Improving the Efficiency of Spam Filtering through Cache Architectures

Ashok Khanal, Bahman Motlagh
EECS/IST/CECS
University of Central Florida
Orlando, FL- 32816
bmotlagh@mail.ucf.edu

Taskin Kocak
Dept. of Electrical & Electronic Engineering
University of Bristol
Bristol BS8 1UB, UK
T.Kocak@bristol.ac.uk

Abstract— **Blacklists (BLs), also called Domain Name System-based Blackhole List (DNSBLs) are the databases of known internet addresses used by the spammers to send out the spam mails. Mail servers use these lists to filter out the e-mails coming from different spam sources. In contrary, Whitelists (WLs) are the explicit list of senders from whom e-mail can be accepted or delivered. Mail Transport Agent (MTA) is usually configured to reject, challenge or flag the messages which have been sent from the sources listed on one or more DNSBLs and to allow the messages from the sources listed on the WLs. In this paper, we are demonstrating how the bandwidth (the overall requests and responses that need to go over the network) performance is improved by using local caches for BLs and WLs. The actual sender's IP addresses are extracted from the e-mail log. These are then compared with the list in the local caches to find out if they should be accepted or not, before they are checked against the global DNSBLs by running 'DNSBL queries' (if required). Around three quarters of the e-mail sources have been observed to be filtered locally through caches with this method. Provision of local control over the lists and lower search (filtering) time are the other related benefits.**

Keywords – DNSBL, Blacklist, Whitelist, Spam, Cache

I. INTRODUCTION

Just a few decades ago, e-mails were very much trusted. It was common for MTA's to relay e-mails. Slowly spammers began to take advantage and open relay became a nuisance. Today, spam along with the e-mail viruses has become a primary reason to blacklist someone.

Spam filters detect and prevent unwanted e-mails from entering our mail box. They examine various parts of e-mails to determine if they contain spam messages or are sent from the spammer machines. Broadly there are two categories of spam filtering [20]. *Origin or Address based* filtering uses network information (IP addresses and e-mail addresses); the examples being DNSBLs, challenge response etc. *Content based* filtering looks inside the e-mail and examines the message content; the examples being *Bayesian* and *rule based filters*. Content based spam filtering usually has a higher success rate (lower false positives) of finding spam and less potential for deliberately blocking senders [21].

Filtering out large amount of spam e-mails, that too from an increasingly varying sources, is difficult and time consuming without origin-based spam filtering (DNSBLs). In this work, we focus on filtering e-mails using the local caches and the DNSBLs. First, actual senders (IP addresses) are extracted using a recent e-mail log. These senders are then searched in the local caches (list containing IP addresses that are whitelisted or blacklisted) to find out if they should be accepted by the mail servers or not. Depending on the *hits and misses* in the local caches, they are then sent for a query against the external DNSBL server. Senders listed in the DNSBL(s) will be added to the BL cache and those not listed will make up the WL cache. The caches (where frequently accessed data are stored for rapid access) occupy the hard drive memory.

Caching methodology has shown to help reduce the name resolution overhead tremendously [1]. Because a significant fraction of DNS lookups never receive answers, they can adversely affect performances related to latency and bandwidth in the e-mail traffic. A study of spam traffic and comparison of popular DNSBLs [2] has shown that a very high percentage of all DNS queries accounting for DNSBL queries. Main motivation for this work comes from the fact that we need to reduce these DNSBL queries as much as possible, so that the resources used for this purpose, can be used more beneficially elsewhere.

DNSBLs are managed by various groups; each with their own focus lists and different policies in regards to how an IP address gets on (and off) the list. It is necessary to converge these lists because they vary widely in terms of maintaining their domains and responding to the queries. More uniform database and standard protocols for catching spam sources is a necessity now. It is also necessary to make the DNSBLs more complete (that contains a reasonable fraction of all spamming IP addresses) and responsive (that yields a low time period between the start of spamming and time when the spam source becomes black listed), to make them more effective [8].

II. RELATED WORK

Research works pertaining to the spam control have been going on actively for more than a decade now; primarily in the pursuit of overpowering the spammers or at forcing them into a disadvantageous position. Study of spam traffic as a network level behavior (say by tracing spam sent to a single domain over a period of time) has been helping to keep track of the distribution of spam senders (location, IP range) [16].

Proceedings of the Fifteenth IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'07)
978-1-4244-1854-1/08/$25.00 ©2008 IEEE

One of many spam filtering approaches, *challenge response system* is a program that replies to an e-mail message from an unknown sender by subjecting the sender to a test designed to differentiate humans from automated senders. It ensures that the messages from people can get through and the automated mass mailings of spammers will be rejected. Once senders have passed the test, they are added to the recipient's WL of permitted senders and will not have to prove themselves next time they send a message. This method has proven to be highly effective but not that practicable because of the necessity of a direct human involvement [3]. Other approaches like combating the affect of the spam on end-user mailboxes [4], lead spammers experience a significant bandwidth limitation and additional delay which partly shifts the cost of sending the spam back to the spammer's end. Other sought after approaches include moving away from the reactive methods into the proactive/preventive measures, to prevent the spammers harvesting e-mail addresses.

Spam is being sent from an increasingly larger set of IP addresses [8]. Spammers also are smart enough to be very transient and increasingly agile while distributing the spam. For combating this, there have been works on the unified technology where different types of content based and source based filtering are used together [2, 15]. Additional layers of filtering are also implemented: 'adaptive filtering' being integrated with Spamihilator software [5] or filter testing with 'greylists' (temporarily rejected list) at the SMTP connection level [6].

Though extensive filtering helps combat the spam, in the downside, it also hampers a free and open flow of communication among the network users [7]. Unquestionably, there are also harsh consequences to face: blocking of the legitimate e-mails and blacklisting senders wrongfully accused of spamming.

III. PRELIMINARIES

A. Basic E-mail System

In its simplest form (Figure 1) e-mail works in the following manner: the originating sender creates an e-mail by using his Mail User Agent (MUA), an e-mail client. The sender's MUA then transfers the e-mail to a Mail Delivery Agent (MDA). MDA routes the e-mail into the local mailboxes or forwards it if it is not locally addressed. If the e-mail is forwarded, MTA, a mail server, uses the Simple Mail Transfer Protocol (SMTP) to send and receive messages between different systems. The e-mail also usually passes through different filters before entering the user's inbox.

Two different servers are usually running in a mail server machine: SMTP (for sending) & Post Office Protocol version 3 (POP3) and Internet Message Access Protocol (IMAP) (for receiving). All these servers listen to a unique port number for any incoming e-mail: SMTP server uses port number 25, POP3 uses port 110 and IMAP uses port 143.

Say, a person with e-mail address **one@mailserver1.com** is sending an e-mail. He uses the services of MUA and the e-mail client. If the recipient is another user at same mail server, **two@mailserver1.com,** SMTP (simple mail transfer protocol) will hand over the mail to POP3 (post office protocol) server using MDA. SMTP server breaks e-mail into two parts: recipient (*one*), domain name (*mailserver.com*) and puts it in the message queue. If the recipient is a user of different mail server, say, **three@mailserver2.com,** SMTP contacts DNS (Domain Name System) to get the IP address of the receiving mail

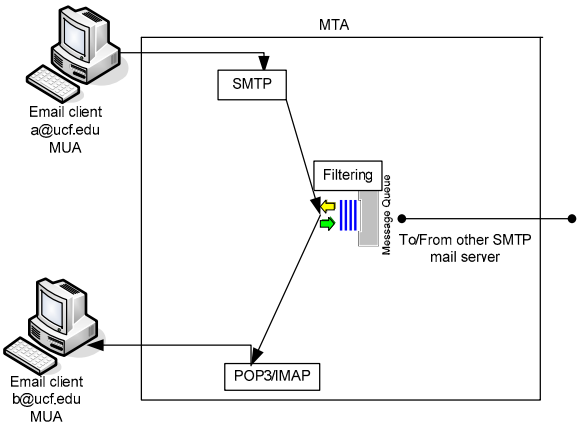

Fig. 1: Typical E-Mail System

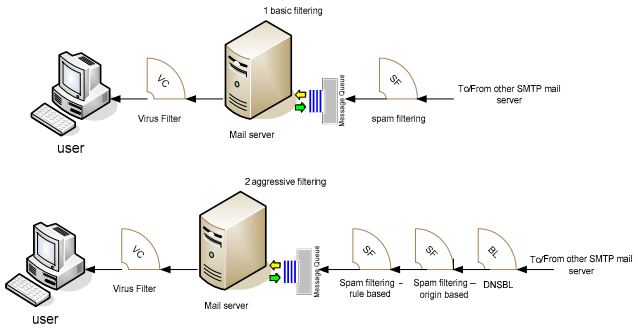

Fig.2: Varied Aggressiveness of E-Mail Filtering

server. MTA and SMTP then work together to handover the mail to a receiving server. In the receiving side POP3 and MDA work to put the message into the inbox. If the MTA finds out that an e-mail is blacklisted, it simply flags the suspected spam or directly deletes it.

Figure 2 shows varying aggressiveness of the E-mail filtering system. The filtering layers (external DNSBLs, local BL, local WL, various kinds of spam filters, virus filters etc) can vary in sequence from one mail server to another, with some mail servers including more filters than other. This will then vary the aggressiveness of the e-mail filtering process. The requirement of the number of filters depends on the volume of the e-mail traffic and the necessity or preference of certain filters more than the others. Most e-mails go through the external DNSBL queries and are filtered with the local database of BL and pass through the spam filters before or after being downloaded into the mail server. E-mail clients usually have their own BLs and WLs filter rules.

B. Common E-mail Systems and Headers

Almost every e-mail passes through at least four machines during its lifetime. This includes two mail servers and two client machines. Say, user1@mail.ucf.edu client wants to send an e-mail to user2@myisp.com. "user1" composes e-mail which is passed to its mail server (mail.ucf.edu). This mail server sees if it is destined for another "mail.ucf.edu" client or not; since it is for another mail server, it contacts "mailhost.myisp.com" and delivers the e-mail. The e-mail remains at this mail server until "user2" connects to the internet and opens his mail box. During these events mail headers are constantly being added to the user's message. The headers are added specifically, at the composition time (by e-mail program "user1" is using); the time when the e-mail program hands over the message to the mail server "mail.ucf.edu" and then at the transfer point from the "mail.ucf.edu' mail server to the "myisp.edu" mail server.

Some of the fields that are recorded during these transactions are: *Received, From, To, Date, Message-ID, X-mailer, Subject* etc. Inside each **Received** field, following pieces of information are embedded: ***from*** *hostname;* ***by*** *host name;* ***via*** *physical path;* ***with*** *protocol;* ***id*** *message-ID;* ***for*** *final e-mail destination.* Different E-mail systems have their own styles of received fields and additional information.

In the following example, *user1@mail.ucf.edu* (of mail.ucf.edu mail server) is sending a message to *user2@mailhost.myisp.com* (of mailhost.myisp.edu mail server). If an e-mail passes though an intermediate relay, there will be total of at least two received headers like this:

Received*: from intermediate.sender.com (some.firewall.com)(xx.xxx.xxx.xxx) by mailhost.myisp.edu with SMTP; id 12jh3hk21h3; 16 Nov 2003 19:50:37 -0000*

Received*: from mail.ucf.edu [xxx.xx.xx.xx] byintermediate.sender.com (server name 1.0) with SMTP id 34kjdfh3kjh3; Sun, 16 Nov 2006 13:38:22 -0600*

As the e-mail travels from one server to another, receiving mail server puts 'Received:' field on top of the previous 'Received:' field. We trace the path taken by the message by traveling up the header. The "by" (senders) should match with the "from" (receivers) in the preceding 'Received:' field in terms of IP addresses or domain names. For suspect (spam) e-mails, there usually would be non-matching/forged IP addresses or domain names.

C. E-mail Log

A major source of senders and relays is an e-mail log (e.g. a syslog file). Syslog is a standardized logging service that can be used on almost any computer platform [13]. It provides the breadcrumb trail we need while tracking down the computers that sent or relayed the e-mails. Syslog's standard log line is formatted as: [date] [time] [system] [tag]: [ID] [message]. The date and time sections of this format act as the log message's timestamp. The system section specifies which computer sent the syslog message. It may show an IP address or a hostname (domain name). The tag section tells which application or the process generated the message. The message section contains additional information of the process that generated the message and the text of the message. At times 'tag' and 'content' fields are considered to be the parts of the message field. A mail server usually processes multiple mail requests at a time. Multiple entries from the logs that pertain to the same e-mail are combined into a single entry.

As with the e-mail headers, different log files contain different logged information. With some log files, we obtain information on various filters and services each e-mail (identified by the message ID) has to pass through. Others provide information on the status of each e-mail (successfully sent, service currently deferred, service unavailable, from unknown sender etc). This helps us understand if the message has completed its journey from the originator to the recipient's mail box or if the system needs to complete the transfer again. The senders, hosts and relays can be identified with the IP addresses or the domain names appended to the specific strings (*from, host, relay*). Other log files have a tabularly formatted data containing different sections or headers like *Date, Time, Client-ip, Client-hostname, Server-hostname, Server-IP, Recipient-Address, Event-ID, MSGID, Priority, Recipient-Report-Status, Total size, Number of Recipients, Origination-Time, Encryption service, Version,* etc, which provide numerous information about the sending and relaying machines.

Spammers often forge the headers of the e-mails to avoid losing their accounts and to evade the e-mail filters. They insert forged contents in the e-mail headers that would point to somebody else as a real sender. As a result, e-mail logs are generated that contain false e-mail and IP addresses of the senders. With more scrutiny, we also can see a substantial time gap (forged sections are often prepared ahead of time), and 'out of place' routing list (we can trace route of the e-mail) in the fake e-mail headers and e-mail logs.

D. Blacklist

The IP addresses that are blacklisted are often those (machines) that have been observed by the DNSBL operators to be sending spam e-mails, hosting spammers or by policy, those that are not allowed to send e-mails directly. The list operators actively monitor the internet for reports of e-mail traffic from various sources sending the spam and issue their list. This list is then used by mail servers, ISPs and organizations to block out the spam sources.

DNSBLs have different criteria for including IP addresses. They list one or more records of verified spam services, unsolicited bulk e-mail sources, 3rd party exploits like proxies, open relays, mail servers that have a high spam-to-legitimate-mail ratio or IP addresses of hosts which do not reply to (DNSBLs maintainer's) test e-mails [9,10,11,12]. Similarly they have different ways of categorizing these IP addresses. Some DNSBLs use different domains (called query domains) to separate IP addresses based on the reasons they are blacklisted. One domain may contain IP addresses for known spammers while other contains open relays. DNSBLs

return a reason code for a DNSBL query (e.g.127.0.0.2-6). Even if all the IP addresses have been listed in one domain, the reason code will explain what kind of spammer it is. For example, a code of 127.0.0.2 might represent a verified spammer, while a code of 127.0.0.4 might represent illegal 3rd party exploits (e.g. proxies) [10]. This type of categorization allows a service provider to go for particular service it needs.

The delisting process of the DNSBLs varies widely: Some wait for proper de-listing requests, some use automated expiry timer and others wait until spamming is halted completely. In general the DNSBL maintainers mostly look for accountability from the administrator of the IP address, who has to request and respond to a de-listing message, before delisting him.

When an ISP (Internet Service Provider) is blacklisted, so are all the clients associated with that ISP. Thus, numerous clients can be potentially affected due to a handful of spammers. DNSBLs are not faultless. Some maintainers list e-mail sources without thorough investigations, while others simply are not efficient enough to catch majority of the spammers. Some DNSBLs are not updated frequently enough to maintain a continued effectiveness. The latency period (lag time between the spammer spamming and the administrator identifying them) can give ample time for the spammers to send millions of e-mails.

E. DNSBL Query

A recipient MTA or MUA can use its local BL cache or query one or more global DNSBL's databases in real-time to filter out the spam. DNSBL sites use the DNS protocol to accept queries and provide replies. Typically a mail server will construct a special DNS query including the address of the incoming SMTP connection, and kill the connection if the DNSBLs server returns a special IP address (reason code) implying that the sender is in their list. DNSBL client can be built into MTA or can be a separate program which will automatically query a DNSBL site/domain while filtering. Performing a DNSBL lookup of a mail server located at 1.2.3.4, involves a DNS query of "4.3.2.1.DNSBLs". Here, IP address is reversed like in the in-addr.arpa mechanism [21]. As described in section 3, since each DNSBL has its own set of responses and their meanings, one has to be aware of these while performing the DNSBL query.

IV. SYSTEM ARCHITECTURE

Figure 3 shows the high level architecture of the e-mail system with the cache implementation. The local caches are initially empty and they get filled up depending on the responses of the DNSBL queries. The IP addresses are first compared to the list in the local caches. If a sender is on the WL, further searches in the BL cache and an external DNSBL query is not required; it will be sent directly to the user's mail box. However, if it is not found in the WL cache, the entry is searched at the BL cache; if it is found here, the e-mail is rejected or dropped or returned. If the IP address is not found in the caches, an external DNSBL query is carried out with that IP address to determine if it is listed or not in one or more DNSBLs. In our work, the DNSBL query has been carried out using the popular DNSBLs [15]. An IP address is added to the WL cache for all 'false' or negative responses from the DNSBL and to the BL cache for all 'true' or positive responses.

The journey of an incoming e-mail (from the time it enters the mail server to the time it is listed in the cache) in our design is shown by the flow chart in Figure 4. IP addresses of the senders and relaying machines are extracted from the log file (for the incoming e-mails to our mail server). The log file includes all the connection attempts to the mail server (both regular e-mails and the e-mails rejected by the local host).

The cache search process is summarized below:

```
Search(String IP) {
 SearchWLcache(String IP){}
  (if !foundInWLcache) {
    SearchBLcache(String IP){}
      { (if !foundInBLcache)
          QueryDNSBL (String IP){
               if( response = positive) write to the BLcache;
              if( response = negative) write to the WLcache;
          }
      }
  }
}
```

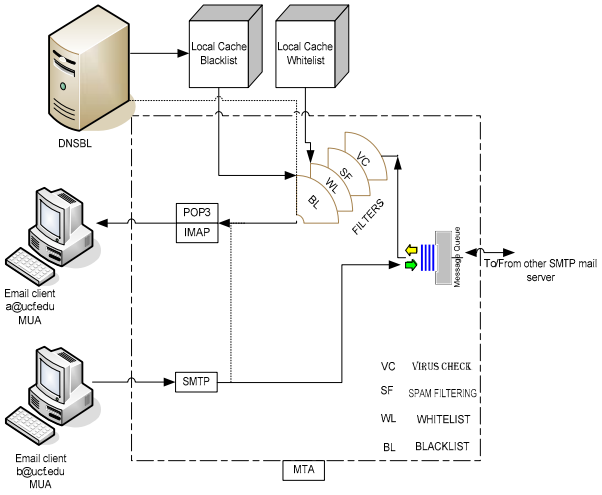

Fig. 3: E-mail system with Cache Implementation for Spam Control

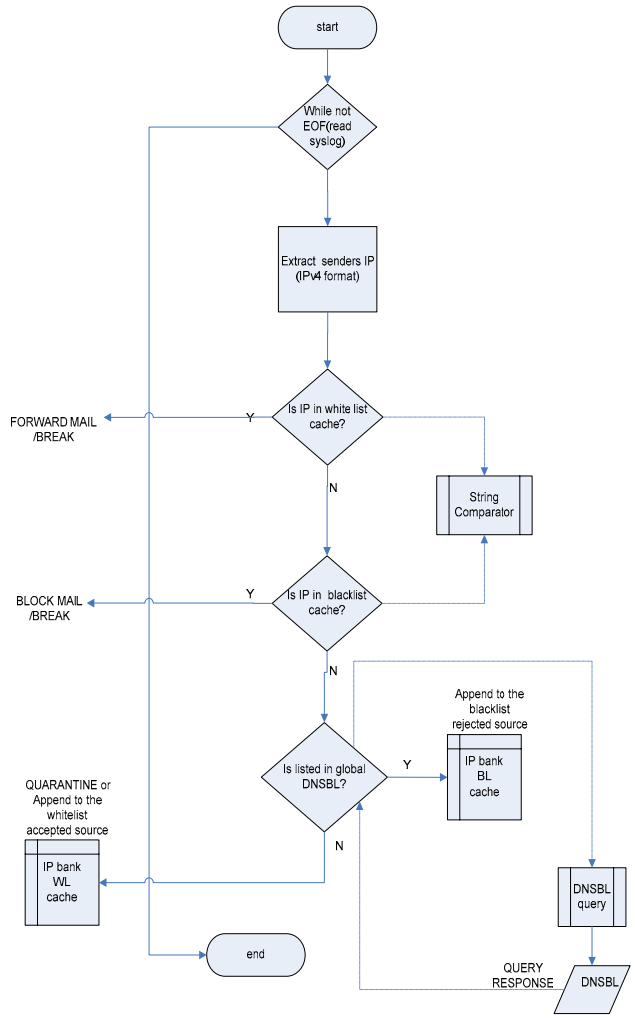

Fig. 4: Flow Chart of Cache Implementation for Spam Control

V. Implementation and Experimental Results

The simulation code was developed in JCreator IDE and compiled with JDK1.5. The program reads and compares the data set (*IP addresses as string data type)* with those that in the local caches. The sender's IP addresses are extracted in a chronological order from the e-mail log. WL and BL caches are implemented using the secondary (hard drive) memory. InetAddress class from the package java.net provides the method to resolve host names to their IP addresses and vice versa. Ordered or sorted list for the cache and the implementation of efficient list search algorithms are not used because we are focusing our work on the network traffic only. Also the latency affect is not considered for a simple reason that different computers run with different speed.

A batch (for the simulation purpose) is defined as the total number of IP addresses (test data) sent for the DNSBL query in a single run/execution of the code. Table 1 lists the DNSBLs used for the simulation purpose. Combined zone (domain) in a particular DNSBL are used wherever possible so that they reduce the number of queries required. The tests are carried out within a week of the collection of the IP addresses from the log file.

All the (external) senders attempting a connection to our department's mail server were considered. We did not use the incoming e-mails from of the mail server users. It was discovered that around a half of the total connection attempts were made from unique machines.

Table 1: List of DNSBLs used for the Simulation

Test No.	DNSBL used	DNSBL query
1	zen.spamhaus.org list.dsbl.org dnsbl.sorbs.net bl.spamcop.net t1.dnsbl.net.au no-more-funn.moensted.dk cbl.abuseat.org dnsbl.tqmcube.com	Individually queried to each server
2-4	zen.spamhaus.org list.dsbl.org dnsbl.sorbs.net bl.spamcop.net	Simultaneously queried to all servers
5	Same as in Test 1	Simultaneously queried to all servers
6	Same as in Test 2	Simultaneously queried to all servers

Test 1: Aggressiveness of the DNSBLs

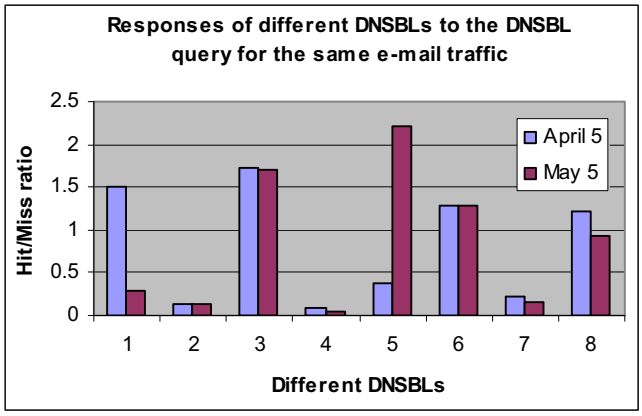

Fig. 5: Different DNSBL's query hits to miss ratio, compared with the senders in the same e-mail traffic. Senders IP addresses were collected during a work day in March (2007)

Figure 5 shows how the coverage of each DNSBL varies: there are some highly conservative lists (2, 4, and 7) which *list* much fewer spam sources than other aggressive lists. The number and the type of spam sources each domain lists vary widely. While some DNSBLs overlap known spam sources, open relay lists and other categories, others have a separate list for each of them. The results of these DNSBL queries (*hits and misses*) for the same dataset, change with time because of

the dynamic nature (listing and delisting of the IP addresses) of the DNSBLs.

Test 2: Local Cache Implementation and the Total Cache Hits

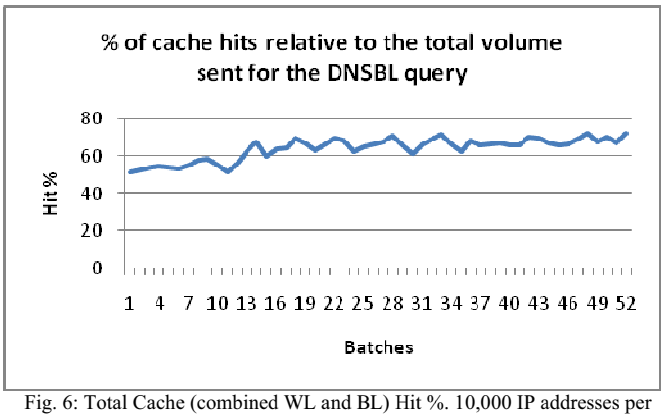

Fig. 6: Total Cache (combined WL and BL) Hit %. 10,000 IP addresses per batch were sent for the DNSBL filtering. The IP addresses were collected during the 2nd week of March (2007). Four DNSBLs were used for this test.

Figure 6 shows the percentage of *hits* (in combined WL and BL caches) per batch for a uniform data set (in volume). The senders found in the local caches do not have to travel to the external DNSBL server for filtering purposes. We can see that up to 70% e-mail sources can be filtered locally after several batches of test runs. This means seven out of ten incoming e-mails do not have to go through an external DNSBL query test for the spam filtering purpose. For a large traffic, this brings about a significant increase in the availability of bandwidth for purposes other rather than the communication with the DNSBL servers.

Test 3: Local Cache Implementation and the Total Cache Hits

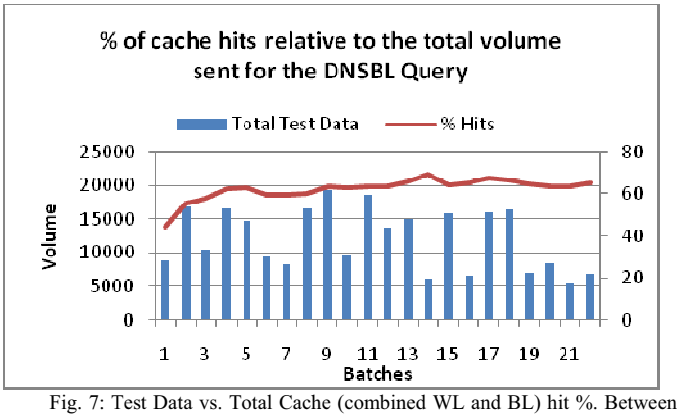

Fig. 7: Test Data vs. Total Cache (combined WL and BL) hit %. Between 5000-20,000 IP addresses (random) per batch were sent for the DNSBL query. The IP addresses were collected during the 1st week of March (2007). Four DNSBLs were used for this test.

Figure 7 shows the results when a random number of entries (between 5,000 and 20,000 IP addresses) per batch were tested. The cache hit percentage is around 70%. The random entries were tested to see if the volume of the test data (IP addresses) affected the results.

Test 4: Local Cache Implementation and the Total Cache Hits

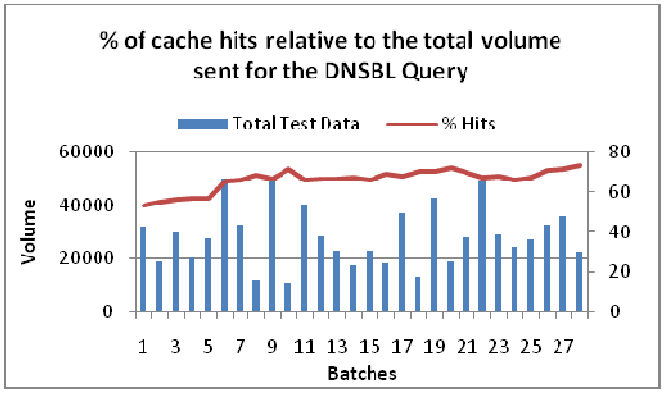

Fig. 8: Test Data vs. Total Cache (combined WL and BL) hit %. Between 10,000-50,000 IP addresses (random) per batch were sent for the DNSBL query. The IP addresses were collected during the 2nd week of March. Four DNSBLs were used for the DNSBL queries

Figure 8 shows the results of testing a larger set of test data, again using a randomly generated number between 10,000 and 50,000. The cache hit percentage goes over 70% during the latter batches with the increased volume of the data set and subsequently the cache size.

Test 5: Comparison of Usage of Different Numbers of DNSBLs

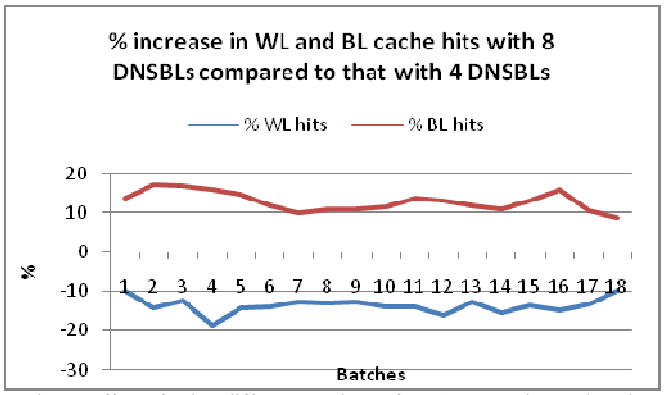

Fig. 9: Effect of using different numbers of DNSBLs on the Cache Hits. Between 10,000-50,000 IP addresses (random) per batch were sent for the DNSBL query. The IP addresses were collected during the 2nd week of March. Responses of queries with eight DNSBLs and four DNSBLs were collected with the same data set.

With increased DNSBLs usage, one would typically expect more IP addresses being blacklisted (more spam sources caught). This means more senders are added to the BL cache and eventually this leads to an increased chance of local BL hits. To verify this assumption, we repeated Test 4, using eight popular blacklists (instead of four) and then calculated the % increase in the hits for the WL and the BL separately. Figure 9 shows that the BL cache hits are increasing (positive % increase) and the WL cache hits are decreasing (negative % increase) with the batches. This is because the senders (spammer machines) would have less chance of missing the BL cache. Our research shows that the ISPs use anywhere from a couple to tens of DNSBLs to filter out the spam.

Test 6: Total Cache Hits

We collected senders (IP addresses) from a total of five day's connection attempts made to our mail server (around 120,000 unique senders) and tested them all in a single batch. Four DNSBLs were used and the total hit percentage in the local caches was found to be 75.27%. This reiterates the fact that with the local cache implementation, around three quarters of total connection attempts can be resolved locally (for concluding if an incoming e-mail is a spam or not).

Preliminary results from the data collected from a different e-mail log/mail server, shows a higher percentage (>80%) of cache hits. This is possibly due to a greater spam attack on this server and large number of e-mails sent from the same sources.

VI. CONCLUSIONS AND FUTURE WORK

Our results have shown how the cache implementation lessens the overall requests and responses that need to travel over a network for the DNSBL resolution. This factor, along with the apparent reduction in the search time for locating the source of an e-mail in the local caches compared to that over the network, can be quite significant for a large e-mail traffic. We also have a better control over the cache entries, which can be updated or modified as per our requirements.

To increase the practicality of the tests discussed, they should be carried out with the logs from different mail servers and (the data tested) soon after they are generated. Catching every single spam source is not possible, thus, the focus should be to find ways to catch most of them. The number of DNSBLs used, affect the probability of a spam source being caught (also the number of *hits* at the local caches). While a higher number of DNSBLs protects a mail server better (same with the number of spam filters), we face trade-offs among the security, the cost and the speed. Only at the expense of resources and latency, greater number of spam sources are caught. A massive cache size is also not advisable; the search time at the cache can increase enormously. The entries in the caches also need to be updated frequently (listing and delisting of the IP addresses). Selection of DNSBLs to be used would be dependant on the kind of service the administrator wants.

As a continuation of this work, we plan to use e-mail logs from at least three mail servers (for a comparative analysis) and work on the mechanisms to update the local cache lists regularly (for increasing the verifiability of the lists -similar to what some DNSBLs do). Faster tools (e.g. VHDL) for simulating the results would be used. Though the latency effect is hard to measure (because of the varying speeds of the computers) some experimentations with the search time will be attempted.

REFERENCES

[1] A. Shaikh, R. Tewari, and M. Agrawal, "On the effectiveness of DNSbased server selection," in Proceedings of the IEEE INFOCOM, Anchorage, AK, April 2001.

[2] J. Jung, E. Sit, "An empirical study of spam traffic and the use of DNS black lists", Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, Taormina, Sicily, Italy, Pages: 370 – 375, 2004.

[3] B. Laurie and R. Clayton, "proof-of-work" proves not to work". In Proceedings of the Workshop on Economics and Information Security, Minneapolis, MN, May 2004.

[4] M. Tran and G. Armitage, "Evaluating The Use of Spam-triggered TCP/IP Rate Control To Protect SMTP Servers" Australian Telecommunications Networks & Applications Conference 2004 (ATNAC2004), Sydney, Australia, December 8-10 2004.

[5] L. Pelletier, J. Almhana and V. Choulakian, "Adaptive Filtering of SPAM", Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04).

[6] G. González-Talaván, "A simple, configurable SMTP anti-spam filter: Greylists ", Computers & Security Volume 25, Issue 3, May 2006, Pages 229-236

[7] Bhagyavati, N. Rogers and M. Yang, "E-mail filters can adversely affect free and open flow of communication", ACM International Conference Proceeding Series; Vol. 58 archive, Proceedings of the winter international symposium on Information and communication technologies

[8] A. Ramachandran, D. Dagon, N. Feamster, "Can DNS-Based Blacklists Keep Up with Bots?" College of Computing, Georgia Institute of Technology, CEAS 2006 - Third Conference on E-mail and Anti-Spam, July 27-28, 2006.

[9] SpamCop Blocking List. Link: http://www.spamcop.net/. Last visited: December, 2006.

[10] The Spamhaus Project. Link: http://www.spamhaus.org/. Last visited: December, 2006.

[11] Distributed Sender Blackhole List. Link: http://dsbl.org/main. Last visited: December, 2006.

[12] Open Relay Database. Link: http://ordb.org/. Last visited: December, 2006.

[13] The syslog Protocol. http://tools.ietf.org/html/rfc3164#section-4.1. Last visited: December, 2006.

[14] L. Zhang, J. Zhu, T. Yao, "An evaluation of statistical spam filtering techniques". ACM Transactions on Asian Language Information Processing (TALIP) archive Volume 3, Issue 4 (December 2004) table of contents Pages: 243 - 269; Year of Publication: 2004.

[15] Blacklists Compared.
Link: http://www.sdsc.edu/~jeff/spam/Blacklists_Compared.html.
Last visited: February, 2007.

[16] A. Ramachandran and N. Feamster. "Understanding the Network-Level Behavior of Spammers". Georgia Tech TR GT-CSS-2006-001.

[17] P. Danzig, K. Obraczka and A. Kumar, "An analysis of wide-area name server traffic: A study of the Internet domain name system," in Proceedings of the ACM SIGCOMM, Baltimore, MD, Aug. 1992, pp. 281–292.

[18] D. Cook, J. Hartnett, K. Manderson and J. Scanlan, "Catching Spam Before it Arrives: Domain Specific Dynamic Blacklists", Fourth Australasian Information Security Workshop (AISW-NetSec 2006), Hobart, Australia. http://crpit.com/confpapers/CRPITV54Cook.pdf.

[19] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian approach to filtering junk e-mail". In Learning for Text Categorization -- Papers from the AAAI Workshop, pages 55--62, Madison, Wisconsin, 1998

[20] F. D Garcia, J-H Hoepman and J. v Nieuwenhuizen, "Spam Filter Analysis", 2004, paper presented to 19th IFIP International Information Security Conference, Toulouse, France.

[21] Domain Name System (DNS) Security. Link:
http://www.geocities.com/compsec101/papers/dnssec/dnssec.pdf. Last visited: April, 2007.

[22] A plan for spam. Link: http://www.paulgraham.com/spam.html. Last visited: December, 2006.