Alternate Architecture for Domain Name System to foil Distributed Denial of Service Attack

Journal of Internet Banking and Commerce, April 2006, vol. 11, no.1
(http://www.arraydev.com/commerce/jibc/)

P.Yogesh, Senior Lecturer, Department of Computer Science and Engineering, College of Engineering, Anna University, Chennai-600025, India.
Dr.A.Kannan, Assistant Professor, Department of Computer Science and Engineering, College of Engineering, Anna University, Chennai-600025, India.

Web: http://www.annauniv.edu/
Email: yogesh@annauniv.edu kannan@annauniv.edu

P.Yogesh received his B.E and M.E degrees in Computer Science and Engineering from Madurai Kamaraj University, India,and is currently working as a Senior Lecturer and also working toward the Ph.D. degree at the Department of Computer Science and Engineering, College of Engineering, Anna University, Chennai, India. His main research interests span various areas in computer communication networks, including high performance networks, wireless and ad hoc networks and multimedia communication.

A.Kannan received his M.E and Ph. D degrees in Computer Science and Engineering, from Anna University, Chennai, India. He is working as an Assistant Professor in the Department of Computer Science and Engineering, College of Engineering, Anna University, Chennai, India. His current research interests focus on Database Management Systems, Data Mining and Warehousing, Artificial Intelligence and Computer Networks.

Abstract

The Domain Name System is an important part of the Internet infrastructure and maps symbolic Domain Names to IP addresses. The DNS is a hierarchically arranged distributed database. At the top of the hierarchy is the root. The root is a single point of failure in the DNS architecture. It has been subject to variety of Denial of Service (DoS) attacks. Eliminating the root from this architecture eliminates the single point of failure. This involves storing the addresses of the top-level domain servers at the name servers, so that they can be reached without going through the root. In this paper we propose two architectures, both capable of foiling the DoS attack. The architectures differ in the capabilities of the clients and servers, and provide different cost-benefit tradeoffs. It has been found that a scheme that avoids a root server for name resolution and includes caching capabilities at the client itself, reduces bandwidth requirements, and improve! s response times, resilience to DoS attacks.

1. Introduction

The Domain Name System (DNS) is a distributed database of information about hosts on the Internet.

Its function is to make that information available over the entire Internet. â€œDistributedâ€ signifies that the information in the DNS is spread over multiple computers. The Domain Name System provides a means of retrieving that information from anywhere on the network.

Through the 1970s, just a few hundred hosts populated the ARPANET, an experimental wide-area computer network across the United States. All the information about these hosts was stored in a single file, HOSTS.TXT. The Network Information Center (NIC) of SRI, an independent, non-profit R & D organization in Menlo Park, California maintained it. Changes for the file were submitted to the NIC, and compiled into a new HOSTS.TXT once or twice a week [1]. As the number of hosts grew very large, problems arose with HOSTS.TXT. The network traffic and processor load on SRI-NIC was becoming uncontrollable. There was no authority over host names as well. There was nothing to prevent someone from adding a host with a conflicting name and breaking the entire system. Another problem was maintaining consistency of the file across the entire network. Changes to the hosts or additions were being made before a copy of the file could reach all of the ARPANET.

This system of a flat name space did not scale well. A name space system should ensure the uniqueness of names, allow local administration of data, and be available to the rest of the Internet. Keeping data up-to-date would be much easier then. Paul Mockapetris proposed a solution to this problem by designing, the Domain Name System. [2]. This new architecture involves a hierarchical name space. In this design, uniqueness of names is easily ensured. Keeping data up-to-date is much easier with local management and implementation of zones of authority.

The Domain Name System is a distributed hierarchical system for resolving host names into IP addresses. The DNS has a root domain at the top of the hierarchy and directly below are the top-level domains. The root of the tree has no name. All siblings of a domain must have unique names. Children of a domain are called subdomains of the parent [3]. The structure of the DNS name space is similar to the structure of the file system on a computer, with the root at the top. In DNS, the root is written as a single dot "." in text. Each partition of the database represents a domain. Each partition can be divided further into subdomains. A domain also has a domain name, which identifies its position in the database. In DNS, the full domain name is the sequence of labels from the domain to the root, with dots separating the labels. In a computer file system, the names of the directories are read from the root to leaf. Making names hierarchical eliminates the probl! em of name collisions. An organization's registered domain name is unique, which makes them free to choose any names they wish within their domains. Whatever name they choose, it will not conflict with other domain names, since it will have their unique domain name tacked onto the end of it.

At the top of the DNS hierarchy is the root. The data in the root zone is replicated in 13 computers around the world. These servers contain the key records, which store the addresses of the top-level domain names along with their IP addresses. The IP addresses of the root servers are well known to all resolvers. Resolvers contact one of these servers for information about top-level domains. In the absence of more specific information, name resolution begins at the root. Since these servers are crucial to successful domain name resolution, they have been repeatedly subject to a kind of attack known as the distributed denial of service attack.

Attackers take control of several machines on the Internet and use them to send a flood of queries to the root servers to overwhelm them. Fortunately, the root servers themselves handle only a small fraction of domain name queries. This is because the name servers that query them, cache the responses, generally for 48 hours. Only if the root servers are still unavailable would the average user be affected. If the attack succeeds, web, email and other services that depend on the DNS would gradually become unusable. In this paper we propose two architectures to overcome the problems of the existing DNS. In the first modified architecture, the root server is eliminated. This requires storing the IP addresses of the top-level domains (TLD) in the name servers. The second architecture also eliminates root server. In addition, the client module is able to perform caching and recursive querying.

2. Related Work

Paul Mockapetris designed the DNS of the Internet [2]. Paul Albitz and Cricket Liu helped to understand the working of the DNS through an Open Source System tool called BIND [4]. BIND is one of the most popular tools for managing domains on the Internet. Milton Mueller explained the shortcomings of ICANN, the organization that establishes the policies that govern the domain name system. The issues in the internationalization of the DNS [5] have been touched upon by Vinton G Cerf

while tracing the evolution of Internet Technologies. Dan Pei and Lixia Zhang discussed the possibility of a DoS attack on the DNS [6] and means to secure the System against such attacks. They also provided a classification scheme of the various threats and countermeasures. Stephen Cherry gives a description of a real DoS attack in. Chakrabarti and Manimaran have classified various possible attacks over the Internet into four categories [7] namely DNS hacking, routing table poison! ing, packet mistreating and DoS.

Gerco Ballintijn, Maarten van Steen and Andrew Tanenbaum have proposed [8] a new scheme to improve both scalability and usability in naming replicated resources on the web. Jung, Sit, Balakrishnan and Morris have analyzed the performance of cached DNS [9] with respect to the distribution of name popularity and the distribution of TTL (Time To Live) values. Richard Liston, Sridhar Srinivasan and Ellen Zegura [10] have investigated the degree to which metrics for wide-area DNS performance such as mean response time, number of servers contacted, and root and generic top-level domain server performance, differ across locations in the Internet.

Garber has discussed the Distributed Denial of Service (DDoS) attack over the Internet and the potential of these attacks to shut down the web sites and thereby the business involved [11]. Michael F Schwartz and Calton Pu have discussed the general framework for gathering and harnessing widely distributed information in a diverse and growing internet environment [12].

3. Problem Description

The root name servers have information about the authoritative servers for the top-level domains. They are at the top of the DNS hierarchy. If no other information is available, resolution has to begin at the root. Thus, they are crucial to the operation of the Internet. If they were unavailable for an extended period, all resolution on the Internet would fail. To alleviate the load on the root servers, the intermediate servers cache responses so that further requests for information about the same domain can be serviced from the local cache. **Fig. 1. depicts the existing architecture of the DNS.**

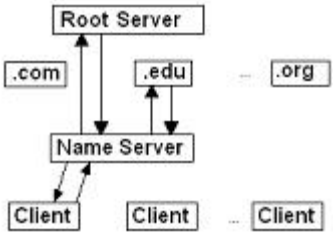

Fig. 1. Existing Scheme of DNS

Restrictions in the size of the UDP queries on the DNS limit the number of root servers to 13. These are identified by a letter prefix that varies from A to M. However, there are more than 100 root servers in operation now. They are located in every part of the globe. This is accomplished through a technique known as anycasting. Anycasting is a technique by which the same IP address is assigned to multiple computers on the Internet [13]. They are used as mirrors to provide the same service at different points in the network. Requests are routed to the nearest mirror. Three of these root servers are in India. They are the F-root server at Chennai, I- root server at Mumbai, and the K-root server at Noida (Delhi). All root servers are on an equal footing. In the past, the zone file was distributed to the other operators from the A-root server. The distribution of the zone file has since been changed. All root severs load a file that has been produced by Inte! rnet Assigned Numbers Authority (IANA). After being produced by the IANA, the file is stored on a number of distribution servers. The IP addresses and locations are hidden to make them less susceptible to malicious attacks and hence they are often known as hidden servers. The root name server operators fetch the file from these servers in a secure fashion. Each operator then distributes the file to the servers they operate in the manner they choose.

The size of the root zone file is small and it changes very infrequently. To reduce the loads on the root servers, the records they serve have a TTL value of 48 hours. In spite of anycasting and the large TTL values for the entries, load at the root servers remains high; close to 1,00,000 queries per second. To reduce the load on the name servers, recent versions of the name server software cache negative responses as well. It has also been found that a substantial number of queries are those that are trying to perform inverse lookup on the private IP address range.

Inverse lookup is when programs try to find the domain name from a particular IP address. They form a significant part of the DNS traffic. The private IP address range is meant for use on intranets and should never leak into the Internet. However, such leaks happen due to misconfigured clients and servers. Moreover, some of the older name-server software donâ€™t cache negative responses, resulting in such traffic repeatedly reaching the root servers. To overcome this problem, servers known as blackhole servers or prisoners are used. When such queries arrive at the root servers the root servers respond with the IP address of the blackhole servers. This prevents the name servers from retrying the same query with the root server. They contact only the blackhole servers thereafter. The blackhole servers in turn answer such queries with authoritative "doesnâ€™t exist" replies. The client may retry the query, but now the local name server, which has cached the res! ponse from the root, contacts the blackhole server, without going through the root server.

To start resolution, when the cache is empty the resolver needs to know the IP address of the root. These addresses are hard-coded into the resolver library routines. The resolvers use the Round Trip Time (RTT) as a metric to identify the nearest root server. At the beginning, these servers are assigned random RTT values lower than any real world value, so that each server is tried at least once.

3.1 Alternate Routes

ICANN, a nonprofit organization, created by the US Department of Commerce is the governing body for the top-level domains of the Internet. It makes decisions regarding addition of new Top Level Domains (TLD), the ownership of these TLDs, and oversees the domain-name registrars that sell and control domain names. The addition of new TLDs has not matched the market demand. Some people believe that this paucity is deliberately maintained to favor large corporations. For this, as well as ideological reasons numerous people have started running their own root servers. Prominent among those are the roots of AlterNIC (stopped), Pacific Root and New.Net. With these operators, we can register names in domains such as. glue, .geek, .parody, .corp, .fam, .per, .web, .job, .lib, .ppp, .sat, .www, .men, .not, .ngo, .tech, .game, .mp3, .med, .club, .law, .family, .sport as well as names in other European Languages. Setting up a root server is technically trivial. The d! ifficulty is in getting people to configure their name servers to query these servers. Some operators such as New.net have tried to overcome this problem by extending Internet Corporation for Assigned Names and Numbers (ICANN)â€™s root rather than modifying them, i.e. they retain the entries in the ICANNâ€™s root as they are and add their own entries. They then persuade Internet Service Providers to have their name servers refer to these servers as the root servers. They also provide browser plugins that modify the resolvers of web browsers so that they query New.netâ€™s servers bypassing the local name server as well as the ICANN administered roots.

3.2 Denial of Service

As explained in the previous section, the root server is crucial for successful name resolution. So most Denial of Service (DoS) attacks target the root server though other servers can be targeted as well. **One popular method of DoS known as the Distributed Denial of Service (DDoS) attack is explained as in Fig. 2.**

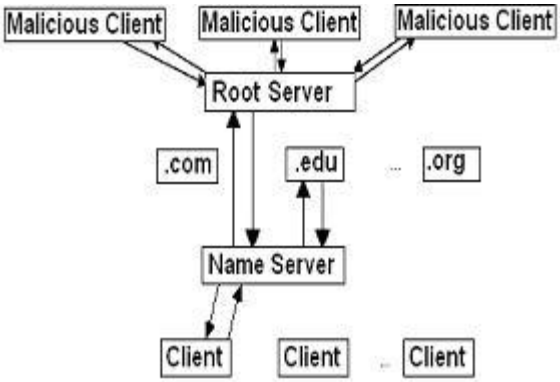

Fig. 2. Distributed Denial of Service (DDoS) Attack

The root servers contain the key records that store the top-level domain names along with the IP addresses of their authoritative servers. The IP addresses of the root servers are well known to all name servers. The name servers are programmed to contact one of these servers for information about top-level domains. To accomplish DoS, attackers take control of several machines on the Internet and use them to send a flood of queries to the root servers to overwhelm them.

Another method takes advantage of negative responses i.e., responses that indicate the domain name in the query cannot be resolved. Sending back the negative response for a DNS name that could otherwise be resolved, results in a DoS for the client wishing to communicate with the domain in the query. The other way DoS is accomplished is for the rogue server to send a response that redirects the client to a different system that does not contain the service the client desires.

4 Proposed Scheme

4.1 DNS without Root Server

In the first modified architecture, the root server is eliminated. This requires storing the IP addresses of the top-level domains in the name servers. The advantage of this scheme is that one query-response sequence is eliminated before a request from a client is serviced. It also provides better resilience to DoS attack since the single point of failure in the architecture is eliminated. The attacker now has to target many more servers to achieve denial of service. The drawback is a slight increase in memory requirements at the name server since it has to store the IP addresses of all the top-level domains rather than that of the root server alone. This increase is of the order of a few kilobytes and is negligible.

All modules other than the name server are same as before. The name server has been modified to contact the servers of the top-level domains directly. The root server is absent. To achieve DoS, the malicious client now has to send strings to all the top-level domains, which necessitates additional processing. The number of strings received per server in a given period is reduced allowing query resolution to proceed normally even while an attempted DoS attack is on. This is depicted as in fig 3.

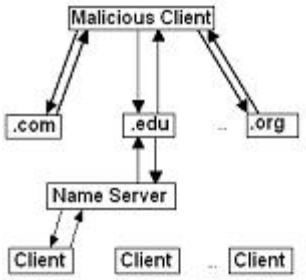

Fig. 3. Proposed Architecture of DNS

4.2 Rootless DNS with Client Caching

The root server is not present in this architecture as well. In addition, the client module is able to perform caching and recursive querying. This helps to cut down traffic between the client and the name server since a repeated request is serviced at the client itself. Security is further enhanced since most name resolution can proceed when in addition to the top-level domain servers; the local name server itself is under attack. The other modules are the same as in the previous case.

5. Implementation

To demonstrate the DoS attack, a DNS architecture is setup by creating separate modules for the DNS client, the DNS server, the com, org and edu top level domain servers and the root server. Query resolution proceeds as follows: the client sends the domain name that it wants resolved into an IP address to its local name server. The name server checks its own cache to see if the sought mapping is available. If the mapping is available, it supplies the mapping from its cache itself. Else, it contacts the root server for the IP address of the corresponding top-level domain. From this response, it contacts the top-level domain for the IP address of the domain. If such a domain exists, the top-level domain returns its IP address, which the name server forwards to the client. The name server also caches this response, so that further requests for the same IP address can be supplied from its cache itself. The client uses this IP address for communication with t! he host. This client module is part of many applications that need to communicate over the Internet. So its implementation is kept as simple as possible. Its capability is limited to formulating a request and forwarding it to a local name server. Specifically, it is not programmed to cache responses. Hence, it is also known as a stub resolver. Thus if some application needs the IP address of the same domain name, the client has to make a fresh request to the local name server.

DoS attack occurs when malicious clients flood the root server with a large number of random queries. Normally, the attackers take control of many hosts and use these hosts to flood the root servers to cause maximum disruption. The root servers are tied up servicing these queries that requests from legitimate clients get poor or no response.

5.1 DNS Client

The DNS Client possesses a domain name and requires the corresponding IP address. The client that performs this function is also known as the resolver. Since the resolver has to be part of many applications, it is normal practice to keep it as simple as possible. Its capability is limited to formulating a request and forwarding it to the local name server. It does not perform any function such as caching of responses. Hence, it is also known as the stub resolver. It merely accepts a domain name from the application and forwards it as a request to the local name server. It then accepts the response from the name server and passes it on to the application.

5.2 Name Server

The name server accepts requests from the client and returns the corresponding IP address. It does this through a process known as recursive querying. As soon as it receives a request, it searches its own cache to see if that domain name is already present. If present, it supplies the IP address to the client immediately. If not, it contacts the root server for the IP address of the corresponding top-level domain. With this IP address, it contacts the top-level domain for the required domain. It passes the reply to the client and caches the response. Any further requests for the same IP address are serviced from the cache.

5.3 Root Server

The root server contains the IP addresses of the authoritative servers of the top-level domains. There are more than 250 top-level domains in the DNS. These include the country-code domains as well as the generic domains. Any domain name can be resolved by following the list of authoritative servers starting with the root domain. When the name servers contact the root servers, they respond by providing the IP addresses of the corresponding top-level domains.

5.4 Com, Org, Edu Domain Servers

These servers contain the records for the domains registered under them. Their function is similar to that of the root servers, namely to provide the IP address of the record that matches the domain name in the request.

5.5 Malicious Client

The malicious client generates random strings and sends them to the root servers seeking their IP addresses. The root server is so busy with servicing these requests that at the time of this DoS attack, it is unable to respond to legitimate queries from name servers.

5.6 Algorithms

Algorithm1 simulates the DoS attack over DNS with root server. From this we observe that when the DoS attack is taking place, the root server is unable to respond to queries from the name server and once the DoS attack stops, query resolution proceeds normally. Algorithm 2 simulates the DoS attack over DNS without root servers and cached name servers. This algorithm proves that even while the DoS attack is taking place, DNS is able to resolve the queries successfully. Reason is malicious clients are not able to disturb the functioning of the TLDs because of their large numbers. Algorith 3 simulates the DoS attack over DNS without root servers and cached clients. Here also DNS attack is thwarted, and as far as possible, client resolves the requests locally.

Algorithm 1 - DoS Attack over DNS with root servers
Input: Domain Name to be resolved
Steps:
1. Client sends request to name server
2. Name server checks its cache for the domain name. If a matching entry exists, the corresponding IP address is send to the client. Else, the name server asks the root server for address of the corresponding authoritative TLD Server.
3. Name server asks the TLD Server for the IP address of the domain name.
4. Name server caches the response and returns reply to the client.
5. Malicious client begins attack by flooding the root server with random strings.
6. While the DoS attack is taking place, the root server is unable to respond to queries from the name server.
7. Once the DoS attack stops, query resolution proceeds normally.
Output: IP address corresponding to the input string if DoS does not take place, otherwise DNS fails.

Algorithm 2 â€“ DoS Attack over DNS without root servers
Input: Domain name to be resolved.
Steps:
1. Client sends request to Name Server
2. Name Server checks its cache for the record. If a match exists, the IP address is send to the client. Else, the Name Server asks the TLD server for the IP address of the domain.
3. Name Server returns reply to the client.
4. Malicious client begins attack by flooding the top-level domain servers with random strings.
5. Even while the DoS attack is taking place, query resolution proceeds normally.
Output: IP address corresponding to the domain name

Algorithm 3 â€“ DoS Attack over DNS without root servers and Cached clients
Input: Domain name to be resolved.
Steps:
1. Client receives string to be resolved.
2. Client checks its cache for the query. If a matching entry exists, the corresponding IP address is send to the application. Else, the client asks the name server for the IP address of the input string.
3. Name server returns reply to the client.
4. Malicious client begins attack by flooding the top-level domain server with random strings.
5. Even while the DoS attack is taking place, query resolution proceeds normally.
Output: IP address corresponding to the input string.

5.6 Performance Analysis of Root Servers in the Internet

Table I gives the average time taken to reach each of the 13 root servers. These are the averaged values of experiments performed over different days and at different times of the day. There are substantial delays in the case of some servers and worse; some servers are unavailable some or all the time. A name server that contacts one of these servers has to make another request and wait for its response.

Table I
Average Response Times of Root Servers

Server	Operator	IP Address	Response Times (ms)
A	VeriSign Naming and Directory Services	198.41.0.4	403
B	Information Sciences Institute	192.228.79.201	Unreachable
C	Cogent Communications	192.33.4.12	363
D	University of Maryland	128.8.10.90	423
E	NASA Ames Research Center	192.203.230.10	Unreachable
F	Internet Systems Consortium, Inc.	192.5.5.241	25
G	U.S.DoD Network Information Center	192.112.36.4	Unreachable
H	U.S. Army Research Lab	128.63.2.53	424
I	Autonomica/NORDUnet	192.36.148.17	234
J	VeriSign Naming and Directory Services	192.58.128.30	303
K	Reseaux IP Europeens – Network Coordination Centre	193.0.14.129	48
L	Internet Corporation for Assigned Names and Numbers	198.32.64.12	363
M	WIDE Project	202.12.27.33	259

It is seen that the average time to reach a root server in the case of available servers is 285ms. Average database access times (command processing time + rotational latency + seek time) are about 6ms. So, the time elapsed before a response in case a functional server is contacted is 291ms. However, 3 out of the 13 servers are unreachable and there is a 3 in 13 chance that it is one of these that the name server will try to contact first. In such cases, the name server will have to make another request to a working root server to get the address of the top-level domain server. Accounting for this, the average time elapsed before the name server gets the address of the top level domain server is

360ms. This delay is averted in the proposed architectures. The price to be paid in the form of additional memory requirements is about 10KB (10,150 bytes) at each name server. This price is negligible considering the improved performance and security.

Both the proposed schemes provide an improvement of about 360ms per request that would otherwise have gone to a root server. Response from a local name server is practically instantaneous. Therefore, there is no further performance gain to be had in caching the responses at the client. However, the resilience of the architecture is further improved, since name resolution can now proceed even if the local name server itself is under attack or unavailable for other reasons.

6. Conclusions and Future Work

This work modeled the existing architecture of the DNS and one of the most common attacks on the DNS namely the distributed denial of service attack. It suggests means of foiling these attacks through alternate architectures. Two architectures have been proposed, both capable of foiling the DoS attack. The architectures differ in the capabilities of the client and server, and provide different cost-benefit tradeoffs. It has been found that a scheme that avoids a root server for name resolution and includes caching capabilities at the client itself, reduces bandwidth requirements, and improves response times, resilience to DoS attacks at a modest increase in the complexity and memory requirements of the client.

TTL values are usually set very conservatively. Some experiments suggest that up to 85 percent of data that is timed out at the name servers is still valid. Performance can be further improved if the application begins the process of establishing communication with the IP address in the timed out record while the name server fetches the new record. If the IP addresses in the timed out and new records match, as is very likely, then the communication can continue. Only in the rare case of a changed address has fresh communication to be initiated.

References

1. Paul Mockapetris, â€œDNS Encoding of Network Names and Other typesâ€, RFC 1101, April 1989.

2. Paul Mockapetris and Kevin Dunlap, â€œDevelopment of the Domain Name Systemâ€ Proceedings of SIGCOMM â€™88, Computer communication Review Vol. 18, No. 4, August 1988, pp. 123-133.

3. Cachin C., Samar A., â€œSecure Distributed DNSâ€, IEEE International Conference on Dependable Systems and Networks, , July 2004, pp. 423-432.

4. Paul Albitz and Cricket Liu, â€œDNS and BIND, Fourth Edition.â€ Oâ€™Reilly and Associates, Incorporated, 2001.

5. Vinton G. Cerf, â€œOn the Evolution of Internet Technologiesâ€, Proceedings of the IEEE, Vol. 92, No. 9, September 2004, pp. 1360-1370.

6. Dan Pei, and Lixia Zhang,â€ A Framework for Resilient Internet Routing Protocolsâ€, IEEE Network, March/April 2004, pp. 5-12.

7. Chakrabarti A., Manimaran G., â€œ Internet Infrastructure Security: A Taxonomyâ€, IEEE Network, Volume 16, Issue 6, December 2002, pp. 13-21.

8. Gerco Ballintijn, Marten van Steen, Andrew S. Tanenbaum, â€œScalable Human-Friendly Resource Namesâ€, IEEE Internet Computing, Volume 5, Issue 5, October 2001, pp. 20-27.

9. Jaeyeon Jung, Sit E., Balakrishnan H., Morris R., â€œDNS Performance and the Effectiveness of Cachingâ€, IEEE/ACM Transactions on Networking, Volume 10, Issue 5, October 2002, pp. 589-603.

10. Richard Liston, Sridhar Srinivasan, Ellen Segura, â€œ Diversity in DNS performance measuresâ€, Proceedings of the second ACM SIGCOMM Workshop on Internet measurements, November 2002, Marseille France.

11. Garber Lee, â€œDenial of Service Attacks rip the Internetâ€, IEEE Computer, Volume 33, Issue 4, April 2000, pp. 12-17.

12. Michael F. Schwartz, Calton Pu, â€œApplying an Information Gathering Architecture to Netfind: a White Pages Tool for a Changing and Growing Internetâ€, IEEE/ACM Transactions on Networking, Volume 2, issue 5, October 1994, pp. 426-439.

13. Zegura, E.W.; Ammar, M.H.; Zongming Fei; Bhattacharjee, S., â€œApplication-layer anycasting: a server selection architecture and use in a replicated Web serviceâ€, IEEE/ACM Transactions on Networking, Volume 8, Issue 4, August 2000, pp. 455-466.