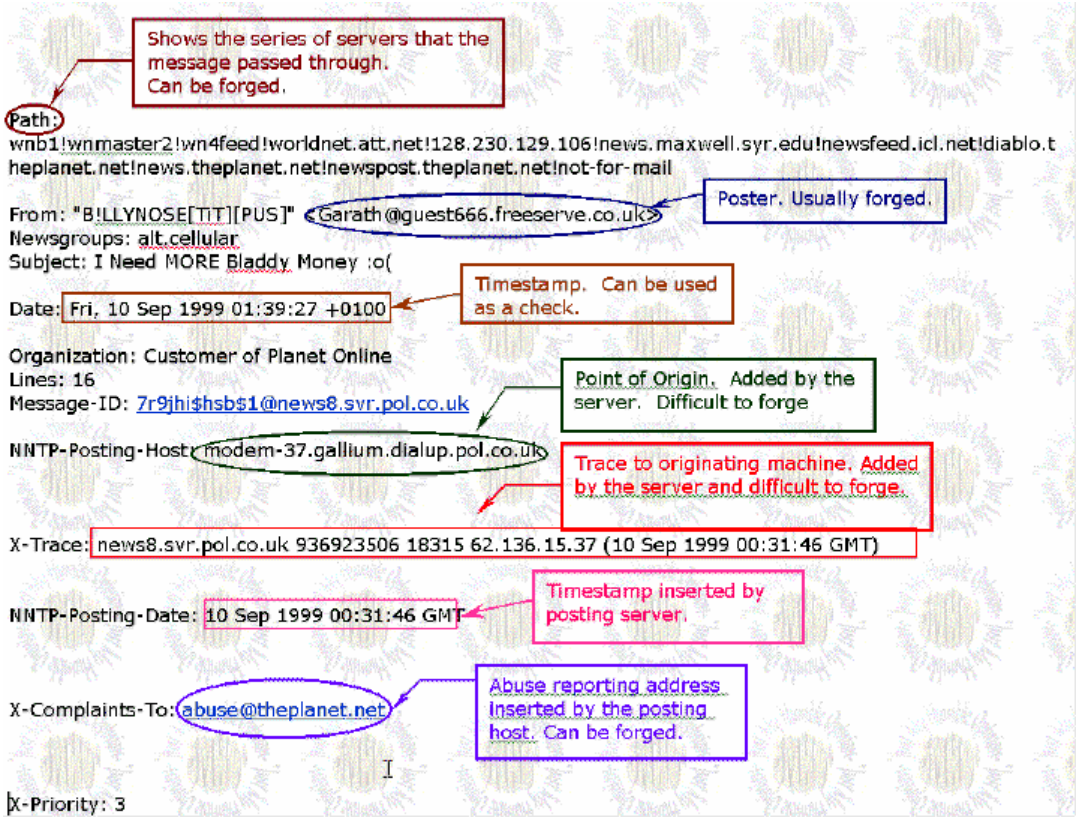


Figure 7: Relevant parts of the Usenet header(Marjie 1999)

The web site explores all the path inputs in turn (starting from the last entry) and uses a combination of networking tools and common sense to verify all the paths in the header. The common sense is largely derived from the knowledge as to whether the header is easily compromised or not.

Since Usenet predates Internet, its path header syntax predates the DNS. This means that the path entries shown are either from a name registered in the UUCP map or the full DNS name of the server.(Ingvoldstad 2001). This is due to the fact that Usenet traffic may occur on both UUCP (though less so these days) and TCP (NNTP).

(Cotse.Net 2004) lists some headers collected from a sample of selected newsgroups in 1998. The list is exhaustive and has been very useful in determining obscure headers found in some of the messages encountered during this research. Each header is also given an indication as to how frequently they occur (using the *rarely, common, etc* classification).

All Agents MUST generate header fields so that at least one space immediately follows the ‘:’ separating the header field name from the body. No field can be empty. Headers can be roughly classified as *mandatory*, *optional*, and *additional*. The mandatory headers are

```
from, date, message-id, subject, newsgroups, path.
```

The optional headers are

```
reply-to, sender, follow-up, expires, references,
control, distribution, organization, keywords, summary,
approved, lines, xref
```

Additional headers are used to provide extra information about an article. Some are entirely custom-made while others are so frequently used that they have become standard (such as the *nntp-posting-host* header). The ‘x-’ prefix indicates the fact that these are *extra* headers:

```
nntp-posting-host, x-newsreader, x-trace, x-complaints-
to
```

The headers of most interest to us are

```
path, message-id, nntp-posting-host, injection-info, and
x-trace.
```

6.2. Description of main headers

(Scragg 2002) gives an account on how to trace the headers of a Usenet Posting. His examples are clear and descriptive:

> *Xref: news.demon.co.uk alt.retribution:17474 alt.flame:253990 aus.flame:19841 alt.fan.karl-malden.nose:348243 demon.local:198435*
> All this tells us is that on my news server, this post received the article numbers that are quoted after the newsgroups. No help to anyone.
>
> *From: me@alt.retribution*
> Easily faked, forged or plain stupid. From: headers are often found in either spam, junk or deliberate flames.
>
> *Newsgroups: alt.retribution,alt.flame,aus.flame, alt.fan.karl-malden.nose,demon.local*
> Only tells us which newsgroup it was posted to.
>
> *Subject: <deleted as possibly defamatory>*
> *Date: Thu, 30 Mar 2000 20:08:14 GMT*
> *Organization: Retribution*
> Doesn't give us anything other than when the article was posted
>
> *Message-ID: 38e7b209.48314@news.freeserve.co.uk*
> This should usually tell us which ISP the article was injected from. It can be faked but often shows the real ISP. If there is more than one, it means the ISP or software automatically adds it and the poster has tried to disguise themselves.
>
> *Reply-To: me@alt.retribution*
> See information for From: - cannot be trusted
>
> *NNTP-Posting-Host: modem-86.north dakota.dialup.pol.co.uk*
> Usually shows the host or dialup that the post was made through and is very often added by the ISP and it will then show the dialup that the post was injected through. Can be used by ISPs to then trace who posted an article.
>
> *X-Trace: newsg4.svr.pol.co.uk 954446679 6807 62.137.85.86 (30 Mar 2000 20:04:39 GMT)*
> X-Trace is added automatically by a lot of news servers to help track down fake posts and spam.

If present it is a good indication of the real ISP involved.

NNTP-Posting-Date: 30 Mar 2000 20:04:39 GMT
Again, can be faked - worth checking the posting time as above to see if they are appropriate. Not much use as it usually only says when the article was ready for posting.

X-Complaints-To: abuse@theplanet.net
Many ISPs add this automatically when news is injected through their servers. If present then is usually a very good indication of whom to complain to.

X-Newsreader: Forte Agent 1.5/32.452
X-No-Archive: yes
Not a lot of help, only tells us their newsreader and they've requested that it is not archived at such as Google Groups.

6.3. The 'path' header

As a news article travels through the news servers, each server will append its identity to a section of the header, called *path header*.

Relating to the path header, the RFC 1036 standard suggests that as a system forwards a message, it adds its own name to the path list. Usually the right-most name is the name of the originating system; however, to enable compatibility with older systems, an extra entry can be added to the right with the name of the sender. The string *not-for-mail* is now used instead of the name of the sender (since at one time the whole path could be used as a mail address for the sender).

The protocol lists the following uses for the path header:

- Monitoring Usenet routing for performance.
- Establishing a path to reach new hosts.
- Enabling a host to determine who already has the message so there is no need to send it to any system in the path.

The path header is present in every article and should be read from right to left (first server to last server). However the examples taken from this project's results sometimes show the path in reverse, which is a consequence of parsing. The paths in the examples given below should be read from left to right, which is, after all, more intuitive. The start of the path is easy to detect since it always begins with the *not-for-mail* entry.

When a server receives an article and adds its identity to the path, it may *stamp* extra data to its entry. Two stamps are used regularly by today's servers, *posted* and *mismatch*.

The *posted* stamp

This stamp may be added by the first server that receives the article. Therefore it should be early in the path. For example, the following path section shows that the *posted* stamp is added appropriately, the first 2 nodes being internal:

```
not-for-mail! 53ab2750!
read2.cgocable.net.POSTED! feed.cgocable.net!
news.sprintnetops.net! news-out.readnews.com!
transit3.readnews.com
```

Note that the number `53ab2750` at the beginning of this path is either a client ID that identifies the user or a control code. Also, for clarity of presentation, spaces have been added following every '!' symbol.

The *mismatch* stamp

The *mismatch* stamp indicates that the previous agent failed to be so verified. This is a pretty good indication of foul play, or at best it indicates sloppy server configuration. If the transit server has been told in its configuration that the aliases for a peer are X, Y, and Z, and it receives an article from that peer whose path stamp isn't X, Y, or Z, it will stamp *mismatch* next to that path entry:

```
<IP address>.MISMATCH
```

Take the case of the following path section:

```
lotis.uk.clara.net! monkeydust.news.clara.net!
195.245.201.10.MISMATCH! wagner.news.clara.net!
news.clara.net! solnet.ch
```

The networking command *whois* will reveal that 195.245.201.10 is a *claranet* IP, indicating a miss-configuration, either of server-server communication or of a server to allow injection of articles from end-user IP's.

6.4. The ‘nntp-posting-host’ header

This header specifies the IP address or the domain name of the posting host. The address is usually of the poster’s client or his local NNTP server. It is added by the first server that receives the article. Though not one of the mandatory headers it seems to appear quite often. In my sample research, I determined the occurrence of this header in 62.5% of the articles.
For example, take the following path:

```
not-for-mail! 53ab2750!
read2.cgocable.net.POSTED! feed.cgocable.net!
news.sprintnetops.net!
news-out.readnews.com! transit3.readnews.com
```

That same article also had the following header:

```
NNTP-POSTING-HOST 24.141.179.155
```

Using *nslookup*, this revealed:

```
Name:    d141-179-155.home.cgocable.net
Address:  24.141.179.155
```

This indicates that the posting was done on one of the *cgocable.net* hosts, probably through a dialup, indicating that this stamp is valid, though it needs to be kept in mind that all this can be part of a more elaborate hoax which also uses path preload.

6.5. The x-trace header

The *x-trace* header is generated by secure news servers to identify all postings through their site. This allows for cross referencing. Generally, the *x-trace* header shows the posting server’s IP. However, sometimes the x-trace header is incomprehensible, for example, one of the articles (article 71) showed the following x-trace header:

```
X-Trace:sv3-mlDqHqsFbdeZS/g6PtG9vGOWxSvkEiPLQfEu2eb/
GjlfMN/IBM50BJeVbzWLjrBZw4BMb5GoONpYICL!bdubim4XLejk
JVKZ4UAgXOqcOLe0zPFLi7ZjwxIX+GVeLBAHldlXzvmTzLGCbw==
```

This could be a result of encryption, but more likely may be an indicator of spam. Indeed, *User Control Clients* like *NewsAgent* often build headers using random text.

More often, as stated, the IP address of the posting server appears. Another article (Article 34) had the following header section:

```
Path: pubnews.gradwell.net!news-peer-lilac.gradwell.net!
bbc! newspeer1.nwr.nac.net! newspeer.monmouth.com!
newscon06.news.prodigy.com! prodigy.net!
border1.nntp.dca.giganews.com! nntp.giganews.com!
postnews.google.com! j33g2000cwa.googlegroups.com! not-
for-mail

Message-ID:
<1143645815.373682.168300@j33g2000cwa.googlegroups.com>

NNTP-Posting-Host: 130.13.254.85

X-Trace: posting.google.com 1143645820 3130 127.0.0.1
(29 Mar 2006 15:23:40 GMT)
```

If the article already contains an *x-trace* header, the server renames it to *x-orig- x-trace* and adds its own genuine *x-trace* header. Therefore the simple fact that an *x-orig-x-trace* header exists points to the fact that the header may have been falsely constructed before it arrived at the posting server, e.g.

```
X-Orig-X-Trace: 17982142291 134617 141.133.201.172
```

It may be useful to use networking commands such as *traceroute* and *whois* to the address contained in the *x-orig-x-trace* header to further verify the validity.

6.6. The message-id header

The *message-id* is generated by the news server software. It is unique for every message and should never be duplicated within 2 years. In practice, the code is based upon the date and time and therefore may never be duplicated. Even if the message is dropped, it still can be traced by its *message-id*.

The *message-id* header most often carries the address of the server that the article was posted on and should therefore correspond to the posting server's domain name address.
Following example shows how the article was posted on the German server *united-newsserver.de*:

```
Message-ID: <44731b5b$3$59880$afc38c87@news6.united-
newsserver.de>
```

The RFC2822 standard has tightened the guidelines on the format that the *message-id* should take. While the following notation would be considered semantically equivalent,

```
<abcd@example.com>
<"abcd"@example.com>
<"ab\cd"@example.com>
```

only the first of them is syntactically permitted by the RFC2822 standard.

6.7. Other Headers relevant to the research

Many headers have not been named so far. A list of the ones relevant to our discussion is given next.

The *xref* header

xref identifies the original article number on the Usenet server. It specifies newsgroups that should match the *newsgroups* field, which is a mandatory field that specifies all the newsgroups the message is intended for. For example, the RFC 1036 gives an example listing two newsgroups with their message number:

```
xref: seismo news.lists:461 news.groups:6378
```

(Baker 1997) states that analysing the *xref* header is a very effective tool for determining whether the message is spam or not. If too many newsgroups are listed (more than 8) then changes are you are reading a spam header.

While Usenet articles are uniquely identified by their message ID, they are also given a number by each Usenet server as they are received. These article numbers, which differ from one system to the next, are usually listed in this cross-reference header.

The *xref* information is used when a message is crossposted to multiple groups. In which case, as soon as a user reads the message in one group, it is marked as having been read in all the others where it was posted. This way, if the user later reads one of those other groups, they will not see the message again. An example follows:

```
Xref: authen.yellow.readfreenews.net
misc.health.infertility:38
misc.health.aids:131249
misc.health.alternative:546141
misc.health.arthritis:59688
misc.health.diabetes:350034
misc.health.infertility:120586
misc.health.injuries.rsi.misc:15961
```

The *injection-info* header

This header is a new addition and is not very common yet. In RFC 2822, the *injection-info* header is described as follows:

> *The Injection-Info header field provides information as to how an article entered the Netnews system and to assist in tracing its true origin. It can also specify one or more addresses where complaints concerning the poster of the article may be sent(Lindsey 2006).*

This header has many parameters and aims to replace common headers:

> *NOTE: Some of this information has previously been sent in non-standardized header fields such as NNTP-Posting-Host, X-trace, X-Complaints-To, and others. Once an injecting agent uses Injection-Info, it should have no need to send these non-standard header fields(Lindsey 2006).*

6.8. Headers of binary newsgroups

The binaries newsgroups are those located in *alt.binaries.** newsgroups. Binaries constitute the biggest part of the postings on today's Usenet.

From the statistics given in the supplementary, it can be seen that as of today, the top 5 newsgroups carrying spam are:

alt.binaries.dvd.music
alt.binaries.dvdrcore
alt.binaries.multimedia.erotica.voyeurism
alt.binaries.dvd.erotica
alt.binaries.x

The subject in the header is often a good indication of whether this message is spam or not. One issue worth mentioning is the fact that many binary posting headers seem to have very long paths. Take the following path for example.

```
Newsgroup: alt.binaries.dvd.music
Server: 210.60.184.3
message-ID:
<jxd8g.370589$4P2.264906@fe03.news.easynews.com>

Path: news.isu.edu.tw! news.nsysu.edu.tw!
ctugate! Spring.edu.tw! news.nctu.edu.tw!
newsgate.cuhk.edu.hk! newsfeed.asianetcom.net!
newsfeed1.dti.ad.jp! jpix! news0.dion.ne.jp!
newsfeed2.kddnet.ad.jp! newsfeed2.kddnet.ad.jp!
newsfeed.kddnet.ad.jp! newsfeed.kddnet.ad.jp!
newsfeed.dacom.co.kr! feeder.kornet.net!
newsfeed.skline.co.kr! nntp.kreonet.re.kr!
newsfeed.kreonet.re.kr! kreonet.re.kr!
keepthis.news.telefonica.de!
takemy.news.telefonica.de! telefonica.de!
newsfeed.arcor.de! proxad.net!
feeder1.cambrium.nl! feed.tweaknews.nl!
tudelft.nl!binfeed1.tudelft.nl! multikabel.net!
feed10.multikabel.net! newsfeed.news2me.com!
newsfeed2.easynews.com! easynews.com! easynews!
easynews-local! fe03.news.easynews.com.POSTED!
not-for-mail
```

7. Common Path and Path Preload

7.1. Definitions: Path preload, Common Path, Adjacent Server, and Boundary server.

Common Path and Path Preload exemplify the two stages this research covers, and understanding the difference between them is crucial. Both concepts are covered in the introduction. Following are a revision of these definitions and the introduction of some new definitions.

Path preloading is a technique used by spammers in which a string of fake server identities is inserted into the path header, even though the article did not traverse these servers. The first server that receives the spam is led to believe that the article already traversed the servers listed in this path preload, and consequently this receiving server will add its own identity to it. Such a server will be referred to as a *boundary server* from now on; the reason for this will be explained further on.

A *common path* is found by recording the path headers of the same article downloaded from widely dispersed news servers and determining which sections at the beginning of these path headers are common to all articles. The servers that are next in row to this common path are referred to as *adjacent servers*.

It must be taken in consideration that the common path may not accurately reflect preload, even if preload is present; for example, the last few entries in the common path may be strongly-linked servers, thereby effectively extending the common path past the path preload.

The hypothetical example given below illustrates many of the important concepts discussed so far. These concepts are used in the final analysis and a good grasp of the terminology is therefore important:

Assume a suspect spam article has been downloaded from 4 different servers, located all over the Usenet. If the first server had the following path header:

```
server1!server2!server3!server4!server5!server6!server7!
server8!server9!
```

The second server,

```
server1!server2!server3!server4!server5!server6!server21
!server22!server23!server24!server25!server26!server27
```

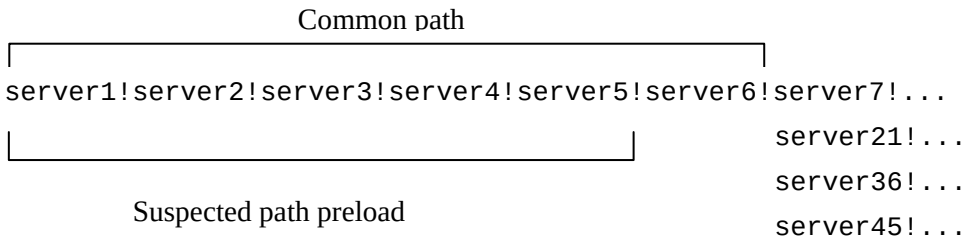
The third server,

```
server1!server2!server3!server4!server5!server6!server36
!server37!server38!server39
```

And the fourth,

```
server1!server2!server3!server4!server5!server6!server45
!server46!server47!server48!server49
```

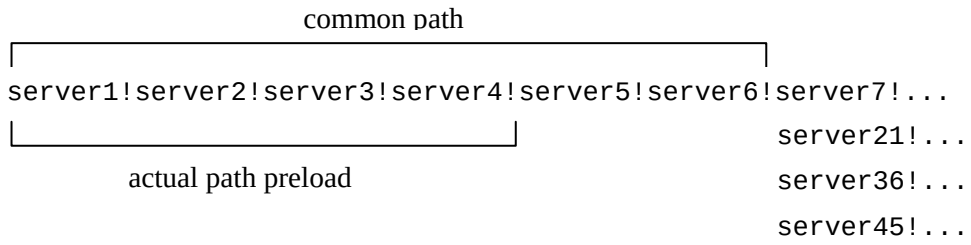
The path common to all these articles runs from server1 through to server 6. This is the *common path*. After analysis, the following map is produced:



The common path suggests that the spammer posted his article to *server 6*, not *server 7*, because the server that the article was (apparently) posted to is also common in all the path headers. Therefore the suspected path preload runs from *server 1* to *server 5*. *Server 6* is referred to as the *boundary server* because it is situated between the common path and the next *adjacent servers*.

However there is a possibility that the whole of the common path is path preload and that the spammer posted this path preload to different servers, in which case the *adjacent servers* are the servers that receive the spam.

Assume now the situation in which the first four servers are path preload, fabricated by the spammer:



The spammer posted his article with this path preload to *server 5*. However, the common path technique determined a common path of *server1* through to *server 6*. The most likely reason for this is that servers 4, 5 and 6 are strongly connected.

Note also that servers 7, 21, 36, 45 are *adjacent servers* and servers 9, 27, 39, and 49 are the servers where the articles where downloaded from.

Assessing Path Preload is a very intuitive operation which requires a lot of expertise and experience. This stage of the project is difficult, indeed, intelligent spammers use path preload in such a way that detection proves very difficult.

7.2. Why path preload is used

In conversation with a news server's network administrator[4], it was suggested that path preload may be used for the following reasons:

1. To prevent detection of the spam by appearing to come from a site with a good anti-spam score.

[4] Thanks to Josh Garliardi from Highwinds.com

2. To prevent detection by not routing it to the sites which are known to run aggressive detectors that will emit *cancel*s.

The main effect of the path preload is that the server receiving the spam is not located at the start of the path anymore; instead he will be either at the last entry of the common path or somewhere inside it.

7.3. When is a common path 'path preload'?

There can be legitimate reasons that a common path is present, and a server can not necessarily tell whether this common path is path preload. Some valid reasons for common paths are:

1. The news server posted to has several internal servers which the article travels through before it leaves the organization.
2. The connections through the first few news servers is strongly linked, and therefore it is unlikely that other connections would have been made before these first links were established.

However it needs to be kept in mind that the introduction of the *nntp-posting-host* header has made this technique much less attractive to spammers that work from the 'client' side.

Another point to consider is that path names can be whatever the servers' administrators configure them to be in their server software. Usually, but not always, they are the actual internet domain names of the servers.

It is also quite common for servers to have incoming feeds from more than one server. This ensures the servers get all articles for any newsgroup. So even when you look at postings made on the same server, it is quite possible for them to take different routes to get to your server.

8. Building the Common Path Detection Tool - preparation

The building of the tool required the following steps, elaborated upon below:

1. Connection issues
2. Determining the platform to build the tool on (server or client side).
3. Which programming languages to use.

8.1. Connection issues

Many large organisations, such as universities, have their own news server, while others allow access to the port 119 for NNTP to enable access to other news servers. To verify access to this port, the following commands can be executed from a host (bold type is your input):

```
chrome:~$ telnet my.news.host 119
Trying aa.bb.cc.dd...
Connected to my.news.host.
Escape character is '^]'.
200 my.news.host InterNetNews NNRP server INN
2.2.2 13-Dec-1999 ready (posting ok).

GROUP alt.test
211 232 222996 223235 alt.test

quit
205 Transferred 116 nbytes in 0 articles, 1
group. Disconnecting.
…
```

Replace *my.news.host* with any NNTP server that is freely available. Many news servers require a fee, though lists of open (free) news servers are available on the Internet. One such news server that can be used to check port availability is located at *news.readfreenews.net.*

8.2. Which platform?

The major consideration at the start of this research is deciding whether to install INN news server software and interact with the Usenet as a server, or else interact with the Usenet via a web application on the campus web server. I choose the second option for the following reasons:

1. University firewalls make server connection very difficult, if not impossible.
2. Established INN code would need to be adapted to suit this particular application. E.g. it would be no simple feat to download the same article from different servers since the INN software inherently tries to prevent this.
3. Using a web server as platform has the prime advantage of *agility*, i.e. accessing different servers all over the Usenet becomes easy.

8.3. Choose Programming Languages

The *PHP* language provides ample tools for text processing and parsing. It also has many *socket* functions, using these functions enables one to access the NNTP servers quite easily, assuming that access on the NNTP port 119 has been granted.[5]

Javascript is sufficient to map out all the paths and a *Mysql* database is used to store all data. Capability is given to copy the database to files and vice versa, allowing the storage of whole sessions, the data of which can be used for further analysis in the future.

[5] Appendix: *PHP example code*

9. Using the Path Analyser - method

Refer to *A Screenshot run-through of the Path Analyser Tool* in the appendix to illustrate the stages described below.

9.1. Read Usenet articles.

It will be found that not many newsgroups are spared from spam. At this stage, it is sufficient to *suspect* articles of being spam, not establishing if they are spam. A quick look at the subject of the article often gives a good indication. Some examples:

- `Very pretty Tokyo girls.060403`
- `Get 500 BUCKS FREE!`
- `Viagra - £31.20 for 28 tabs - Today's special`

The easiest way to read articles is by using a news reader. Several choices of news readers are available; the one used in this project is the freely downloadable *SeaMonkey*.

A very effective way to find spam articles is to visit one of the newsgroups where suspected spam is regularly reported. The articles in these groups often carry an exact copy of the spam headers. One group, *newsguy.spam.sightings,* is especially effective, while the more official newsgroup *news.admin.net-abuse.misc* seems less so.

The web-based application developed for this research enables direct download using sockets. This is useful in situations where more *surgical* downloading is necessary, such as portions of the headers of articles in the *alt.binaries.** newsgroups.

To protect the campus's servers against overload (it is quite easy to inadvertently download masses of data using commands such as *head* or *body*), a quasi-server application, such as *Hamster* is used (quasi, because the server function is

implemented on the client). This enables download of all the necessary articles for a whole session, after which the news reader can be used to access particular articles on this quasi-server.

9.2. Record the message ID of suspect spam

The *message-id* can be found from the news readers' headers, by using the Path Analyser Tool's *Gather message ID's*, or by using a combination of these. If binary articles are chosen it is best to just download the headers.

9.3. Choosing suitable Open Servers

There are many web tools available that list open servers. These tools often classify open servers based on certain characteristics such as speed, volume, amount of newsgroups it carries, binary newsgroups availability, posting allowed, etc. Cross-referencing can be done so that, for example, servers can be found that carry many newsgroups and have good throughput.

Some tools add geo-location facilities, and while it may seem preferable to spread the server locations around the world, as stated previously, this doesn't necessarily mean that these servers are distant as regards Internet connectivity on the IP level.

Other web tools try to determine connectivity between servers, drawing a map in the process. However this procedure is very slow, and most of the mapping seems to involve internal peer servers; as soon as an outside server is accessed, the information often becomes unavailable.

The *Gather Paths* page in the Path Analyser Tool provides a quick and easy way to determine whether an open server is accessible for a particular newsgroup. Once some good servers have been determined, it is advisable to use a proper news reader, such as *SeaMonkey*.

9.4. Gather Paths

The next step is gathering paths for a particular article (message ID). The article's headers are automatically downloaded from the open servers determined in the previous stage (typically around 10). Though the body of the article is the same on all servers, some headers, such as the path headers are not.
Once a header is retrieved, some parsing is done, and the results stored in the database. The Path Analyser Tool does all this automatically.

9.5. Displaying and Mapping paths

Having gathered the necessary data, various functions are applied to list all the paths and determine the common path. Option is then given to map these paths. This mapping is done using Javascript. The resulting map provides an immediate picture of the articles' trajectory to the various servers. The common path on this map is indicated by preceding the name of the server with a square symbol. In addition, the servers adjacent to the common path, the *adjacent servers*, are likewise marked. This is done because the adjacent servers will prove important for further analysis.

9.6. Storing data

The data gathered is stored in 3 different ways: the database, a repository, and a spreadsheet.

The database architecture is given in the appendix[6]. The data is organised into five tables, named *Message, TraceMessage, ServerMessage, Path* and *Server*. This facilitates the determination of the common path and the construction of the maps.

The repository holds the following data:

- Actual articles downloaded.
- A session folder which holds copies of the database tables for all the sessions.

[6] Appendix: *Database Architecture*

- A folder storing lists of open servers.
- A folder storing all the maps for each article.
- A text file listing the common paths for each of the articles.

This repository can be used for several purposes:

1. Data from previous sessions can be loaded into the tool for continuation at a later stage.
2. Data from previous sessions can be used for future analysis once more resources become available.
3. Data is used to do current analysis and compare results.

The appendix[7] shows a subsection of the spreadsheet file. Each article has the following data stored:

- Date of the session
- Session ID
- List of open servers used
- Newsgroup the article is spotted on
- Subject of the article
- Message ID
- Posting host header
- NSLookup (resolving the IP given in the Posting-Host header
- Article ID (personal number used for referral in the repository)
- Servers responded (number of servers that responded and therefore present in the common path for this particular article).

Results of the analysis are also recorded in the same spreadsheet. A full discussion on the columns handling this is outlined further on. Each of the columns has several parameters. The main columns are:

- Posting Host information
- Mismatch information

[7] Appendix: *Spreadsheet sample*.

- Message-id information
- Posted stamp information

10. Results of Common Path Research

The data was gathered over a period of 2 months using the method described above. The research was done in 17 sessions. Each session involved finding up to a dozen or so spam articles and using the Path Analyser Tool to download each of these articles from about 8 open servers. Sometimes only one server could be found, sometimes up to 14 servers could be found carrying the article. This constitutes a total download of around 520 headers and construction of some 87 common paths. The repository holds a copy of all these articles and common path maps, for further research. The research can be easily duplicated since the database files for each session is also stored, so all the sessions can be continued by downloading the files into the database and repeating the operations or adding more paths to it. This is assuming that the article still resides on the Usenet and hasn't been *cancelled*.

In gathering results, other factors besides those relating to path preload were taken in consideration. This was useful because it helped to establish an article as spam, which in turn made it more plausible that preload had been utilised. Indeed as will be outlined in the summary, a more general approach will be necessary to confirm path preload to a reasonable degree.

The following factors were analysed:

1. Frequency of occurrence of the *posted* stamp.
2. *posted* stamps not occurring at the start.
3. Frequency of occurrence of the *mismatch* stamp
4. *mismatch* on a *boundary server*[8]
5. Occurrence of *nntp-posting-host* header.
6. *nntp-posting-host* mismatch, using NSLookup.
7. *nntp-posting-host* not resolved.

[8] Refer to chapter 7.1 for definitions of *boundary servers* and *adjacent servers*.

8. *message-id* doesn't contain id of server posted to.

From all the factors listed, only the *mismatch* and *posted* stamps are directly relevant in the quest to connect common path issues with possible path preload. However, the data at the beginning of the path also relates to the common path, be it secondarily, and therefore is also considered for analysis. This type of data involves the *nntp-posting-host* and *message-id* factors.

Following is the result of analysing the data according to the presence and location of the *posted* and *mismatch* stamps.

10.1. posted

One quarter of the articles had a *posted* stamp; none were in the wrong place i.e. all the *posted* stamps were located at the beginning of the path, not on a *boundary server*, which certainly would indicate preload. Though this stamp is not mandatory, this result hints to the fact that if path preload had been applied in any of the samples,

1. the spammer was able to deceive the receiving server into the belief that the article is in transit, not posted to.
2. the spammer was a news server.

10.2. mismatch

I distinguished 4 different types of *mismatch*:

1. NICP - Not occurring in the common path.
2. OB – issued by a *boundary server*.
3. ICP - Occurring in the common path, but not issued by a *boundary server*.
4. AS – issued by an *adjacent servers*.

Note that the fact that this stamp does *not* occur at certain locations may simply mean that the server does not apply this stamp. Therefore it does not conclusively determine that the path entry matches the receiving server's information.

The following discussion uses ideas and definitions explained in chapter 7.1.

Assuming that the common path is path preload then the last server in this common path is the server that the spammer posted to. This last server is referred to as the *boundary server*. If the server preceding this *boundary server* is stamped with a *mismatch*, it indicates that the boundary server has detected that the last server in the preload is wrong. This means the article is very suspicious.

A *mismatch* occurring elsewhere in the common path is still suspicious because it is plausible that the path preload is only a section of the common path, i.e. the last servers in a common path may be strongly connected. In this case, the stamp may be issued by a server that received the spam.

A *mismatch* issued by an adjacent server to the previous entry is also suspicious since it is possible that a spammer posted a path preload to the *adjacent servers* in an EMP type of spam.

The results were:

number of articles	**type of mismatch**	**percentage**
53	NICP	
0	OB	0%
11	As	12.5%
17	ICP’s.	21%

Total occurrence of *mismatch*

mismatch occurred 64 + 17 + 11 times in the 526 articles, which is in 16% of the articles.

Suspicious articles

In conclusion, both the AS and ICP articles should be regarded as suspicious, they constitute respectively 12.5% and 21% of all the articles analysed.

A great improvement to validate this result even further would be to establish which servers apply the *mismatch* stamp (stamping *mismatch* is not mandatory). This could be accomplished by posting an article with a forged header to various servers and noting whether the server applied the stamp to the forged path entry.

11. Further checks on the Common Path

11.1. Establishing server entries in the common path

It became necessary to do more testing on the common path results, such as validating the servers appearing in the common path and adjacent servers.
Though it is possible to verify the existence of servers in the common path, three difficulties arise:

1. The address in the common path may not be a valid domain address, such as *dedekind.zen.co.uk*. Remember this is not a mandatory requirement for path entry. However, in most cases the addresses examined in this research were valid domain addresses or IP addresses.
2. Many aliases exist for a server, i.e. a server may use several names. Aliases are often the cause for the *mismatch* stamps elaborated on above.
3. The address may not be visible by using the networking commands *whois* and *nslookup*.

11.2. Validate connections in the common path

A great help in the validation of the common path is to verify the connections between the servers in the common path and between the *boundary servers* and *adjacent servers*. The Path Analyser has been extended to investigate these connections; it uses the network command *traceroute* to draw routes to some of these servers. The aim was to establish whether some linkages occurring in the common path also occur in the traceroute.

However it soon became clear that not many connections could be verified. The difficulty proved to be that the Usenet connections did not mirror the IP connections in general. After some enquiries on the Usenet regarding this issue, it became clear that using traceroute to establish NNTP connections might have been possible 10 years ago, because the server connections were quite basic then. Nowadays ISP's and networking campuses use several servers to handle the increasing load on the Usenet; one may handle the incoming posts, one handling the outgoing posts, one handling transit, etc… I talked about this earlier in chapter 2.7. While it is true that an administrator may use tracerouting to establish who its news peers will be, it by no means reflects the path a traceroute may take. To summarize then, we can state 2 factors here that explain why a traceroute is not effective:

1. News servers have strong internal linking, not reflected in a traceroute.
2. Connections between news servers' peers are not necessarily mirrored in connections between servers on a general IP level.

To establish whether the servers are connected, a quick message to either server's administrator may confirm or deny this. However, it will be found that network administrators are not particularly keen on revealing such data because it may compromise their security.

In conclusion, the following procedure applied to a suspected path preload seems appropriate:

1. Analyse various headers of an article to establish foul play, or possibility of foul play.
2. Use Path detection tool to confirm the common path.
3. Search for *mismatch* telltale signs.
4. If foul play is strongly suspected, contact administrator to confirm some connections, especially administrators of *boundary servers*.

12. Stages in determining Path Preload

Some additional analysis has been done on the gathered data in the quest of determining whether the article is spam or not. This analysis is done in order to strengthen the belief that the article is spam, in which case path preload is more likely. Some data has been recorded regarding the *posted*, *message-id*, and *nntp-posting-host* headers. The resulting data analysis for each of these headers is outlined next.

12.1. message-id

Note that in RFC 2822, the *message -id* is expected to contain the domain address or IP address of the host the article is posted to:

> NOTE: Section 3.6.4 of [RFC2822] recommends that the <id-right> should be a domain name or a domain literal. Domain literals are troublesome since many IP addresses are not globally unique; domain names are more likely to generate unique Message-IDs.(Lindsey 2006).

I distinguish between 3 types of *message-id*'s:

1. OK Posting server's identity is included in the *messsage-id*.
2. NM No Match with the posting server.
3. PM Partial match with the posting server's identity.

To verify the posting server's address in the *message-id*, it is compared with the first server entry in the common path.

The results out of 76 original articles were 6 PM's, 8 NM's, 61 OK.
The 6 PM's are acceptable since a perfect match is not required; however the 8 NM's arouse suspicion. This occurred in 11% of all the articles.

12.2. nntp-posting-host

Four categories were established regarding this header

1. OK – Match with server's address found in the *message-id* or the posting server entry in the path (often stamped with *posted*).
2. NM – No match with the server's address found in the *message-id* or the posting server entry in the path (often stamped with *posted*)
3. NR – Couldn't resolve the given IP using *nslookup*.
4. NO – Header was not present.

Out of 79 articles, 10 were OK, 28 were NM, 11 NR's, and 30 NO's.

These results indicate that 37% of the articles do not have a *nntp-posting-host* header. The fact that no header is present does not indicate spam, since the server is not required to add this header.

This header is only useful once the article has been confirmed spam, in which the spammer can be contacted directly instead of the server it was posted to. However, it can be argued that articles that could not be resolved should be suspicious, even though the address may have been legitimately kept hidden.

Again it would greatly enhance the research to be able to establish which servers apply the *nntp-posting-host* header.

13. Results of Path Preload Research

To summarise the results (the numbers are the actual article numbers which are included here for the purpose of analysis):

mismatch
AS: 24, 26, 28, 33, 34, 56, 61, 64, 74, 86, 87
ICP: 28, 47, 69, 71

message-id
NM: 5, 7, 8, 22, 48, 58, 65, 67.

nntp-posting-host
NR: 20, 41, 42, 48, 64, 76, 81, 86, 87, 88

The highlighted articles occur in more than one location, i.e. for such articles there are 2 suspicious occurrences in the header, at least one of which involves a particular placing of *mismatch* in the common path. For this reason it can be safely assumed that the likeliness of path reload has been enhanced. These articles are:

Suspicious articles: 28, 64, 86, 87

Using the criteria outlined so far, the ratio of articles strongly suspected of having path preload is 4 out of 76, a ratio of **5%** of all the articles suspected of being spam.

13.1. Discussion on results

Though a higher probability for path preload has been established in certain articles by combining suspicious location of *mismatch* in the common path with doubtful entries in other headers, it must be said that it is difficult to quantise the degree to which these articles are path preload. Therefore the analysis of these results should be treated with caution and be regarded as a preliminary step towards establishing path preload with reasonable certainty.

It needs to be remembered that these results are build on a relatively small volume of data, even though some effort has gone into ensuring that the data was gathered from a wide variety of open servers on many different newsgroups.
Notwithstanding this, some interesting results have been obtained, especially the finding that there seems to be some correlation between suspicious *mismatch* locations in the common path and some unusual *message-id* and *nntp-posting-host* entries.

Another finding is that possibly the best header to cross-reference with the common path data is the *mismatch* stamp, which occurred once in every 6 articles.

In this analysis, the use of the common path information has only been marginal, since it was only used in combination with *mismatch*. The results therefore are only an initial investigation on how the common path might be used. In chapter 14, *Likely improvements*, some additional ideas will be covered.

14. Difficulties faced

The first stage in this research involved coming to a general understanding regarding all the facets of the Usenet. Following that LINUX had been installed on the home computer with INN to get some understanding of how the INN server software works. At university it became clear that a server couldn't be used because of the firewall restriction, this turned out to be advantageous because building a tool on the web server gave me more agility.

The next stage of the research involved installing a 'semi' news server (Hamster) and a news reader (seaMonkey) on the university computer, in order to be able to read some news articles. This required me to access port 119 which demanded some changes in the switch configuration at the campus.
Building the tool took a lot of time, but didn't really involve any great difficulties. The hardest part regarding the development of the tool was possibly working out how to map the paths; a fair bit of research went into this before I finally decided on a classifying system, written in Javascript.

The greatest obstacle was possibly the verification of the connections in the common path and between servers at the end of the common path and the first adjacent servers. One of the problems that makes verification difficult is the complicated fashion in which today's news servers share their tasks; the transit servers and reader servers both execute different commands thereby affecting the headers in different ways.

Some attempt was made to reveal these connections by extending the tool with a *tracerouting* facility. As stated earlier, I was forced to take a different approach, combining the mapping of the common path with other features in the headers of the suspected spam articles, the results of which proved worthy of interesting.

15. Likely Improvements

The following ideas were conceived during this research and may be taking into consideration for future development:

1. Find alternative ways to establish server connections in the common path. One possibility is to post articles to different newsgroups to both servers (that are apparently connected according to the common path information), and establish the frequency of contact between both servers, if at all. This can be accomplished by using a Usenet Control Client such as Newsagent or by extending the Path Analyser. Various interesting data may be gathered by observing and notifying the behaviour of both servers. The last cause of action in verifying connections should be the notification of the relevant administrator.

2. Use data from other headers, such as *x-trace*. Indeed, comparing data from other headers is very revealing, however, this requires expert knowledge and may be better handled by AI applications that specifically handle and analyse data in a heuristic way.

3. Accumulate large volumes of data in order to gain more accuracy in the analysis and to allow for more complex systems of classification.

4. Expand the database with alias information for each server that has aliases. This list can be updated and verified by posting to them regularly.

5. Consider mapping the common path of articles that are similar but don't have the same message ID, i.e. articles posted using the EMP type of spam.

6. Sending forged headers may lead to some insights as to how servers react to path preload, whether they stamp the path headers, which headers they enforce, etc. All this can be recorded in the database.

7. Download many articles by the same spammer and draw connections. This may establish whether the same path preload exists in other articles also. This involves many more downloads, ideally using data from established Usenet servers.

16. Summary and Conclusion

The Usenet has a rich, but brief, history that is fascinating since it spans the era from pre-worldwide web days up to now. Its technology has its humble beginnings in the UNIX environment and today there are still visible traces of this early technology present, such as the way in which headers are formatted.
Two of the most interesting aspects of the Usenet are the way in which messages are propagated using flooding techniques, and Usenet's means of addressing using *newsgroups* (including *cross posting*).

Usenet's networking system originally used the UUCP transport protocol and eventually moved to the *news net transfer protocol* (NNTP), a type of TCP protocol derived from SMPT, to enable transporting Usenet messages over the Internet.
Two RFC's were devised, one describing NNTP, the other one describing the way Usenet exchanges its messages.

The headers of the transported messages can be analysed. The one that interests us most is the *path* header. Each time a message passes through a server, the server adds its identity to this path header. A common technique used by spammers is to pre-empt this path with a false path section, this technique is called *path preloading*. A path preload makes it difficult to trace the server the spam is posted to. The aim of this project is to use the fact that this path preload section will be common to all headers of the same spam located on different servers.

Various tracing tools already exist to enable tracing through the nodes, but we need a more semantic view of how Usenet traffic flows between the nodes. Only by knowing this traffic flow will it be possible to properly analyse path preloads since we then can identify likely routes taken in the network. However it is difficult to get hold of this type of data. Usenet traffic flow maps exist (as shown in this paper) but, to my knowledge, haven't been updated since 1993.

Additional to addressing the issues above, this paper also covered more general issues regarding spam, its history and techniques, and other types of Usenet mapping.

The core of the research involved the exploration of a technique which could be labelled as the *Common Path Technique*. A great deal of time has gone into coming to grips with the various aspects governing the Usenet and the implementation of the Path Analyser Tool which enabled me to construct *common paths* from articles suspected of spam. Analysing a relatively small volume of data gathered with this tool provided me with some valuable insights which were encouraging.

The tool has also proved very useful for issues besides mapping common paths. The strength of the tool lies in its agility, i.e. the ability to automatically download sections of headers from different open servers in a ‘surgical’ manner.

It was established that there are difficulties in verifying the connections between the servers in the common path and the immediate adjacent servers - difficulties but not impossibilities – indeed, as a last resort it is possible to contact the administrators, though some trust may need to be built first.

The strongest association between the common path and the headers of an article proved to be the *mismatch* stamp that occurs in the path header. This in combination with other anomalies in the header led me to the conclusion that there is a suspicion that about 5% of the suspect spam messages use path preload, though I emphasized that this result should be approached with caution.

Lastly, some ideas were offered, some of which are extensions of the current approach and some taking a completely new direction.

In conclusion, the common path technique proved to be a valuable and important first step in the quest for detecting path preload and it is easy to envisage further improvements, either by means of extending the current tool or combining the tool

with other spam detection techniques. I suggest that a combination of these improvements, together with an ‘expert system’ approach, using heuristic techniques will prove to have the best change of success in the future, taking us one step closer in the fight against spam.

17. Supplementary

17.1. PHP example code

Following code illustrates the basic steps of connecting to an NNTP server and getting the number of articles available on a particular newsgroup. The NNTP command GROUP is given to the NNTP server and its reply message parsed.

```
<?php

$cfgServer    = "news.readfreenews.net";
$cfgPort    = 119;
$cfgTimeOut    = 10;

// open a socket
if(!$cfgTimeOut)// without timeout
    $usenet_handle = fsockopen($cfgServer, $cfgPort);
else             // with timeout
    $usenet_handle = fsockopen($cfgServer, $cfgPort,
&$errno, &$errstr, $cfgTimeOut);

if(!$usenet_handle)
{
    echo "Connection failed\n";
    exit();
}
else
{
    echo "Connected\n";
    $tmp = fgets($usenet_handle, 1024);
}



//$cfgUser    = "xxxxxx";
//$cfgPasswd    = "yyyyyy";
$cfgNewsGroup    = "alt.php";

// identification required on private server
if($cfgUser) {
    fputs($usenet_handle, "AUTHINFO USER
".$cfgUser."\n");
    $tmp = fgets($usenet_handle, 1024);

    fputs($usenet_handle, "AUTHINFO PASS
".$cfgPasswd."\n");
    $tmp = fgets($usenet_handle, 1024);
```

```
    // check error

    if($tmp != "281 Ok\r\n") {
        echo "502 Authentication error\n";
        exit();
    }
}

// select newsgroup

fputs($usenet_handle, "GROUP ".$cfgNewsGroup."\n");
$tmp = fgets($usenet_handle, 1024);

if($tmp == "480 Authentication required for
command\r\n") {
    echo "$tmp\n";
    exit();
}

$info = split(" ", $tmp);
$first = $info[2];
$last = $info[3];

print "First : $first\n";
print "Last : $last\n";

?>
```

The result of executing the above code on a server will resemble the following line:

```
Connected First : 54832 Last : 59086
```

This message shows the NNTP server's ID's of the first and last articles available on the newsgroup *alt.php*.

17.2. 2006 Usenet Statistics

Following statistics give some insight as to the current state of the Usenet server and hosts as regards spam (News.admin 2006).

The top 10 spam hosts:

1	Active-news.com
2	Astraweb.com
3	United-newsserver.de
4	t-ipconnect.de
5	ish.de
6	newsreader.com
7	xsnews.nl
8	io.com
9	col.ru
10	ntl.com

Some other server, encountered frequently in this research, and therefore of interest are:

11	corenews.com
19	freenet.de
23	usenetMonster.com
24	tiscali.com
27	aioe.org
94	eutelia.it

The top 10 spam sites are:

1	Readnews.com
2	Astraweb.com
3	Giganews.com
4	Usenethost.com
5	Chello.at
6	216.74.57.0
7	United-newsserver.de
8	News-service.com
9	Titannews.com
10	Teleline.es

Servers of interest to us are:

27	Googlegroups.com
62	freenet.de
70	earthlink.net
76	usenetmonster.com
77	blueyonder,co.uk

93	thundernews.com
94	prodigy.com
94	aioe.org
99	gradwell.net

Percentage of Newsgroups uniquely accessed

Virtually all 100 are in the alt.binaries.* groups.

First 10 are:

1	alt.binaries.dvd.music
2	alt.binaries.dvdrcore
3	alt.binaries.multiomedia.erotice.voyeurism
4	alt.binaries.dvd.erotica
5	alt.binaries.x
6	alt.binaries.multimedia.erotica
7	alt.binaries.pictures.erotica.anime
8	alt.binaries.pictures.erotica.male
9	alt.binaries.cd.image
10	alt.binaries.erotica

Another web page lists the top 20 servers for 2006 (Top1000.org 2006)

A list of servers is presented which relies solely on servers that send their statistics on a daily / weekly basis. It does not reflect actual Usenet distribution.

Rank	Weight	Pathname
1	27.652	nntp.giganews.com
2	15.430	news.usenetserver.com
3	14.443	news.glorb.com
4	14.119	news.astraweb.com
5	13.972	feed.tweaknews.nl
6	12.972	newshosting.com
7	12.604	proxad.net
8	11.483	feeder.xsnews.nl
9	11.222	newsrouter-eu.astraweb.com
10	11.026	eweka.nl
11	11.024	border1.nntp.dca.giganews.com
12	10.348	border1.nntp.ams.giganews.com
13	10.260	local01.nntp.dca.giganews.com
14	10.148	news-out.readnews.com
15	8.847	news.highwinds-media.com
16	8.778	news.tele.dk
17	8.691	border2.nntp.dca.giganews.com
18	8.296	feeder1.cambrium.nl
19	8.190	newsfeed.freenet.de

17.3. Usenet articles on path preload

Some Usenet newsgroups are forums discussing Usenet spamming. The *news.admin.net-abuse.usenet* newsgroup proved most valuable in this regard. Following are some excerpts that had impact on this research:

> I've seen loads of instances involving highly complex path preloads, intended to obfuscate the actual origins of posts and rogue cancels. Such preloads have also been used in some cases, as part of rather lame attempts at avoiding detection at various sites associated with specific spammers.
>
> You can search groups.google.com for path discussions involving Jerry "SpamZilla" Reynolds <http://tinyurl.com/hapz>, Matt Middleton <http://tinyurl.com/haq4> and Bruce Becker <http://tinyurl.com/haqa>, to get a better picture. These are only a few examples. (Both Reynolds and Middleton were able to forge a wide variety of headers in their posts, as they were running their own dedicated spam servers to inject their spam directly into the news stream. The same is true of Jim Buh's spam operation.)

After posting an article to the newsgroup *news.admin.net-abuse.usenet* trying to get some information for my research, one useful reply was the following article:

> You need to download a copy of Hipcrime's fabulous software and test it by sending multiple messages to multiple groups using various open NNTP servers. You are perfectly within your right to do so because of your 1st Amendment Rights.

Some posters even illustrate preload in the header of their articles:

> Message-ID: dritz-B7DF74.22550117072003@news.supernews.com
>
> Path:archiver1.google.com!news1.google.com! newsfeed.stanford.edu! cyclone.bc.net! sjc70.webusenet.com! news.webusenet.com! sn-xit-02! sn-xit-06! sn-post-01! supernews.com! news.supernews.com! i.put! this.here! dritz

17.4. Header example

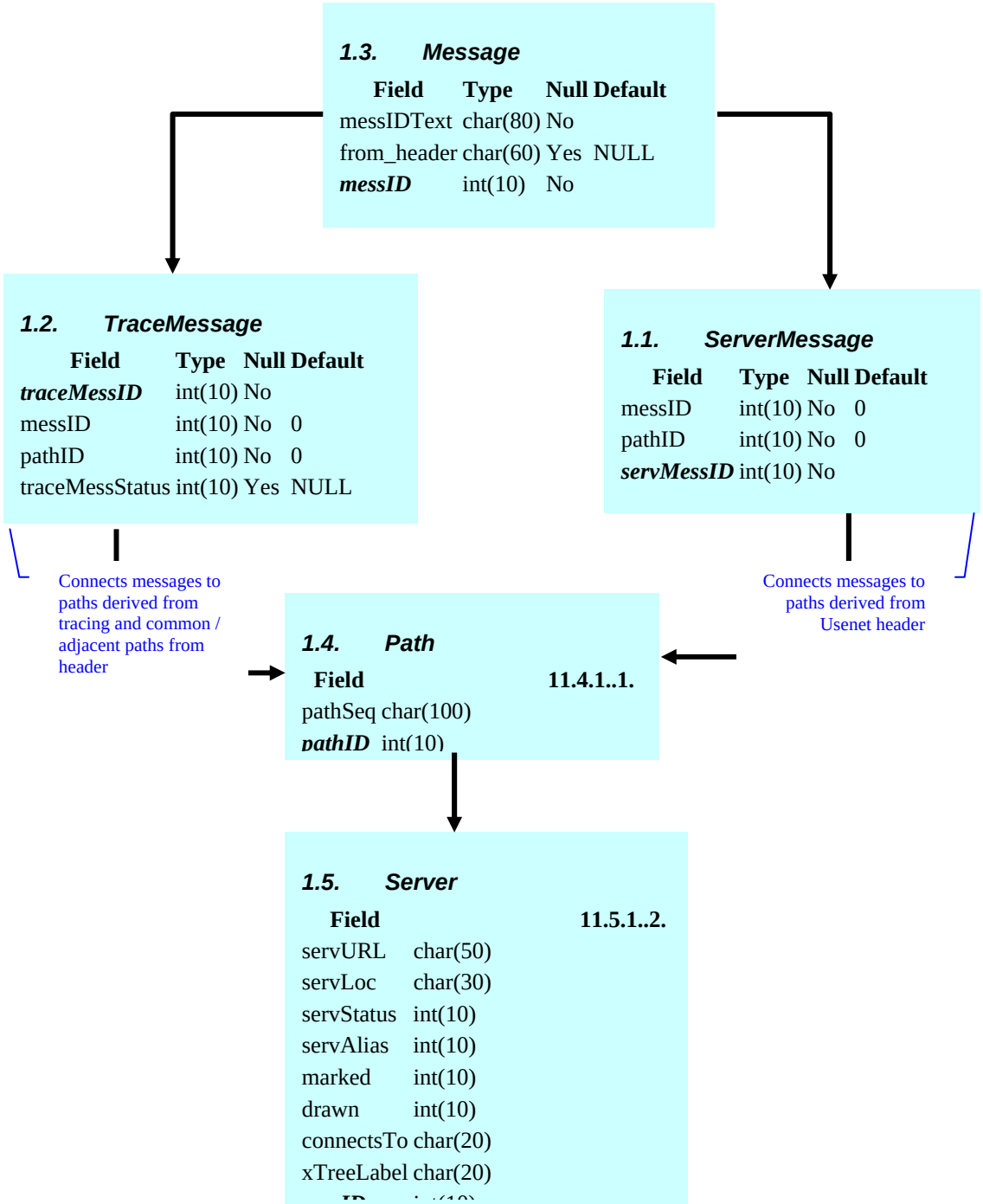
Following example shows the header of an article (article 40). Note the long list of newsgroups in the *Newsgroups* header.

```
Xref: localhost misc.health.infertility:38
Subject: FREE Diet Sample Pack
From: "YouCanReallyLoseWeight" <kdsprod@comcast.net>
Date: Mon, 20 Mar 2006 13:49:07 -0500
Message-ID: <1142880390_2897@sp6iad.superfeed.net>
Lines: 106
Path: authen.yellow.readfreenews.net!green.octanews.net!news-
out.octanews.net!news.glorb.com!usenet.INS.cwru.edu!news-
out.nuthinbutnews.com!sp6iad.superfeed.net!sp6iad.superfeed.ne
t!not-for-mail
Newsgroups:
misc.health,misc.health.aids,misc.health.alternative,misc.heal
th.alternatives,misc.health.arthritis,misc.health.diabetes,mis
c.health.infertility,misc.health.injuries,misc.health.injuries
.rsi,misc.health.injuries.rsi.misc,misc.health.injuries.rsi
X-Proxy-User: $$no5i9e50zlby
MIME-Version: 1.0
Content-Type: multipart/alternative;
 boundary="----=_NextPart_000_0286_01C64C25.0F178B60"
X-Priority: 3
X-MSMail-Priority: Normal
X-Newsreader: Microsoft Outlook Express 6.00.2900.2670
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2900.2670
X-Report:  Please report illegal or inappropriate use to
<abuse@newsfeeds.com>. Forward a copy of ALL headers INCLUDING
the body. (DO NOT SEND ATTACHMENTS)
X-Comments2:  IMPORTANT: Newsfeeds.com does not
condone,support,nor tolerate spam or any illegal or
copyrighted postings.
X-Comments:  This message was posted through Newsfeeds.com
X-Old-Xref: authen.yellow.readfreenews.net
misc.health.aids:131249 misc.health.alternative:546141
misc.health.arthritis:59688 misc.health.diabetes:350034
misc.health.infertility:120586
misc.health.injuries.rsi.misc:15961
```

17.5. Database architecture

This is the architecture of the database used by the Path Analyser tool.

This chart lists the important columns only, in reality additional columns are used for data analysis and mapping.

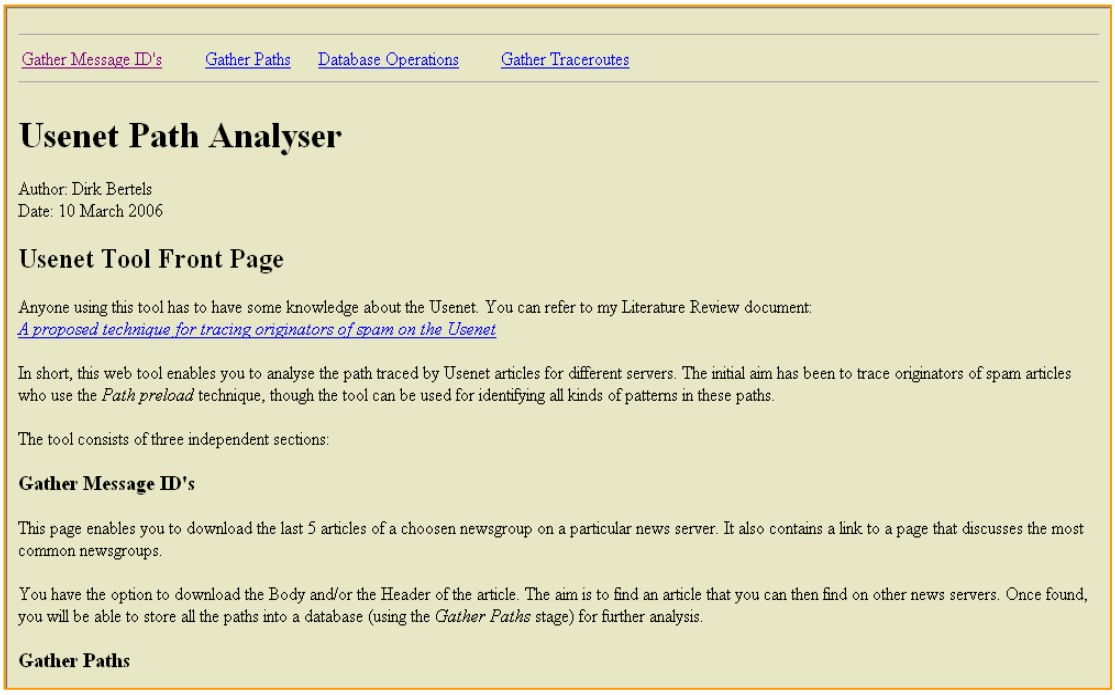


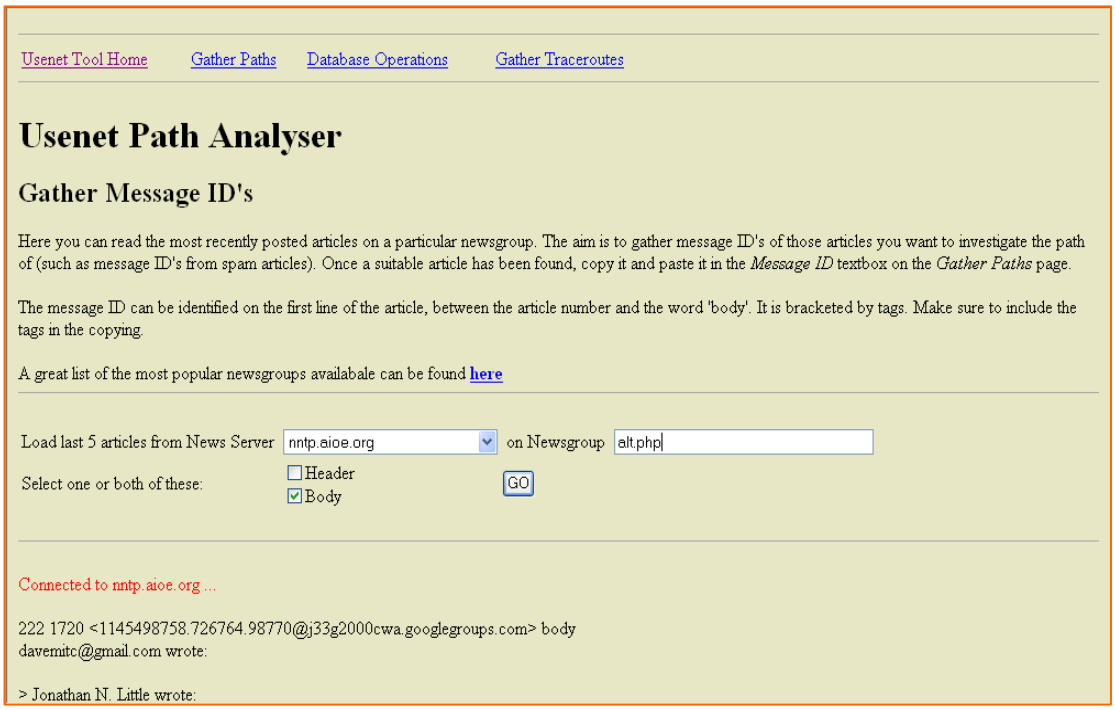
17.6. A Screenshot run-through of the Path Analyser tool

Front page

Explain the different functions of the tool.

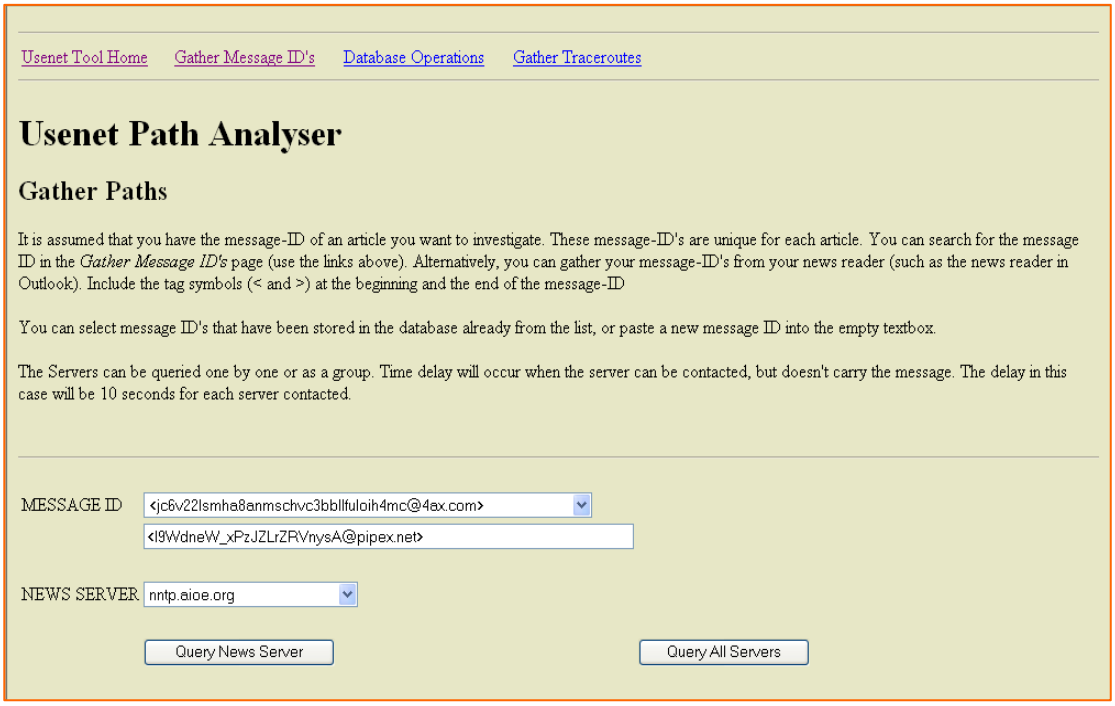
Give links to various other tools available on the web.





Gather Paths page

Automatic download of the same article from different Open News servers



Excerpt of the results of the above operation

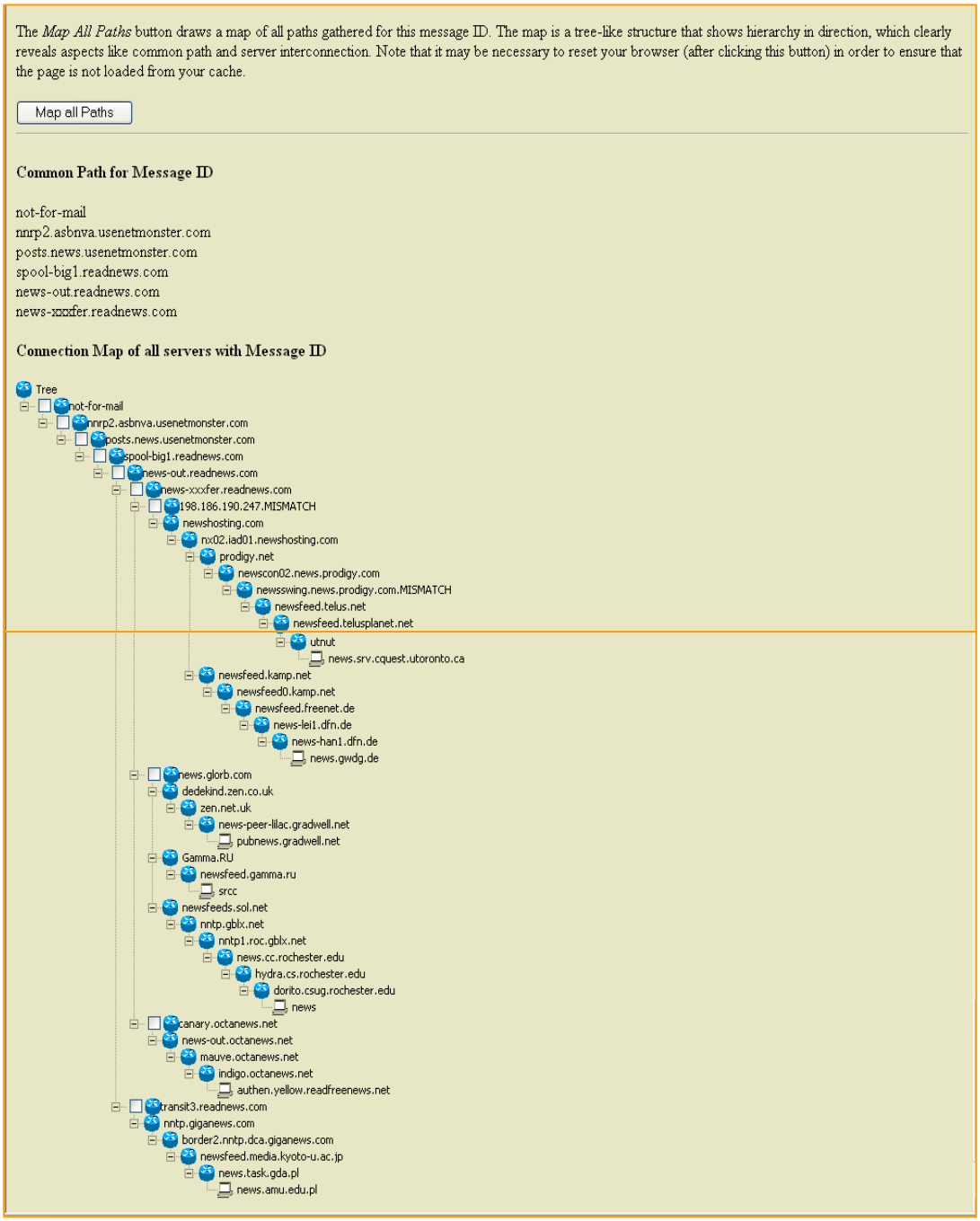
Database Operations page

Reset the database with a list of open news servers.

Save all database information into files thus creating a session that can be downloaded and continued at a later time.

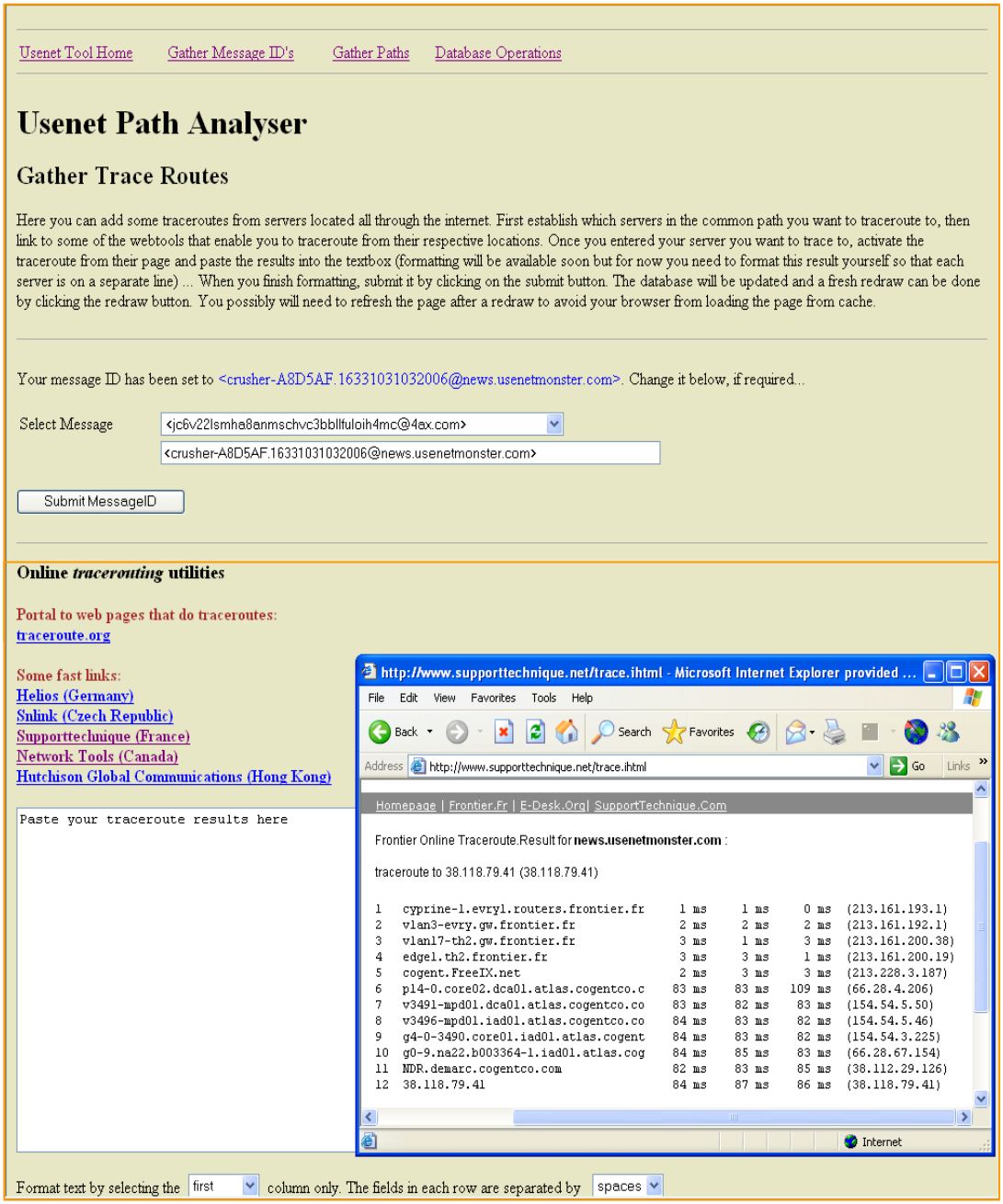
Show all paths (from all open servers that carried the message) for a particular article.

Map all the paths for a particular article (mapping version of the above).

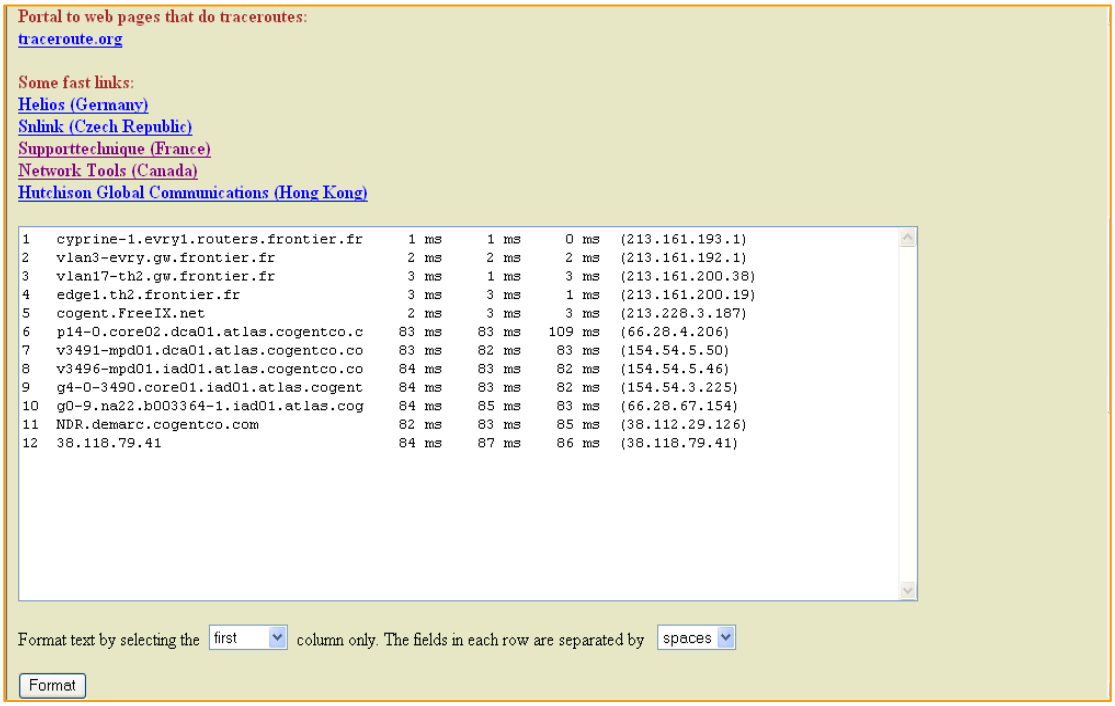


Gather Traceroutes page

Add new traceroute paths to the common path (shown in the illustration above) and the adjacent servers. Both of these are marked with a checkbox.



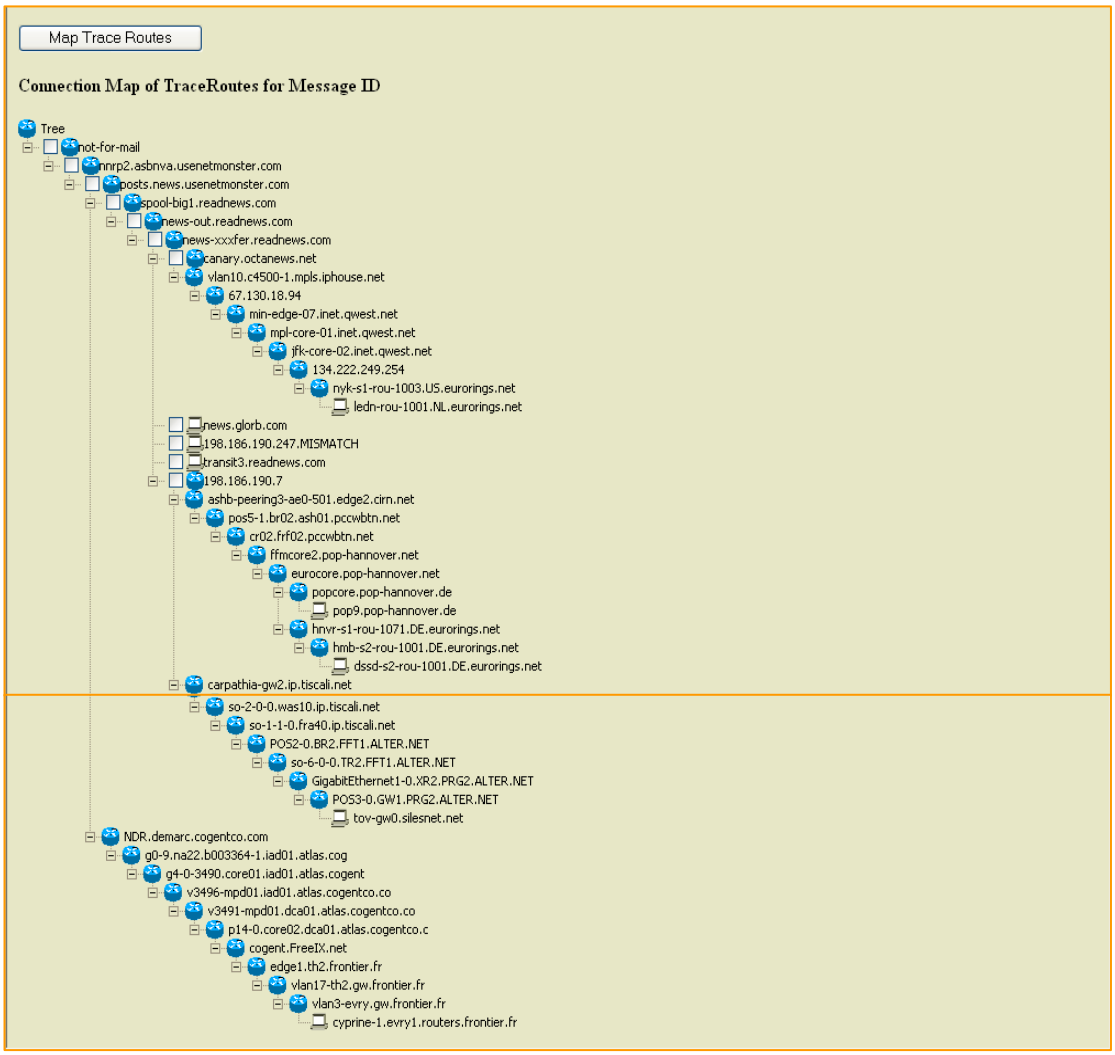
Paste the result in a textarea box.



Format the results.

Submit the result to the database

And create a map of the traceroutes



17.7. Spreadsheet sample

Excerpt from the spreadsheet file. Some columns are compacted to enable an overview of the recorded data.

Open Server			Article						Server	Analysis				
Date	Session	List	Newsgroup	Title	Message ID	Posting Host	NSLookup	Article	resp.	P.H.	M.M	M-ID	X-T	Posted
27/03/2006	S1	L1		Look At this	jS9Vf.233336$Q22.77644@fe1	82.42.38.21	82-42-38-21.cable.ubi	A1.txt	6	ok	no	ok		ok
	S2	L1		become a dotcom millionaire	9k6Qf.106857$8d1.96678@res	24.141.179.155	d141-179-155.home.c	A2.txt	4	ok	no	ok		ok
				Please read this, nothing nas	LmzOf.90173$Q22.62625@fe1	62.30.197.221	62-30-197-221.cable.r	A3.txt	4					
					JulUf.6350$Mj.4055@twister.n	4			4	no	no	ok		ok
					230320061943492754%info@p	5			3	no	1 NICP	NM		no
					HU4Kf.19447$DM.9981@fe3.n	6			1	no	1	ok		ok
					001601c64dd3$e4007067$de1	7			4	no	no	NM		no
					001601c64dd3$e3fdff67$b7b3c	8		NOT	3	no	no	NM		no
					9yOQf.85998$8d1.85973@reac	9			3	no	no	ok		ok
31/03/2006	S3		soc.motts	http://www.votejohntobin.c	t16bqvtb6xf.fsf@nestle.csail.m	nestle.csail.mit.edu								
	S4		soc.man	niggercrusher	4428bfc0$3$22853$6d36acad@	11		SAVED	5	no	2 NICP	PM		no
					4428c022$0$18319$6d36acad	12			5	no	1 NICP	PM		no
					4428bf92$0$19435$6d36acad@	13			5	no	no	PM		no
					4428a7fc$0$22963$6d36acad@	14			5	no	1 NICP	PM		no
					4428a733$0$18319$6d36acad@titian.nntpserver.com				4	no				
					4428a643$0$20689$6d36acad	16			5	no	no	PM		no
	S5		rec.misc	Factory pressed dvd movies,	1136522259.989944.210470@z	218.111.9.251	CAN't Find	A20.txt	2	NR				
			rec.nude	18+ ALL BOYS UNDERWEA	1143390801.945638.298810@z	24.13.191.37	c-24-13-191-37.hsd1.	A21.txt	10	NM				
	S6	L2	rec.guns	Re: Become a published gun	e07c8k$fm5$1@grapevine.war	sniper.cs.umd.edu	128.8.128.79	A22.txt	14	NM	2 NICP	NM		no
	S7	L3	alt.sex	*** FREE Video Download *	1143675803.043090.53050@w	75.3.119.175	adsl-75-3-119-175.dsl	A23.txt	4	NM	no	ok		no
			alt.sex	100% free sex movies and pic	442b261f$3$15316$ba620e4c@	news.skynet.be		A24.txt		no	OB	PM		no
			alt.sex	FIRST CHOICE PHONE SEX	1143661927_13133@sp6iad.	superfeed.net		A25.txt		no	1 NICP	ok		no
16/04/2006	S8		alt.spam	$25,000.00 WEEKLY INCOM	aNrSf.14524$fd.10406@read2	24.141.179.155	d141-179-155.home.c	A26.txt	8	ok	1 OB, 1 NICP	ok		ok
	S9	L4	alt.forsale	$30,000 INCOME EVERY DA	KUIWf.49375$fd.26413@read2.c	24.36.181.48	d36-181-48.home1.cg	A27.txt	5	ok	2 NICP	ok		ok
			alt.forsale	FS: Five PRINCIPLES of Busi	1143665357.11747.0@lotis.uk.	clara.net		A28.txt	5	no	1 OB, 1 ICP	ok		no
			alt.forsale	Stab motor, Free deliveries, 1	4tGWf.2369$m35.166276@ne	70.52.193.92	toronto-HSE-ppp4238	A29.txt	5	NM	1 NICP	ok		ok
			alt.forsale	FREE Work At Home Job Fin	1143592060.478264.24050@z34g2	202.159.21.166		A30.txt	5	?	2 NICP	ok		no
			misc.jobs.misc	Start earning money today by	e0enc5$52v$1@ctb-nnrp2.saix.ne	dsl-165-126-26.telkoma	dsl-165-126-26.telkon	A31.txt	12	NM	1 NICP	ok		no
			misc.jobs.misc	Get 500 BUCKS FREE!	1143381925.754509.98730@e56g2	24.231.71.210	cable-24-231-71-210.l	A32.txt	12	NM	no	ok		no
			misc.jobs.misc	FREE Work At Home Job Fin	e096tg$q18$1@ctb-nnrp2.saix.ne	dsl-165-217-74.telkoma	dsl-165-217-74.telkon	A33.txt	13	NM	1 OB	ok		no
			misc.jobs.resume	** Professional Reference *	1143588491.345836.120490@z34g	130.13.254.85	VDSL-130-13-254-85.	A34.txt		NM	1 OB	ok		no
26/04/2006	S10	L5	misc.immigration.e	EASY WAY TO MOVE TO	<SMydnYECal2keLfZRVn_wA@	giganews.com		A35.txt	11	no	1 NICP	ok		ok
		PATH D	alt.sex	~:~:~ W W W . K I N K	<44307249$0$27404$882e0bbb@	news.ThunderNews.com		A36.txt	9	no	1 NICP	ok		no
		PATH D	alt.sex	:)(WE DO IT ALL ~ EVE	<44305ba5$0$27440$882e0bbb@	news.ThunderNews.com		A37.txt	9	no	1 NICP	ok		no
			alt.sex	<<Very pretty Tokyo girls.(	<1144024774.191902.253430@	58.5.96.102	58x5x96x102.ap58.ftt	A38.txt	8	NM	1 NICP	ok		no
			alt.sex	KATRINA loves YOUNG B(	<1144010710_2319@sp6iad.	superfeed.net		A39.txt	8	no	1 NICP	ok		no
			misc.health.infertil	FREE Diet Sample Pack	<1142880390_2897@sp6iad.	superfeed.net		A40.txt	5	no	no	ok		no
			talk.politics.theory	OFUME FAMILY IN DANG	<1143320708.226664.61830@	69.16.50.209	Can't find	A41.txt	11	NR	1 NICP	ok		no
			talk.politics.theory	DR. PHILLIP C. OFUME, N	<1143920136.257741.105230@	69.16.50.209	Can't find	A42.txt	12	NR	1 NICP	ok		no
			alt.tv	MYSTERY SCIENCE THE/	<1141594227.931587.79310@z	206.149.132.29	29-pool1.ras02.den01	A43.txt	7	NM	1 NICP	ok		no
			alt.health	THE BEST KEPT SECRET	<xfidnUulXbvxq7LZnZ2dnUVZ_	24.35.110.209	cmu-24-35-110-209.m	A44.txt	7	ok	no	ok		ok
			alt.binaries.picture	Free anal sex movies	<716895832606@news.planet	80.61.127.249	ip503d7ff9.speed.plar	A45.txt	0					
			alt.jobs	US-CA-SAN JOSE - VP, C	<KdNWf.127012$G42.89472@fe7	6.usenetserver.com		A46.txt	7	no	1 NICP	ok		ok
3/05/2006	S11	1 server	alt.women.petite	Cd-r Verbatim 0,17 € Dvd \	<bTR4f.20552$o8.13072@twis	85.137.12.174	174-12-137-85.user.a	A47.txt	1	ok	1 ICP, 1NICP	ok		no
			alt.women	photos	<mn.cb497d63ad3a2a2c.5078	5.242.179.199	can't find	A48.txt	3	NR	1 NICP	NM		no
			alt.forsale	ViewsatXtreme2000, Free (	<V4lXf.3357$u15.527671@nev	70.52.193.92	toronto-HSE-ppp4238	A49.txt	11	NM	2 NICP	ok		ok
			news.admin.net-at	Amazing Young TEEN Gal	<dvhu76$vh3$2230@news.nive	pc-11-180-86-200.cm	200.86.180.11	A50.txt	9	NM	1 NICP	ok		no
			alt.binaries.picture	1	<duafgu$3ra$2@ss405.t-com.	83-131-173-48.adsl.n	83.131.173.48	A51.txt	1	ok	no	ok		no
			a.a.mis.talk	Turn $6 into $60,000	<B_udnWSFqKE1ienenZ2dnU	66.218.12.152	dial-66-218-12-152.us	A52.txt	2	ok	no	ok		ok
			alt.binaries.picture	Azz Day	dvp3pe$ga7$1@ctb-nnrp2.saix	ndn-ip-nas-1-p442.tell	155.239.193.186	A53.txt	1	NM	no	ok		no
			alt.gambling	WIN @ WWW.COMETSC.	1144035598.832483.269730@	198.142.196.108	sbrax4-108.dialup.opt	A54.txt	11	NM	no	ok		no
			alt.gambling	$10 FREE! plus 150% Mat	1143880694.795177.108200@	88.1.239.134	134.Red-88-1-239.dyr	A55.txt	13	NM	1 NICP	ok		no
			alt.gambling	platinum play free NEW BO	1144009774.515606.47400@j	83.57.147.216	216.Red-83-57-147.dy	A56.txt	7	NM	1 OB, 1 NICP	ok		no
			alt.gambling	$10 free no purchase i win	1143969547.362302.154350@	83.57.147.216	216.Red-83-57-147.dy	A57.txt	12	NM	no	ok		no

18. References

Baker, S. M. (1997), Tips for avoiding Spam, accessed: April 2005, www.newsrobot.com/sbnews/nospam.html.

Bray, H. (2005). Somehow, Usenet lumbers on. Boston Globe. Boston,

Bulk-News (2002), Bulk News web site, accessed: August 2005, http://www.softwarevault.com/Newsreaders/Bulk-News-2002.xml.

Cotse.Net (2004), List of typical newsgroup headers and their significance., accessed: Sept 2005, http://www.cotse.net/privacy/newsgroup_header.htm.

Dodge, M. (2001), Flowing from Site to Site. Mappa Mundi.May 2001.

Dodge, M. (2001), MIDS maps the Internet world. Mappa Mundi.May 2001.

Emerson, S. L. (1983), USENET: A bulletin board for UNIX users. Byte.219, 1983.

Hayes, B. (2003). "Spam, spam, spam, Lovely spam." American Scientist **91**(3): 200-204.

Hewlett-Packard (2002). Planning a News Server using Internet Express. Tru64 UNIX Best Practice. Hewlett-Packard Company,

Horton, M. and R. Adams (1987), RFC 1036 Standard for Interchange of USENET Messages, accessed: 12 Aug 2005,

Hunt, C. (2002). TCP/IP Network Administration. TCP/IP Network Administration. O'Reilly, O'Reilly**:** 64.

Ingvoldstad, J. (2001). Handling Information Overload on Usenet - Advanced Caching Methods for News. Department of Informatics. Oslo, UNIVERSITY OF OSLO**:** 122,

ITPro (2005), Spam Abuse. IT Pro.January / February 2005.

IWF (2004). Review of Usenet activity, Internet Watch Foundation. **14**,

Kantor, B. and P. Lapsley (1986), Network News Transfer Protocol, accessed: 12 Oct 2005, http://rfc.net/rfc977.html.

Lindsey, C. H. (2006), News Article Architecture and Protocols, accessed: 12 April 2006,

Lueg, C. and D. Fisher (2003). From Usenet to CoWebs: Interacting with social information spaces., Springer verlag.1-85233-532-7.

Marjie (1999), Tracing through a usenet header, accessed: 01 Sept, http://home.att.net/~marjie1/usenet.htm.

News.admin (2006), Usenet Statistics, accessed: 26 March 2006, http://www.newsadmin.com/cgi-bin/newsspam1.

O'Brien, J. C. a. J. (2003), An analysis of spam filters. WORCESTER POLYTECHNIC INSTITUTE**:** 80

Palme, J. (2000). How the Usenet News Protocols Work. Electronic Mail.

Patterson, J. O., M. Anderson, et al. (2003). Spam, Spam, and Spam: A Battle on Many Fronts. Computer Science Seminar, University of Minnesota, Morris.

Quarterman, J. S. and J. C. Hoskins (1986). "Notable computer networks." Commun. ACM **29**(10): 932-971.

Quaterman, J. S. (1990). The Matrix: Computer Networks and Conferencing Systems Worldwide, Digital Press

Sack, W. (2000). Conversation Map:
A Content-Based Usenet Newsgroup Browser. MIT Media Laboratory,
wsack@media.mit.edu

Saiedian, H. and J. Winslade (1992). An interface for acquisition, manipulation and distribution of mail and news messages as objects Proceedings of the 1992 ACM/SIGAPP Symposium on Applied computing: technological challenges of the 1990's Kansas City, Missouri, United States ACM Press**:** 68-74

Saito, Y. (1998), A Usenet Performance Study, accessed: September 2005, powerpoint slides.

Scragg, P. (2002), Network Operations - Tracking Newsgroup Postings, accessed: 20 Feb 2006, http://www.network-operations.freeserve.co.uk/news/index.htm.

Sit, E., F. Dabek, et al. (2004). UsenetDHT: A Low Overhead Usenet Server. MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, Springer-Verlag Berlin Heidelberg 2004,

Smith, M. (1999). Invisible crowds in cyberspace: Mapping the social structure of the Usenet. University of California, Los Angeles, London, Routledge Press, http://research.microsoft.com/pubs/view.aspx?pubid=798

Summers-Horton, K., and Horton, M. (1985). Status of the USENIX UUCP project. USENIX Association, Dallas, Texas, USENIX Association, El Cerrito. Calif.

Top1000.org (2006), Top 1000, accessed: March 28 2006, http://www.top1000.org/.

Turner, T. C., M. A. Smith, et al. (2005). "Picturing Usenet: Mapping Computer-Mediated Collective Action." Journal of Computer-Mediated Communication **10**(4).

Zenger, R. (2002), My spam trap setup, accessed: 12 sept 2005, http://rejo.zenger.nl/abuse/spamtraps.php.

Zocholl, R. (1996), Get that Spammer! , accessed: 10 April 2006, http://www.toppoint.de/~zoc/gspam.html.