Crossroads
The ACM Student Magazine

Peer-to-Peer Collaborative Spam Detection

by ***Nathan Dimmock*** and *Ian Maddison*

Introduction

Although there are many potential solutions to spam, it oftenappears that filtering and blocking are the best. Unfortunately these techniques are inadequate, as evidencedby the continuing proliferation of spam. Here we describe a student project thatevolved collaborative filtering, previouslyimplemented using a centralized repository of spam information, intoa distributed, collaborative, peer-to-peer-based spam detection system.

Survey of Existing Filtering Techniques

Effective filtering solutions must identify spam by distinguishing it from legitimate e-mail.

Rule-based systems filter spam based on patterns of keywords within amessage's text. Unfortunately, spammers often obscure these patterns byencoding letters as punctuation or numbers that are visuallysimilar to human readers, but defeat filters. Moreseriously, simplistic rules can also be triggered by legitimate e-mailsleading to a high rate of false-positive classifications.The very latest rule-based systems use Bayesian inference todynamically "learn" new rules [**4**], butjust as thicker armor begat bigger and better guns, spammers have responded withnew obfuscation techniques that target these inference engines.

Identity-based solutions such as blacklists, whitelists, andchallenge-response systems have also met unsolicited e-mail with limited success. Forgingan e-mail identity is simply too trivial to make blacklists effectiveand challenge-response systems make such a drastic change to the wayin which e-mail works that many of the features that make itso attractive are lost.

The most accurate spam classifier is still a human. To take advantage of this, Vipul's Razor [**12**] keeps acentralized database of spam messages against which incoming e-mail can thenbe queried. Messages are added to the database by humanssubmitting a hash of a message they believe to be spam and areputation system is used to assess the quality of the submissions andprevent attackers, adding erroneous entries to the system.

However, while the Razor database is replicated, it is a proprietary systemwith a limited number of servers, which makes it potentially vulnerableto denial of service attacks, such as those that forced the firmOsirusoft to close down its blacklist of spam-sending servers in August 2003 [**7**]. To counter this, a project was undertaken to design and implement a peer-to-peer-basedcollaborative spam detection system, similar in principle to Vipul'sRazor, but distributed.

Objectives

At the outset of the project the following objectivesfor the system were decided upon.

Privacy and obfuscation:
Collaborating with others requires the sharing ofinformation, but the contents of a user's e-mail must not berevealed. In addition, the system must overcome attempts by spammers to obfuscate the content of their message.

Efficient network usage:
In order for people to contribute to the network it must generate a minimal amount of traffic and avoid consuming a large proportion of the user's available bandwidth.

Trust:
The Internet is an un-trusted environment, but informationprovided by others is the primary tool a collaborative system uses tofilter messages. The reliability of this information must beconsidered.

Security:
Spammers must not be able to subvert the P2P network as that would permit them to render the whole system ineffective.

Interface:
The system must work with as many existing e-mail clients as possible; a system requiring a custom mail client is not acceptable as that would represent a significant barrier to its adoption by the Internet community.

System Architecture

On a peer-to-peer network all nodes are equal and must act as bothclient and server. In this system, the client:

1. collects and parses e-mail messages from the mail server
2. searches the network, querying nodes for their conclusions on similar messages
3. receives responses from these nodes and collates them
4. identifies principals that have responded and combines responses to reach a conclusion
5. redelivers analyzed e-mail messages to the user's normal e-mail client

The server listens for queries from other nodes and answers them withconclusions it has reached on similar messages.When analysis is complete and a conclusion is reached, it must bepublished so the server can answer queries. If a user disagrees withthis conclusion the published information will then be updated.

Both client and server were implemented in Java, using the Javamailand Apache XML-RPC libraries. An overview of the system architectureis given in **Figure 1**.

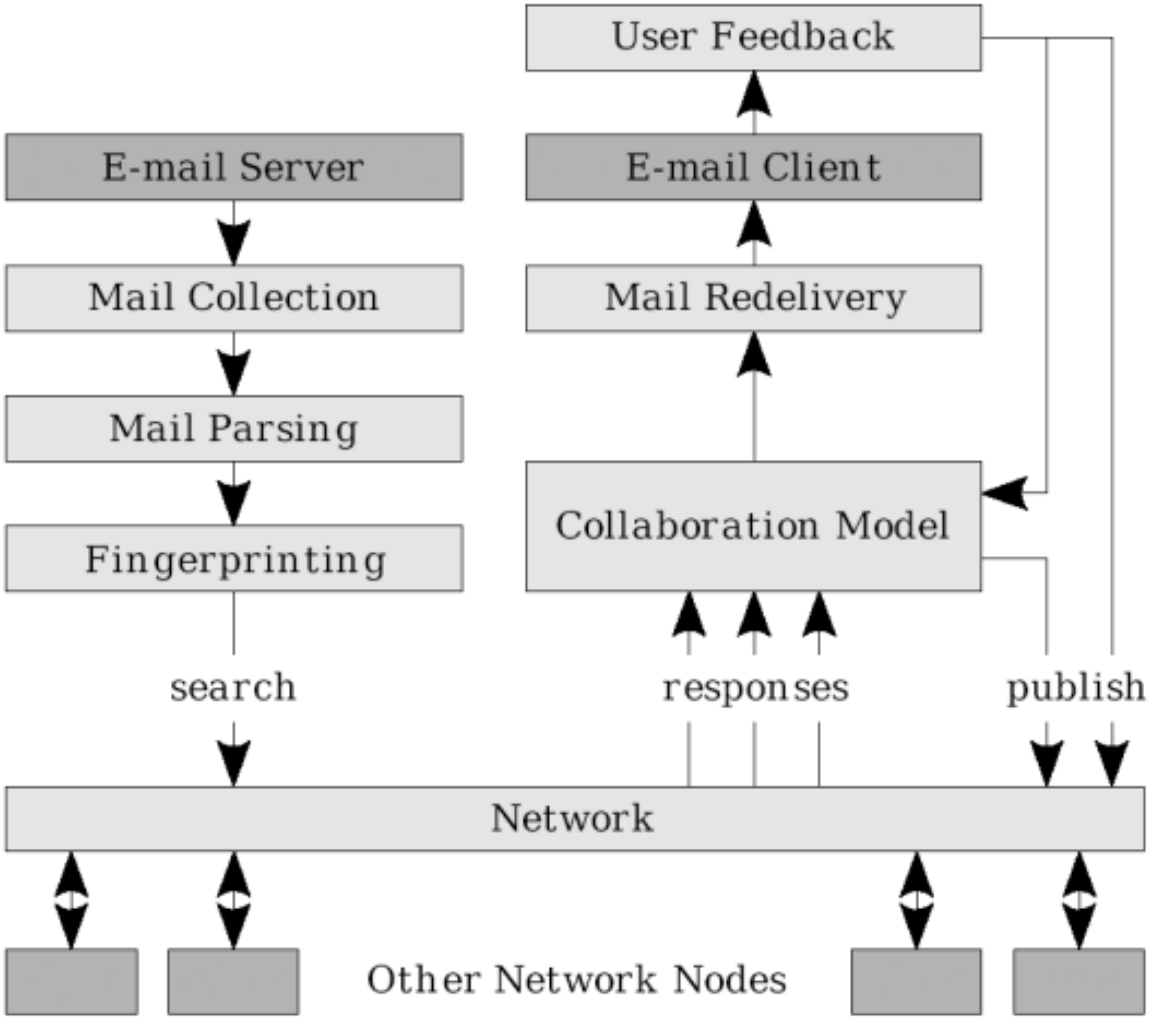

Figure 1: The System Architecture.

De-obfuscating Messages

Automated comparison of spam messages is made difficult due to theobfuscation techniques spammers use to evade existing filters. Adetailed examination of spam messages received over a one week periodshowed the following common obfuscation techniques:

- appending a collection of nonsensical, but actual English words at the end of messages (e.g.,"coco divisive stutter epilogue cloister dividend knapsack elfin ambidextrous lombard renegotiable")
- inserting random spaces, characters, punctuation, and new lines (e.g., Mor tg age, via-gra)
- substituting alphabet characters for similar visual representations such as accented characters, numbers or HTML entities (e.g., pi11)
- inserting invalid HTML tags into words that will not be displayed by themajority of e-mail clients
- embedding an image which includes the actual content of the message

Therefore, to find whether the same spam message has been received byother users, the system must search for messages that are the same buthave been obfuscated differently. The most straightforward method forachieving this is to search for similar (not just identical) messages,thus making theproblem one of approximate matching rather than de-obfuscation.

Approximate Matching

Many algorithms exist that can compare two strings and produce asimilarity measure, but these are not appropriate since comparing twousers' messages would require one to disclose their content to theother. Strings can be hashed such that the original text is guaranteedto be unrecoverable, but by design,similar strings produce pseudo-randomly different output hashes.

Damashek proposes using n-grams for approximate matching [3].This algorithm takes a string such as, "here is an example string" andbreaks it down into groups of consecutive words (entities separated bywhitespace). If N=2, the n-gram process produces: "here is", "is an", "an example", and "example string".

A similar string, such as "here is another example string", produces asimilar set of n-grams. Several similarity measures are proposed, thesimplest being Tversky's Ratio Model [1]:

$$s = \frac{a}{a + b + c}$$

where a is the count of common n-grams in the two sets and$b$ and $c$are the counts of distinctive n-grams that one document has but theother does not. In this technique, comparison survives hashing, matching n-grams produce the same hash and so may still be compared after hashing, butrecovery of the original text is impossible. Consequently, e-mail messages maybe compared without sacrificing privacy. Implementation is also straightforward and language independent.

For speed, substrings of a fixed length were chosen as digest valuesinstead of groups of consecutive words. To demonstrate how this works, using the string "THIS IS ATEST" as an example, and choosing a substring length of four, thefollowing substrings are possible:

```
THIS IS A TEST = "THIS" = 1857C0D491FA3B42C0088F97565597A5THIS IS A TEST = "HIS " =
47EDB96AECA51F93710F7F8F1ACCFE68THIS IS A TEST = "IS I" =
E91C43ED2F6A64A60BF1C51914D51B54THIS IS A TEST = "S IS" =
968665304CD9B855F3B352BC74FA4429THIS IS A TEST = " IS " =
4EBF2020E65B3D8FE7F6B4F2940E2B2DTHIS IS A TEST = "IS A" =
A2AD21457A9B25BA4698A388CD82B574THIS IS A TEST = "S A " =
53E5859B25DD37F82463A0279FD7511DTHIS IS A TEST = " A T" =
A71D3C26FCDEB0D078FC83A9B1020FE1THIS IS A TEST = "A TE" =
8D7626CCFA6E72232CC93DF17938DAA8THIS IS A TEST = " TES" =
F7CA2594F3A82D5F4ED520728638086CTHIS IS A TEST = "TEST" = 3D8BAF1F599C23A12028FD8A56FDAB65
```

If the same fingerprinting technique is performed on the slightlyaltered string, "THIS ISN'T A TEST", 14 digest values areproduced, of which eight are also found in the set of digestsgenerated by the first string.

Networking

The bandwidth-usage of the system will primarily be determined by theimplementation of the peer-to-peer network.The network must also be robustto ensure that a node unexpectedly leaving the network does not produceerrors or lead to network fragmentation. There must also be no singlepoint of failure that could be attacked by the spammers. Threepossible network topologies were considered:

- **Flat -** The simplest form of topology is an unstructured network, such as that used by the Gnutella

[9] protocol. All nodes are considered equal so organization is easy, but load-balancing and tolerance for nodes leaving the network is harder to achieve.

- **Hierarchical -** A more complex topology involves having a number of levels, for example the two-tier system used by the FastTrack [11] file-sharing system. Organizing the structure is more difficult (e. g., who decides which nodes go in which tier), although load-balancing and fault-tolerance are easier to manage.
- **Distributed Hash Tables (DHTs) -** Examples of popular DHTs are Chord [8] and Pastry [10] which allow the creation of scalable, self-organizing, load-balanced, and fault-tolerant structured overlay networks. DHTs are discussed in more detail below, but in summary they are a distributed form of the hash table data structure, where data can be looked-up directly by computing a *key* value from the data which is then used to store and retrieve the data directly.

For the initial prototype implementation a simple, flat networktopology was chosen using flood routing for search requests. Nodesconnect to each other in an ad-hoc fashion, each node supporting up tofive *neighbor* nodes. When it wishes to search the network fora matching e-mail fingerprint, it simply queries each of itsneighbors, which in turn, query each of their neighbors (exceptfor the source of the search request), and so on as can be seen in **Figure2**. As can also be seen in the same figure,looping is common; to prevent looping each node stores a cacheof recently seen search requests and if any incoming search requestmatches a member of this cache then it is not forwarded.

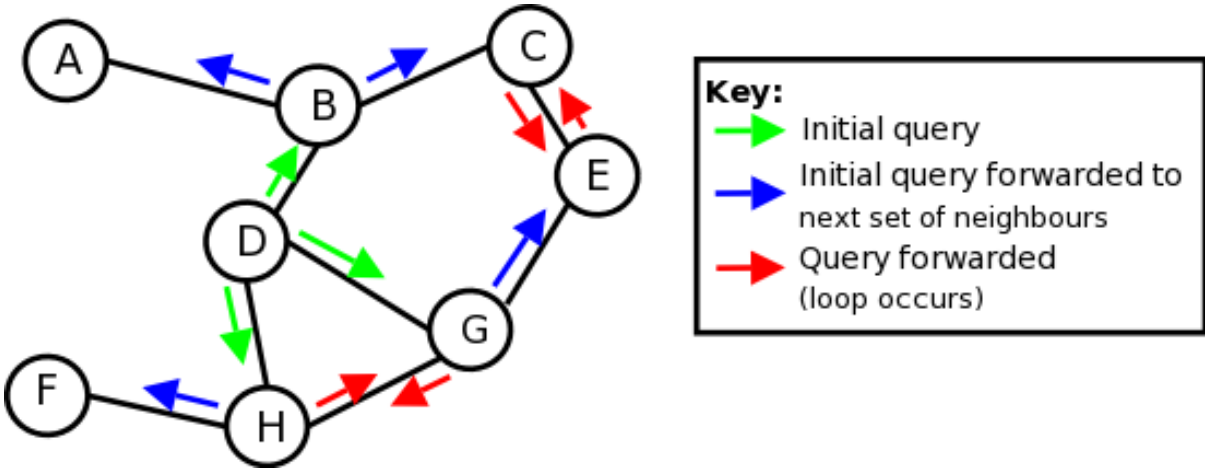

Figure 2: Searching in the simple network topology. Node D initiates a search by querying its neighbors which in turn forward the request to their neighbors.

Searching the Simple Network Topology

When a node receives an e-mail it wishes to search the P2P network fore-mails with similar fingerprints to determine whether this is likelyto be a spam e-mail or not. For example, if the incoming e-mail has afingerprint consisting of the digests {1,2,3} then all fingerprintscontaining any subset ({1},{2},{3},{1,2},{1,3},{2,3},{1,2,3}) of thesedigests are a possible match. The number of digests two fingerprintsshould have in common to be considered a match is discussed in theevaluation below. Since the number of digests produced by thefingerprinting algorithm is proportional to the lengthof the e-mail, it is impractical to broadcast all of these digests tothe entire network. As a compromise, a random sample of digests aresearched for, which still allows approximate matching to take place,although obviously with a potentially higher rate of failure.

A More Scalable Network Topology

Although a simple network topology works well for small networks, itdoes not scale well to the sort of size reached by popular P2P networks. Distributed hash tables (DHTs) have recently been proposed as an effective andefficient paradigm for building scalable P2P networks. A variety of different DHT algorithms have been published, each withtheir own strengths and weaknesses, but given the time constraints of this project it was decidedto use the Chord [8] algorithm as it is also one of the easiest to understand and deploy.

In Chord, nodes connect together as a structured ring as shown in Figure 3 . Each node has a key (a largeinteger) which serves to order it with respect to other nodes in thering. In **Figure 3**, the key of node B isgreater than the key of node A, but less than the key of node C. Tocomplete the ring, the highest and lowest nodes connect together. Asnodes join and leave the system Chord efficiently updates its routingtables accordingly, creating a self-organizing P2P network thatcontinues to work even when membership is volatile. In a network of$n$ nodes, a node with a particular key can be located in O($\log n$) time by exploiting thestructure of the network, making it much more efficient than the floodsearching used in the simple network topology.

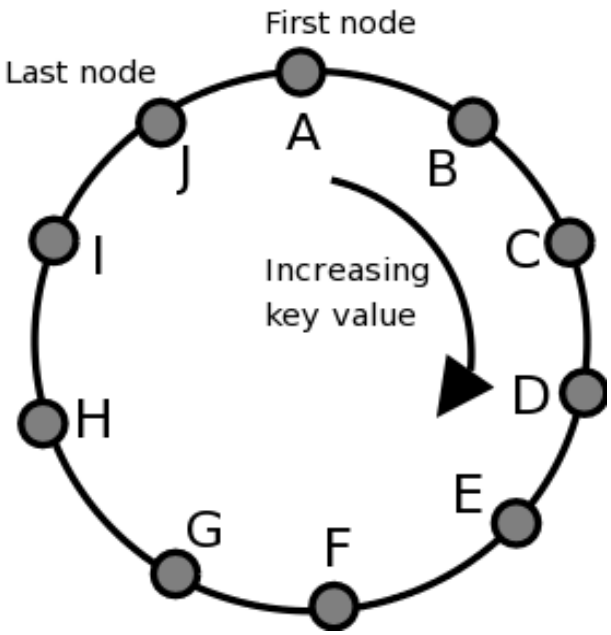

Figure 3: The Chord Network Topology.

Implementation

The first problem encountered was that Chord keys mustbe totally ordered, which e-mailfingerprints, being a set of digests, are not. Methodsneeded to be developed to findsubsets of fingerprints without performinglarge numbers of searches.

The solution was to publish each individual digest within a fingerprint asseparate logical data items in the network. Digests (as large integers) canbe ordered, however searching for a fingerprint now becomes multipleseparate digest searches. As before, the client searches only for a sample ofdigests to minimize bandwidth usage.To address the problems of maliciousprincipals the following structure for the node key is used. The top(most significant) 16 bytes of the key are the hashed digest value ofthe piece of information being stored at this node. The next fourbytes are the IP address and the final (least significant) two bytesare the port number of the node. The advantage of this key structureis that identical digests published

by several nodes will be adjacentin the ring.

Trust and Reputation Model

The most straightforward way of ensuring reliability of informationthat others provide is to build a trust relationship, a list ofprevious interactions (both good and bad), which is used to placegreater weight on information provided by those who are usuallycorrect and vice-versa. Many different trust models have beenpublished, as surveyed by Grandison and Sloman [**5**], but given the authors' existing knowledge and experience working on theSECURE project [**2**], it was decided to use acut-down version of that model for the trust system.

Entity Recognition

Entity recognition forms the foundation of any reputation system, asany trust you assign to a user is conditional on that user being theone to whom the reputation information refers. Identity is a trickyissue on the Internet as many identifiers such as IP addresses aretransitive and unauthenticated, making them easy to spoof, and so itwas decided to use a public-key system. The first time a node joins thenetwork they generate a cryptographic key with two parts, one of whichis kept secret (the "private" key) and the other part is made widelyknown to other users (the "public" key). The private key is then usedto cryptographically sign published e-mail fingerprints; receivingnodes may then use the public key to verify the signature and confirmthe authenticity of the publisher.In order to avoid using potentially confusing and overloaded terminology,from here on members of the P2P network will be referred to as**principals** where a principal is defined as the holder of aparticular private key.

Trust Metric

A trust metric is a measure of how trustworthy one principal considersanother principal. For the initial prototype it was decided toexperiment using the mean-average of the user's previous interactionswith a principal, where each interaction is rated using a float in theinterval [0,1]. This has the advantage of being simple to calculate,store, and use in the decision making process but the disadvantage isthat it is difficult to bootstrap the system. Since any node that gainsa bad reputation may simply generate a new key-pair (and equivalently anew identity), the initial level of trust associated with previouslyunknown principals must be very close to zero, thereby making itdifficult to distinguish between unknown and untrustworthy principals.

Collaboration Model

Upon receipt of a number of replies from the network claiming thatthis fingerprint represents a message which is spam, the node mustdecide whether or not to believe them. The SECURE model uses anoutcome-based risk-analysis for decision making. In the case of ane-mail being spam or not, if the message is marked as spam, the twopossible outcomes are that (1) the e-mail really is spam and (2) thatthe e-mail is actually legitimate.

The potential costs of each outcome, relative to whether it isdecided to *mark* a message as spam or allow it to *pass* intothe user's inbox must now be considered. Costs are expressed relative towhat would be incurred without the spam filter in place (this helps toavoid getting bogged down in questions of exactly how much an e-mail is"worth" to the user). **Table 1** isthe cost matrix for the two outcomes respective to each of the twomembers of the decision set.

	Pass	Mark
Spam Message	0	-1
Real Message	0	E

Table 1: Outcome cost matrix for the spam filtering application.

Note that passing a message always costs zero, since that is whatwould happen without the filter in place. Marking a message provides abenefit (cost of -1) if it is spam, equivalent to the time saved andthe value of not interrupting the user. This is arbitrarily set to bethe unit of cost in this application. Marking a real message has apositive cost of E (the false-positive error cost). E is likely to beconsiderably more than 1, and is configured by the user based on theaverage severity of the consequences of missing a valid e-mailrelative to the cost of their time.

For each principal that responds to a request for information aboutan e-mail fingerprint, a lookup of the user's opinion of the probability thattheir judgment be correct, $\bar{x}$ must be performed. Some principals may also haveindicated that this is not spam, and so the probability, p, thate-mail is real mail is derived using the formula:

$$P(\text{real mail}) = \begin{cases} \bar{x} & \text{principal claimed it was real} \\ 1 - \bar{x} & \text{principal claimed it was spam} \end{cases}$$

The expected cost of marking a message as spam is then computed usingthe standard formula for mathematical expectation from probability theory:

$$p \times E + (-1)(1-p) = p \times (E+1) - 1$$

(with *p* defined as above).

Clearly this principal's claim that the message is spam should only beaccepted if it is in the user's interest to do so, that, is the expectedcost is negative. If the expected cost calculation for any principalsatisfies this condition then the message is marked as spam, furthercalculations on the set of expected costs and benefits would bemeaningless, as information from those principals that are not trustedis irrelevant.

Whether the ultimate decision is to pass or mark the message,principals whose information agreed with this analysis have theirnumber of positive interactions incremented by one, thereby increasingtheir $\bar{x}$, while principals who supplied false information havetheir number of negative interactions incremented, similarlycausing a decrease in their $\bar{x}$. Should the user, when viewingthe e-mail disagree with the decision made, they are given anopportunity to provide feedback whichreverses these updates so those who agreed with the user have theirtrust-rating increased and those who disagreed decreased.

Initial Trust Value

As noted earlier, the trust metric usedin this system makes it difficult to distinguish between principalsthe

user has determined to be very untrustworthy and those which theuser simply has not encountered before. To counter this the systemstores all the judgments it receives from other principals, but onlyacts upon the advice of trusted ones. Once the user feedback isobtained and the "correct" decision known, the trust value of all theprincipals who gave information is updated. This allows trust innewcomers to be slowly built-up over time while ignoring theiropinions until they have gained a sufficiently high level of trust.

User Interface

For the most part the P2P client program runs unobtrusively in thebackground, acting as a proxy between the user's mail client and theirmail server, collecting the user's e-mail and delivering it to theirinbox or spam folder as appropriate. However, if the system makes amistake then the user requires a mechanism to provide feedback; a footeris appended to each e-mail passing through the system. As can be seen in**Figure 4** , this footer contains a URLwhich, if clicked by the user, will in most e-mail clients, cause thepage to be loaded in the user's web browser. The P2P client thereforealso acts as a small web server running on the user's machine andlistens for such page requests - requesting a page associated with arecently processed e-mail is taken to mean that the user disagreed withthe decision that was made and the system is updated as described in theprevious section.

As the software runs as a proxy that may be used in conjunctionwith any mail client it thus meets the *interface objective* set out atthe start of the project.

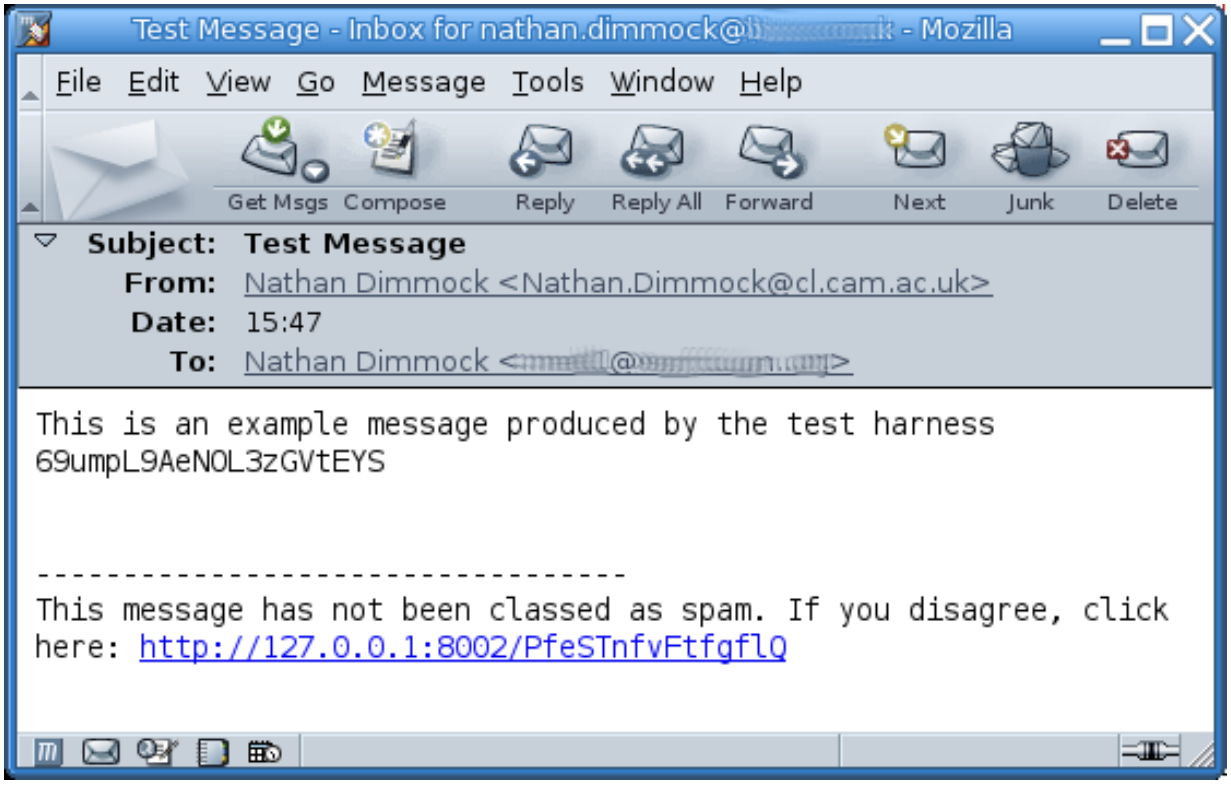

Figure 4: Example e-mail which has not been marked as spam.

Evaluation

Fingerprinting and ApproximateMatching

The approximate matching algorithm has two variable parameters, the sizeof the sample of digests sent out by the querying node and the length ofthe substring used to create the digests. Setting these too

highincreases the network traffic; setting them too low decreases theeffectiveness of the filter. Consequently, optimize their values.

Using a corpus of 50 spam messages, for each test message thecharacters and words appearing to be part of obfuscations weremanually replaced with the Unicode character `0xFFFF` (acharacter not used in any of the sample messages). These markedcharacters were then substituted for other random characters allowedthe creation of "editions" of the e-mail that represent possiblealternative obfuscations sent to other e-mail addresses. Thefollowing experiment was then carried out:

1. Ten pairs of two editions of each message were created.
2. For each pair, a random sample of digests of the first fingerprint of the first member was taken.
3. The sample was tested against the full fingerprint of the second of the pair and the percentage similarity (match fraction) calculated.

The test was repeated varying the size of the digest sample from fiveto twenty and the length of the substring from five to thirty-fivecharacters. The results, shown in **Figure 5** ,confirmed our expectations that a smaller substring length produced ahigher likelihood of achieving a match. **Figure 6**shows in more detail the lines for sample sizes of 5, 11, and 17, whichconfirms that even at smaller sample sizes, the rate of change of thematch fraction is small enough that it was concluded that the optimumsubstring is the shortest possible given the bandwidth available tothe system.

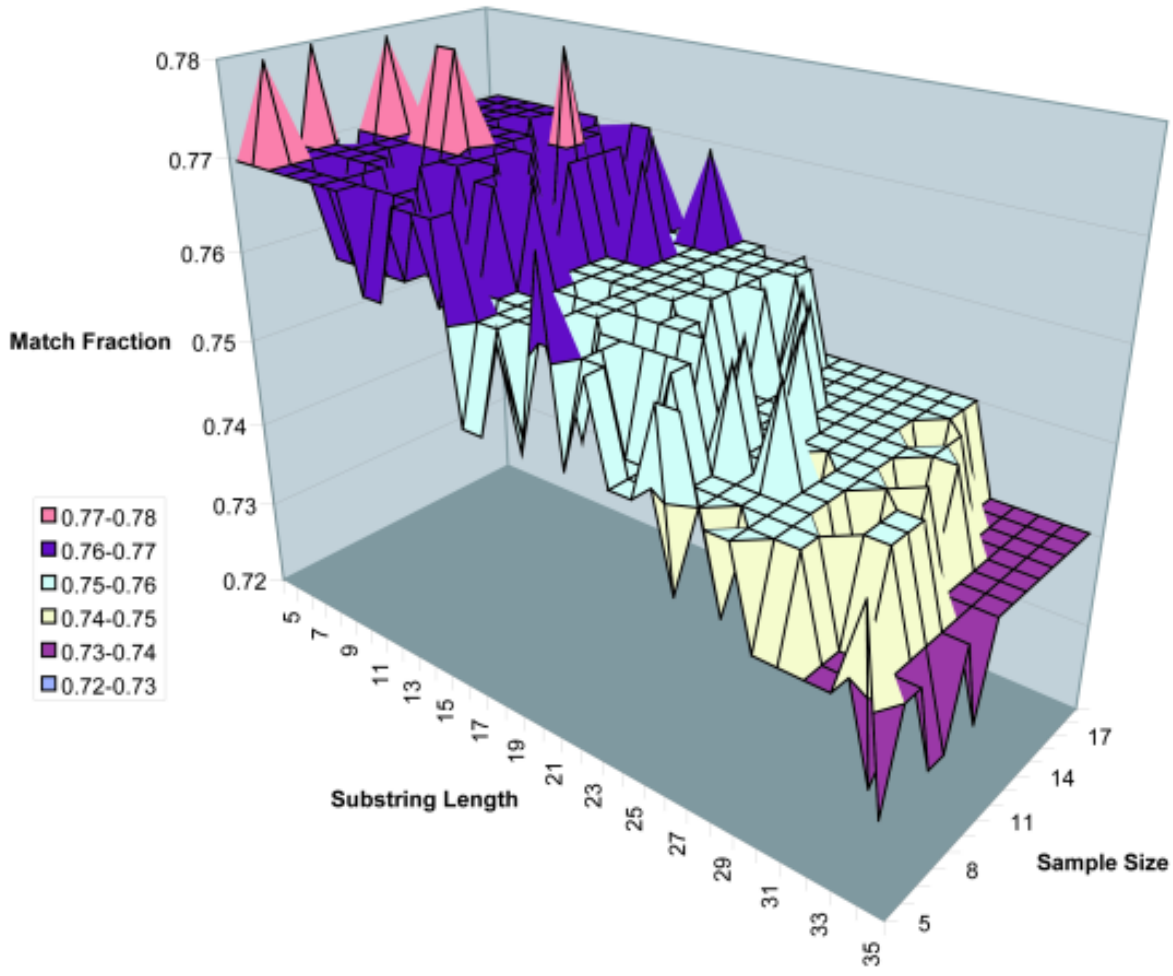

Figure 5: Graph of Approximate Matching Experiments.

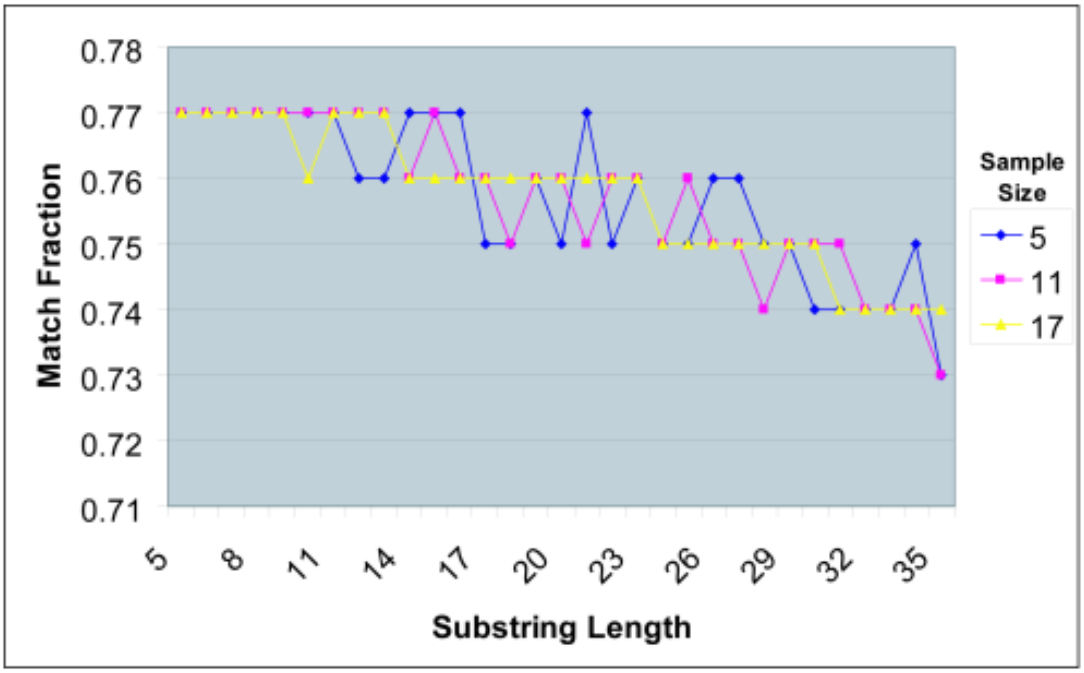

Figure 6: Graph showing how match fraction varies with substring length of three different sample-sizes.

The effect of sample size on the number of messages matched is more difficultto determine. **Figure 5** shows that its effect on thematch fraction in this experiment was minimal but intuitively thesmaller the sample size, the more likely false matches are tooccur. The conclusion was therefore that the choice of value for thisparameter should also be determined by the bandwidth available to thesystem.

The final parameter to be considered in this module is the proportion of common digests two fingerprints should have in order to be considered a match. This clearly plays an important role in the effectiveness of the spam detector since setting it too high will result in spam messages being missed, while setting it too low will result in an increased number of false-positive matches. **Figure 5** shows that the choice of this parameter should be dependent upon the chosen substring length and sample size, but it should be in the region of 70-80%.

Collaboration Model

The aim of the collaboration model is to distinguish between trustworthy users who provide accurate information and those less trustworthy users who, for whatever reason, occasionally provide incorrect information. For example, theoretically the network should partition into two types of users, spammers and non-spammers, but a spammer may try to escape detection by providing accurate information some of the time — for instance they may provide accurate information about their competitors spam, and only false information about their own. Since recent figures [**6**] have suggested that 50% of all e-mail received is now spam, even a node that rated every e-mail as spam (for malicious reasons or simply through mis-configuration) would be correct 50% of the time.

To evaluate the effectiveness of the collaboration model inidentifying good and bad principals, the behaviorof six principals was simulated: one whose information is 100% accurate,one whose information is accurate 90% of the time, and so on. Foursimulations were performed, each with a different initial value oftrust for previously unknown principals. As can be seen from thegraphs in **Figure 7** and**Figure 8** themetric converges to the appropriate value very quickly, regardless ofthe initial trust value used.

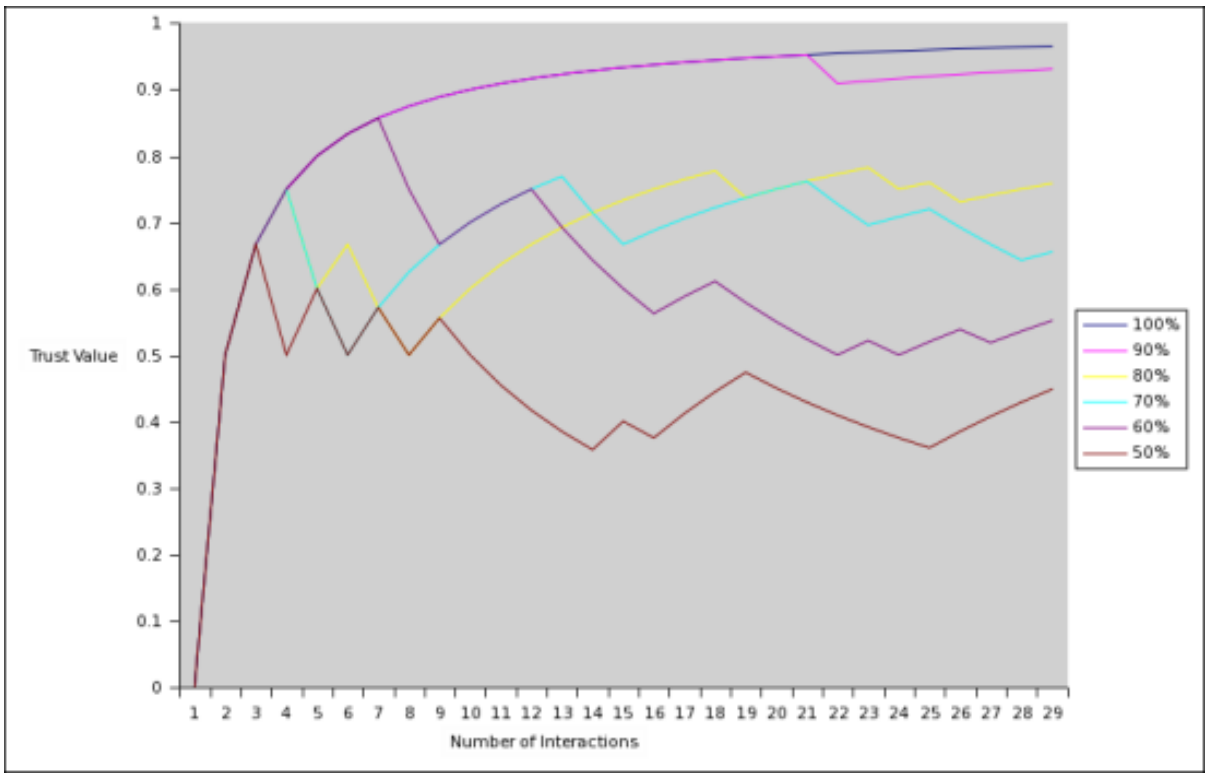

Figure 7: Simulation of six users who publishinformation of varying degrees of accuracy. Initial trust value of 0.

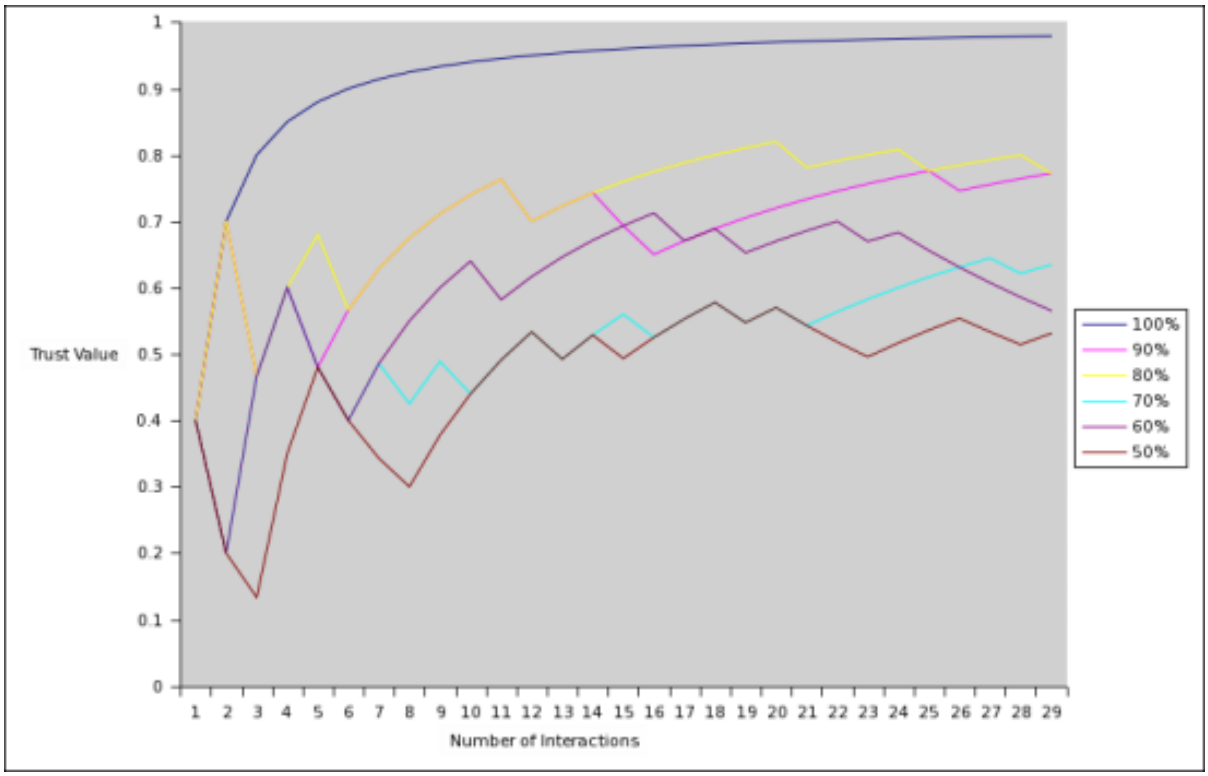

Figure 8: Simulation of six users who publishinformation of varying degrees of accuracy. Initial trust value of 0.4.

Another possible attack on the collaboration model is the so-called **Byzantine behavior**. In this scenario, illustrated in **Figure 9**, the attacker behaves well until their trust value reaches a certain threshold (e.g., 0.9), then utilizes their trusted position to misbehave. When their trust value drops below a lower threshold

they begin to behave well again, and repeat the cycle.

As **Figure 9** shows, unfortunately the trust metric used in this project was too simple to defend against this attack. The problem is that the trust metric initially reaches a high level of trust very quickly and while over subsequent cycles change is many times slower, since there are no controls on identity creation, a spammer may simply generate a new identity - even returning to the default trust level would allow them to gain trust level 0.9 within another seven positive interactions compared to 60 if they continued using the old identity. To defend against this attack, a more suitable trust metric would have the property of increasing very slowly with positive interactions but decreasing very quickly when there is bad behavior, much like the "additive increase, multiplicative decrease" algorithm used in TCP congestion control. Unfortunately there was insufficient time in this project to implement and evaluate such an algorithm.

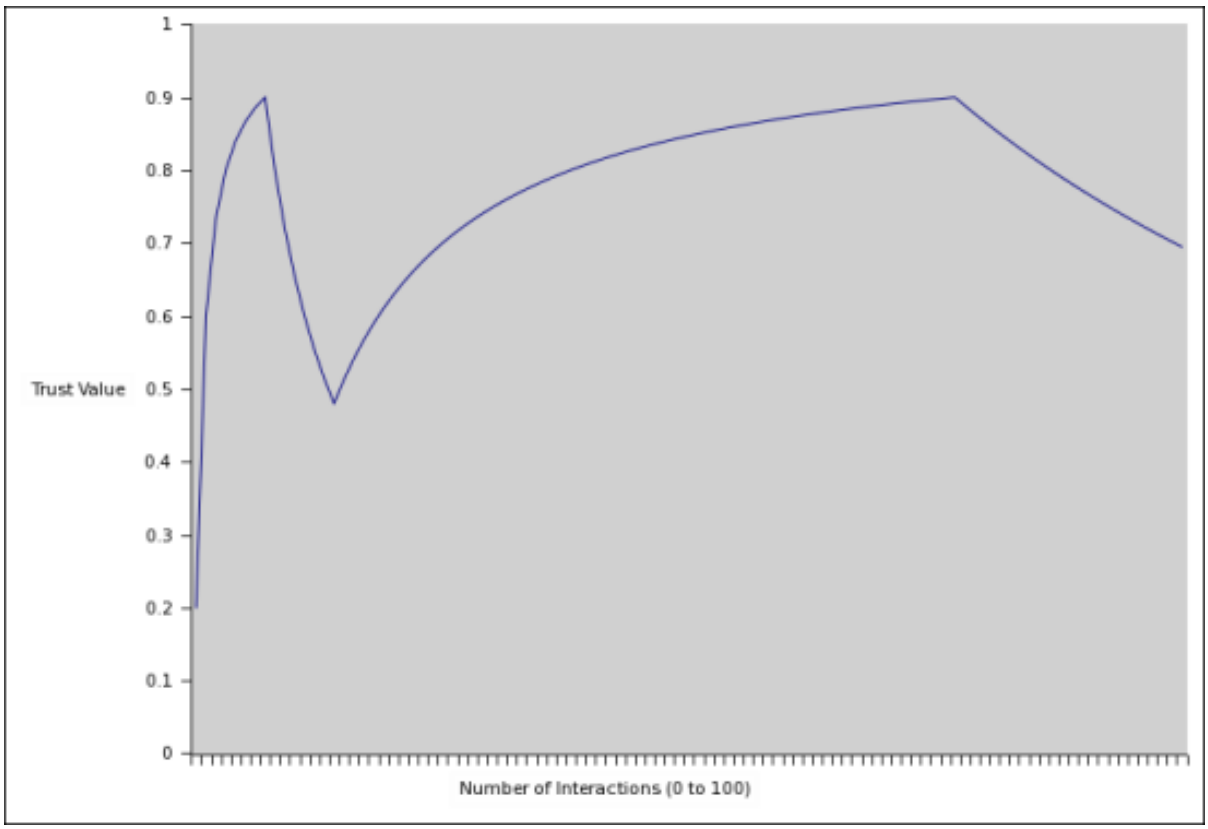

Figure 9: Simulation of a principal who displays Byzantine behavior. Initial trust value=0.2.

Security Threat Analysis

Security is of particular importance. If individual nodes suffer frompoor security properties, an attacker could potentially disruptsufficient numbers and the operation of the entire network isjeopardized and/or the effectiveness of filtering is reduced.

Analyzing the security of a system involves detailing the potentialattacks against it by developing a threat model. Potential threats tothis system include:

- impersonation of other principals
- deliberate return of invalid or incorrect data as a response to a query
- attempts to disrupt the network
- denial of service attacks

As described earlier, impersonationattacks were defended against using public/private keys and digitalsignatures. Likewise, deliberate propagation of misinformation isdefeated through the use of a trust-based collaboration model.

By distributing the database of spam information using a peer-to-peermodel such as Chord, denial of serviceattacks against the network as a whole are much more difficult thanwhen the information is concentrated in a small number of well-knownservers. However, denial of service attacks involving packet flooding directed against aparticular node cannot be prevented at the application level as thisis a security flaw to which all Internet applications are susceptible.

Conclusion

In this article a recent project to design andimplement a peer-to-peer collaborative spam detectionnetwork was presented. It is believed that each of the key objectives has been addressed,namely dealing with the problem of obfuscated spam message designed tofool automated filters, trust in the information supplied by otherusers, security, consumption of network bandwidth, and user interface.

As a distributed system, a full evaluation of the effectiveness of thespam filtering is difficult and time-consuming to perform and thereforebeyond the resources available for this project. However, the testingperformed on the key components gives a good indication of how thesystem would perform in the field.

Future work includes an improved trust metric and collaborationmodel. The current trust metric has some weaknesses, and the collaborationmodel could also be extended to include second-order information aboutprincipals in the form of recommendations. This would help mitigatethe problem of slow converging trust metrics, namely that it takes toolong for a user to build sufficient trusting relationships within thenetwork, as information about other user's experiences with aprincipal could then be imported. Further experiments could also becarried out using alternative distributed hash table algorithms, suchas Pastry or Tapestry to see how they effect the amount of bandwidthconsumed by the system.

Acknowledgements

This work was inspired and supported by the EU Future and EmergingTechnologies project, **SECURE** (SecureEnvironments for Collaboration among Ubiquitous RoamingEntities). Theauthors would like to thank Dr. David Ingram for his insightfulcomments and assistance inanalyzing the risk of spam filtering.

References

1

Bradshaw, J. Introduction to Tversky Similarity Measure. February 1997. <**http://www.daylight.com/meetings/mug97/Bradshaw/MUG97/tv_tversky.html**>.

2

Cahill, V., Gray, E., Seigneur, J.-M., Jensen, C. D., Chen, Y., Shand, B., Dimmock, N., Twigg, A., Bacon, J., English, C., Wagealla, W., Terzis, S., Nixon, P., Serugendo, G., Bryce, C., Carbone, M., Krukow, K., and Nielsen, M.Using Trust for Secure Collaboration in Uncertain Environments.*IEEE Pervasive Computing* 2(3), August 2003, pp. 52-61.

3

Damashek, M.Gauging Similarity with N-grams: Language-independent Categorization of Text.

Science 267, February 1995, pp. 843-848.

4

Graham, P.*A Plan for Spam.* August 2002.<**http://www.paulgraham.com/spam.html**>.

5

Grandison, T., and Sloman, M.A Survey of Trust in Internet Applications.*IEEE Communications Society, Surveys and Tutorials 3*(4), 2000.

6

Gray, P. Spam Now More Than Half of All Received Email.Silicon.com, June 2003. <**http://networks.silicon.com/broadband/0,39024661,10004430,00.htm>**.

7

Gray, P.Osirusoft 'Closes Doors' After Crippling DDoS Attacks. ZDNet Australia, August 2003. <**http://www.zdnet.com.au/news/communications/0,2000061791,20277794,00.htm**>.

8

Kaashoek, F., Morris, R., et. al. The Chord Project.<**http://www.pdos.lcs.mit.edu/chord/**>.

9

Ripoeanu, M.Peer-to-peer Architecture Case Study: Gnutella Network.In *Proceedings of the First International Conference on Peer-to-Peer Computing*, IEEE, 2001, pp. 99-100.

10

Rowstron, A. Pastry: A Substrate for Peer-to-peer Applications. <**http://research.microsoft.com/~antr/Pastry/**>.

11

Sharma, A.Peer to Peer: The Fasttrack Network. September 2002.<**http://www.pcquest.com/content/P2P/102091205.asp**>.

12

Vipul's Razor. <**http://razor.sourceforge.net**>.

Biographies

Nathan Dimmock (**nathan.dimmock@cl.cam.ac.uk**) is PhD student at the University of Cambridge Computer Laboratory. He holds a BA(Hons) from the University of Cambridge and is currently undertaking a PhD, doing research into trust-based systems, with particular interest in the field of ubiquitous computing.

Ian Maddison (**ijm@cantab.net**) recently graduated from the University of Cambridge with a BA(Hons) in Computer Science. A large proportion of the work described in this article was done as part of his final year project.