2011

Experimental study of DNS performance

Ananya Tripathi
Amity University, Uttar Pradesh, India

Farhat Khan
Amity University, Uttar Pradesh, India

Akhilesh Sisodia
Amity University, Uttar Pradesh, India

Originally published in the Proceedings of the 9th Australian Information Security Management Conference, Edith Cowan University, Perth Western Australia, 5th -7th December, 2011

EXPERIMENTAL STUDY OF DNS PERFORMANCE

Ananya Tripathi, Farhat Khan, Akhilesh Sisodia
Amity School of Engineering & Technology, Amity University,
Uttar Pradesh, India
{ananya.tripathi90, efarhatkhan, sisodiaakhilesh}@gmail.com

Abstract

An abbreviation for Domain Name System, DNS is a system employed for naming computers and network services. This system is organized into a hierarchical scheme of domains. Naming service provided by DNS is used in TCP/IP networks, such as the Internet, to easily locate computers and services like mail exchanger servers, through user-friendly names. When a user enters a DNS name in an application, DNS services resolves this name to other information associated with the name, such as an IP address. This paper presents the evaluation of a DNS server performance in the experimental backgrounds to establish the fact that frequent caching of results will improve the response time of the queries. It also simulates the client –server DNS model on OPNET. It thus proposes a performance-enhancing model for its better throughput keeping in mind, the various execution measures of DNS server like parallel requests, traffic distribution and least response time, which were tested on the DNS server.

Keywords
DNS, OPNET, DNS performance

INTRODUCTION

Domain Name System (DNS) (How Stuff Works,2011) maps domain names to its corresponding network locations, thus, providing information to most of the Internet applications and services for their operation. It is a globally distributed database.

Information related to domain names or host, like, reverse maps (mapping of IP address to host name) and mail routing information is also provided by the DNS server. Clients query name servers for values in the database. The organization of DNS name space is hierarchical. This eases the local administration of sub domains. The root of this hierarchy is centrally administered. At present there exist a collection of 13 such root servers (Root Servers in the World, 2011). Sub domains are delegated to other servers and this process may be repeated at all levels. These servers are authoritative for their portion of the name space (How Stuff Works,2011). Mappings in the DNS name space are called resource records. Two most commonly accessed resource records by a DNS server are address records (*A records*) and name server records (*NS records*) (DNS Books, 2011)). An *A record* specifies an IP address corresponding to a domain name (web address). An *NS record* specifies the name of a DNS server that is authoritative for a domain name, and thus is used to handle delegation paths (DNS Books, 2011).

To start off with, the DNS server at the ERNET India Lab (ERNET India, 2011) was tested for different conditions like parallel requests, repetitive requests etc. to get the real time results. Apart from this, the basic DNS client server model was simulated over OPNET (OPNET, 2011), a network simulation tool which helps to build an environment to test and support modeling of networking models. This helped in deriving a model for the better working of the DNS server. Also, the security aspect of the DNS records was tested (DNSSEC) (Lioy, Maino, Marian & Mazzocchi, 2000) by carrying out client server interactions on parallel virtual machines.

Thus, this paper has two objectives. First, it seeks to understand the performance and behavior of DNS from the client view point and, ultimately, it evaluates its performance for different conditions to analyze the potential throughput increase.

Summary of Results

DNS performance is measured in terms of performance rate received by the client and how the varying size of the database of resource records affects the response time. To facilitate this, DNS packets were captured at the client and server sides, thus calculating the response time. The course of this activity was stretched in a span of one month.

The response time on a real time system, i.e. the ERNET India Lab server, varied from 1ms-3ms for a file size of 350-500 entries. This figure became 0.233 ms – 0.268 ms when tested on OPNET for a file size varying from 50-300 entries. The parameters that were tested in this simulation tool included back-to-back queries, queries for good domain names and bad domain names (Forouzan 2006), delayed queries and queries with cached records. The server comprised of three types of resource record files, i.e. small file with approximately 50 entries, medium file with approximately 300 entries and large file with approximately 550 entries. The Packet Sniffer used in this experiment is *"Ethereal"* (Ethereal, 2011)*,* which was installed on the DNS server.

The rest of this paper presents the findings and substantiates these results. Section 2 presents an overview of DNS. Section 3 describes the traffic collection methodology and some salient features of the data collected. Section 4 analyzes the client-received performance of DNS, while Section 5 analyzes the effectiveness of caching. We conclude with a discussion of our findings in Section 6.

BACKGROUND

The DNS – Domain Name System comforts the user by translating the human understandable web address to the machine understandable IP address (Albitz & Liu, 2006). This address is translated to an actual 32 bit Internet address, on the system. The DNS protocol makes this translation possible, dynamic, fast, and available on many locations Agarwal et al. , 2003). In terms of structure, the DNS database resembles the UNIX file system (DNS Basics, 2011).

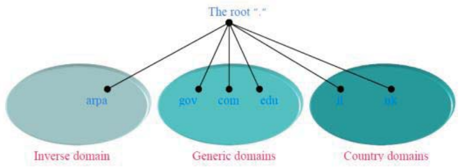

Fig 1 Domain Names Hierarchy

Starting form the root node, the database appears as an inverted tree. Each node has a text label (an identifier which is relative to its parent). The division of domain names starts at the root, which serves as the first level. The second level in the tree is consisted of three groups: generic domains, country domains and inverse domains (Forouzan 2006).

In DNS hierarchy administration of each domain can be divided among different organizations. Each organization can further divide its domain into a number of *subdomains* and handout responsibilities for those subdomains to others. Domains can thus contain both hosts and subdomains. This delegation of subdomains creates different zones. The part of a domain under administration of a single *authority*, that is a specific *name server*, is called *zone*. A name server that is authoritative for a particular zone has all the information needed for that zone.

Thus the DNS database is a distributed database of all the hosts throughout the network. Using the protocol's algorithms the needed information can be reached in this database (Azgomi & Khazan,2003).

Process of Resolution of query

The DNS server adopts two algorithms for resolution of a query namely **recursive** and **iterative**. These algorithms facilitate the client to get the most accurate and quick response corresponding to the request made by it. The default configuration of a DNS server is that it follows the recursive algorithm to resolve a query (Configure a DNS server,2011).

1) *Recursive Algorithm:* The server acts as resolver, querying another server and so on recursively, until it receives final answer, and sends it to client. This procedure is forwarding, as the server at each level

forwards the request to another server till the query is resolved (Danzig, Kumar, Miller, Neuman & Postel, 1998).

2) *Iterative Algorithm:* The server repeats the same query to multiple servers until receives answer, which is sent back to the client. The server returns the best possible answer it has (on the basis of cached records or zone data) or may return a referral (Balakrishnan Jung Morris & Sit, 2002). A referral is a pointer to a DNS server, which is authoritative for a lower level domain name space (BIND 9, 2005). Thus the client can now query the server for which it has obtained a referral (Andrews, 1998). This process continues till it has found the answer to the exact query, and is then returned to the client.

The entire process of translating a domain name for a client application is called *lookup*. A *query* by the client is sent to the DNS server in a DNS request packet (Azgomi, Khazan,2003). A *response* refers to a packet sent by a DNS server in reply to a query packet. This response from the server terminates the lookup procedure, by returning either the requested name-to-resource record mapping or an error indication.

OPNET Simulation Tool

OPNET provides supports the modelling of communication networks by providing a comprehensive development environment (Chipps , 2006-2008). It provides a platform to analyse behaviour and performance of the modelled system by performing event simulations. The OPNET environment incorporates tools for study of any networking model, by providing design of models, their simulation, data collection, and data analysis (Dolev & Wiener, 2002).

In the **"DNS client-server model"** used in our experiment, there was a need to customize an existing model provided by OPNET i.e. the Ethernet server & Ethernet workstation (Chipps, 2006-2008).

Using the OPNET capabilities new packets and nodes were added in this model. Also the codes of specific processes, which allow the implementation of the protocol, were added.

DATA COLLECTED

The data collection intended at recording the response time of the different queries made under various conditions. As mentioned earlier this collection was made in two scenarios i.e. one in a real time environment (on the ERNET India Lab server) and the second on the OPNET platform, which is a network simulation tool.

Real Time Testing

In the real time testing, the response time was recorded on the basis of duration of sending and receiving of the DNS packet. The traces were collected in June 2011, at the link that connects the ERNET India labs to the rest of the Internet. At the time of the study, there were 24 internal sub networks sharing the router, used by over 500 users and over 1200 hosts. The first trace was collected from 10:00 A.M. to 2:00 P.M. on June 3 2011, IST.

The response time varied from 1ms (for repetitive queries) to 3 ms (for new queries). Figure 2 shows the results.

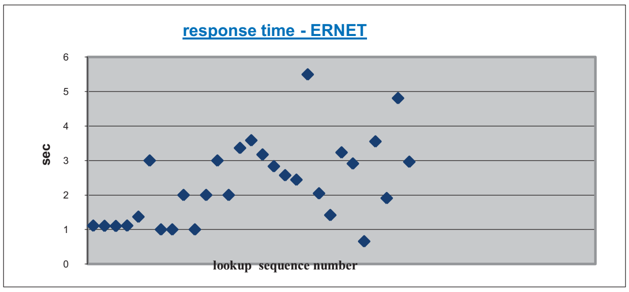

Fig 2 Response time for the queries collected at ERNET India Lab. The average time comes is 2.337 ms.

As observed during the data collection, the repeated queries were served quicker than the new ones. This clearly establishes the fact that the entries that were already cached in the DNS server were the cause of this quicker response.

The time of 30 incoming queries on the DNS server at ERNET India Lab, and its corresponding response was recorded and thus represented in the above graph.

Testing on OPNET

The DNS is implemented at the application layer, however according to the unique model of OPNET node model, additional modifications needed at lower levels, especially the tpal layer (Dolev & Wiener, 2002). On the OPNET Ethernet node model, which was the base model in our experiment, the tpal layer is an interface between the application and the transport layer, and is used in order to prevent the user from dealing with different type of transport protocols (UDP, TCP) (Farrera et al.). For the experiment a client-server topology (one client, one server) was created and simulated. The DNS scenario starts with the initialisation of client and server variables, registering the process and waiting for queries to arrive. The processes occurring at the client and server locations are:

Client: Firstly the client registers itself with the server followed by which, a DNS query is generated in the application node and passed to the DNS node. The DNS node starts a new process (for the lookup mechanism) that encapsulates the query in a DNS packet and sends it to the designated DNS server. This process then enters a wait state. The response is encapsulated in the same DNS packet by the server and sent back to the client. This packet is extracted at the client node. The packet is then passed to the DNS node. Here, the response is matched to a waiting process and then the process ends (Dolev & Wiener, 2002).

Server: At the server end, the moment a query is received, a new process is registered and is correspondingly resolved to give the IP address to the client. If it has answer, result is sent back to the sender and process ends. If no answer is available, the query is forwarded to another server, and the process enters into wait state. When response reaches to the requesting server, the relevant waiting process is resumed and the response is sent back to the sender of the query (Farrera et al.).

The implemented client server model was tested on parameters like server response time according to server database size (with and without delay), two back-to-back queries, bad domain names and repetitive queries. They were tested on three different file size, namely small (50 entries), medium (100-350 entries) and large (350-500 entries). The performance is expressed in terms of response time with the help of graphical representations in all the above-mentioned conditions, and is described in the next section.

DNS PERFORMANCE: CLIENT PERSPECTIVE

This section analyses several aspects of DNS performance from the client's perspective. The graphs are represented in terms of lookup sequence number versus response time.

On the Basis of Server Database File Size

Fig. 3, 4 and 5 show the results of the response time for small , medium and large file sizes respectively. The average time comes out as 0.233 msec, 0.268 msec and 0.277 msec respectively Fig. 6, 7 and 8 show the results of the response time for a small, medium and large file size respectively with 0.1 sec delay. The average time comes out as 0.300 msec, 0.314 msec and 0.308 ms respectively.

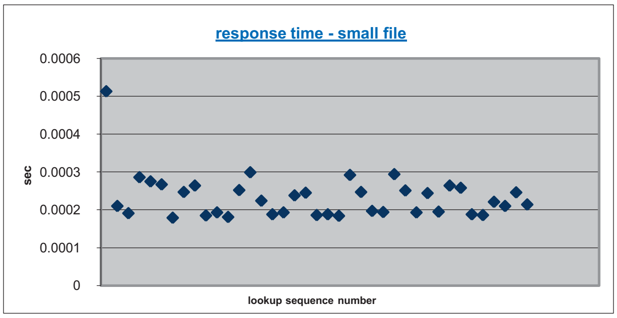

Fig 3 Response time for a small file size. The average time is 0.233 ms.

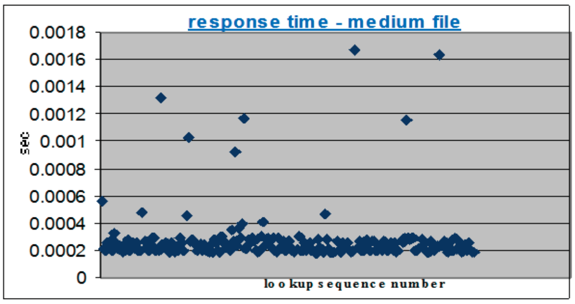

Fig 4 Response time for a medium file size. The average time is 0.268 ms

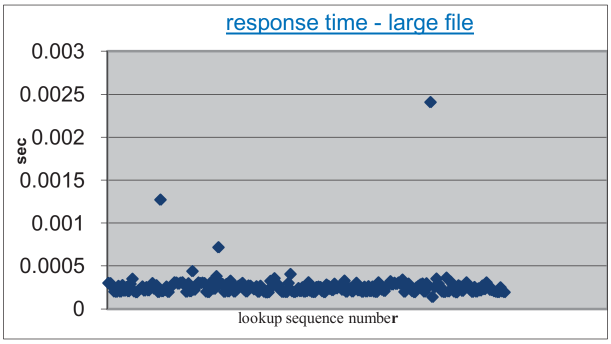

Fig 5 Response time for a large file size. The average time is 0.277 ms

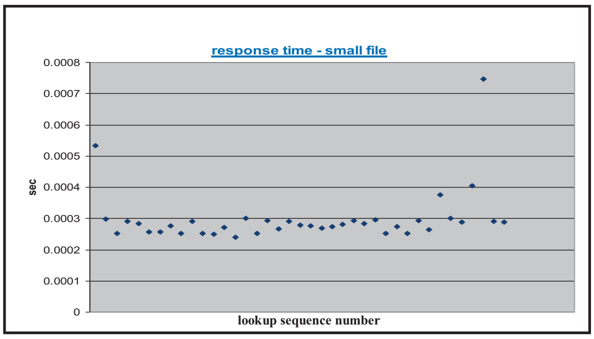

Fig 6 Response time for a small file size with a delay of 0.1 sec. The average time is 0.300 ms

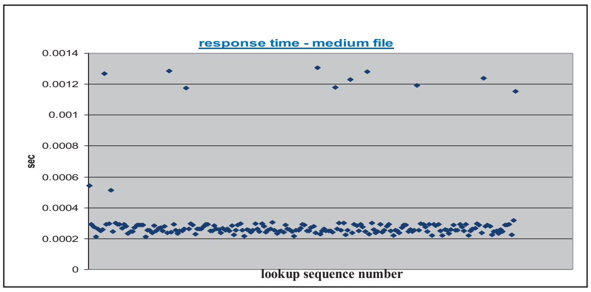

Fig 7 Response time for a medium file size with a delay of 0.1 sec. The average time is 0.314 ms

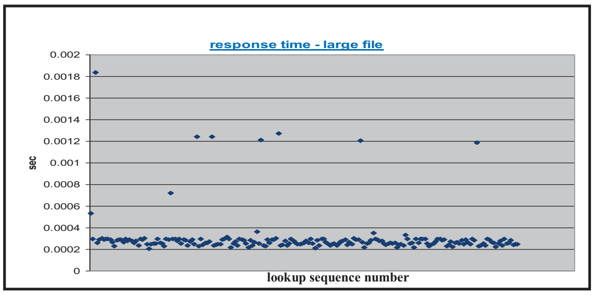

Fig 8 Response time for a large file size with a delay of 0.1 sec. The average time is 0.308 ms

Thus it can be inferred that the size of the file is almost insignificant, and the difference of response time between the small and medium (50-350 entries) is larger (0.035 msec) than the difference between medium and large (350-500 entries) which is 0.011 msec. Also, the average response time for delay query is slower than queries with no delay. The reason is that the non-delayed queries were fast enough to utilize the server before it switched to other processes. On the other hand, the delay between queries was long enough for the CPU to switch to other processes, so for every delayed query the time of switching back was added to the response time (Xinjie, 1999).

Back to back Queries

Fig. 9 and 10 shows the results of the response time for a small file size for the first and second requests respectively. The difference in time comes out as 0.024 ms in this case. Fig 11 and 12 shows the results of the response time for a medium file size respectively. The difference in time comes out as 0.006 ms in this case. For a large file size this difference in time comes out as 0.012 ms.

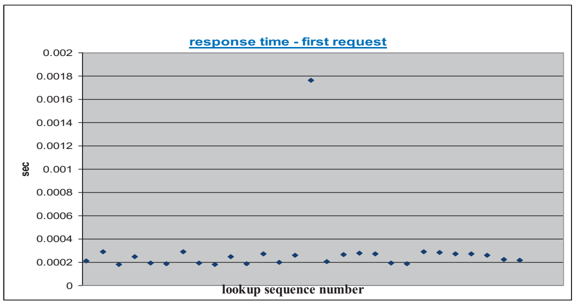

Fig 9 Response time for a small file size- first request

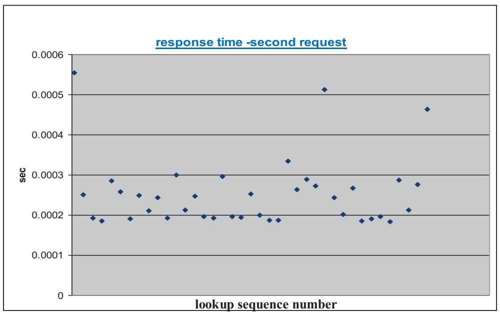

Fig 10 Response time for a small file size- second request

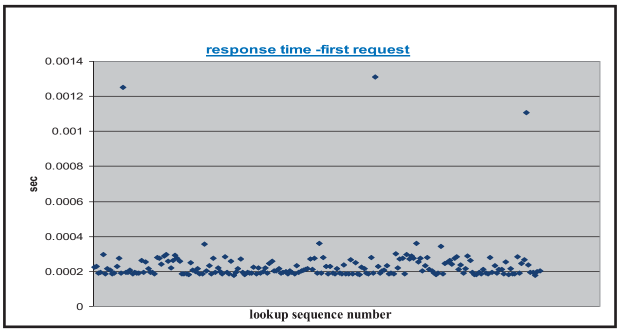

Fig 11 Response time for a medium file size- first request

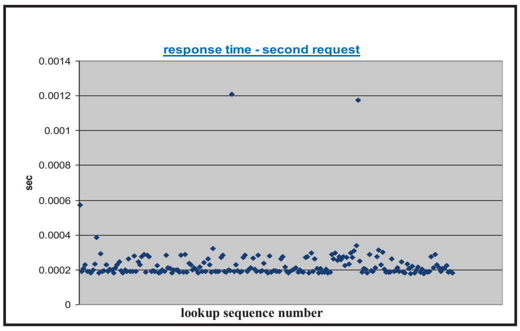

Fig 12 Response time for medium file size- second request

Thus it can be inferred that in all sizes the second query receives quicker answer (about 0-40 ns faster) than the first query. We have found an interesting fact during our research that when the file size is small with delay, the response time is 400 ns faster than the first one. The reason for faster resolution is the cache managed by the DNS server for every request it gets. Thus the cache resolves any repeated request without looking the whole database to give faster response.

Bad domain names

Fig. 13 shows the results of the response time for a request with bad names, meaning addresses that do not exist in the database. The average time comes out as 0.332 ms in this case.

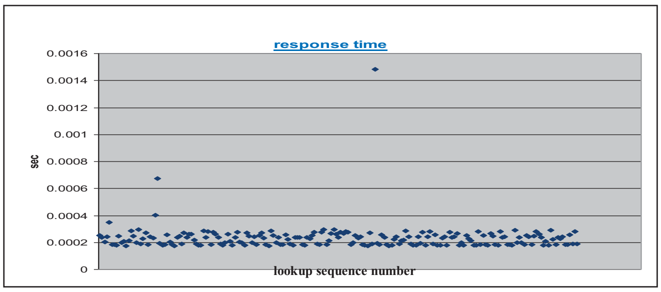

Fig 13 Response time for a bad name request

The reason for the longer average response time is probably due to the fact that the DNS server forwards the request to another server after searching its own at database (Beakcheol et al., 2009). As mentioned earlier, this process is called *forwarding*.

Repetitive Queries

This test was carried out with a delay of 0.1 ms between the two queries for the same name. Fig. 14 and 15 show the results of the response time for a small file size for the first and second requests. The difference in time comes out as 0.019 ms in this case. Fig 16 and 17 shows the results of the response time for a medium file size. The difference in time comes out as 0.026 ms in this case. For a large file size this difference in time comes out as 0.011 ms.

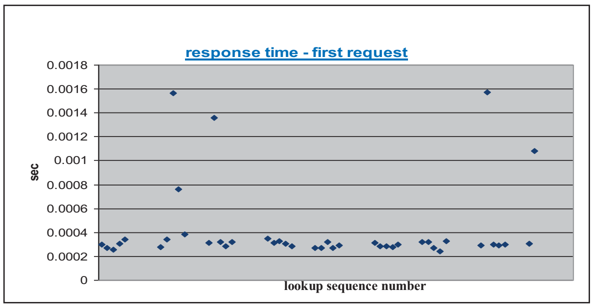

Fig 14 Response time for a small file size – first request

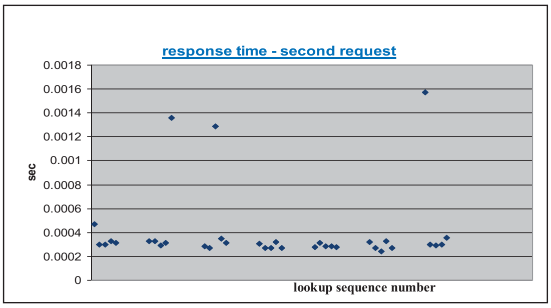

Fig 15 Response time for a small file size – second request

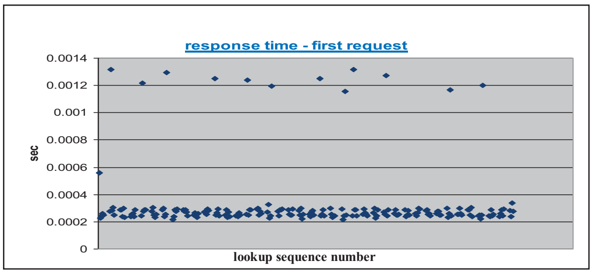

Fig 16 Response time for a medium file size – first request.

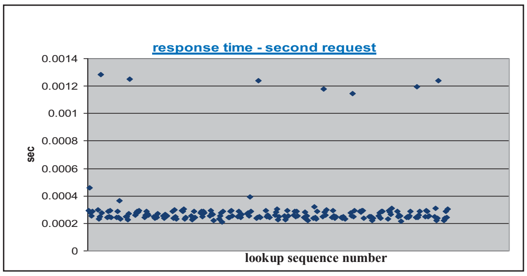

Fig 17 Response time for a medium file size – second request.

We can see that the significant difference between the first and second query is in order of 10-20 ns for the same address due to the DNS cache.

EFFECTIVENESS OF CACHING

The results in previous sections analyzed the collected traces to characterize the actual performance of DNS server. Significant drop in response time was recorded in case of repetitive queries thus establishing the fact that the cached queries are served faster (Deb & Srinivasan , 2008). Also, the difference in time for small and medium file size establishes the fact that caching the records will be useful. Caching can be done by increasing the value of TTL field (Time to live) of the resource records. This cached information if shared among many servers, will help them improve their response time.

CONCLUSION

This paper establishes the fact that frequent caching of results will significantly improve the response time of the queries and will make user experience, a little fast. This paper has presented a detailed analysis of traces of DNS and associated traffic collected on the Internet links of the ERNET India Laboratory. Also, the DNS server was analyzed in an experimental background on OPNET, thus testing the performance of DNS from client's perspective. This helped to establish an improved model for the DNS, which employs frequent caching.

REFERENCES

Agarwal A., Agarwal T., Chopra S., Feldmann A., Kammenhuber N., Krysta P., Vocking B. (2003) “An Experimental Study of k-Splittable Scheduling for DNS-Based Traffic Allocation. Technische University Munchen.

Albitz Paul, Liu Cricket (2006), DNS & BIND 3rd edition. O'Reilly & Associates (Sebastopol), 1998

Amit Dolev & Amir Wiener (2002) "DNS Client – Server Model on OPNET”, Computer Networks Lab, Technion - Israel Institute of Technology

Andrews M. (Mar. 1998), “Negative caching of DNS queries (DNS NCACHE)”, RFC 2308.

Application and network performance with OPNET.(2011). Available: http://www.opnet.com/

Azgomi Abdollahi Mohammad, Khazan Golriz. (2003). “DNS Spoofing Attack Simulation for Model-Based Security Evaluation”,International Journal of Advanced Science and Technology

Balakrishnan Hari, Member, IEEE, Jung Jaeyeon, Morris Robert and Sit Emil (October 2002), “DNS Performance and the Effectiveness of Caching”, EEE/ACM TRANSACTIONS ON NETWORKING, VOL. 10, NO. 5

Beakcheol Jang, Dongman Lee, Hyunchul and Kim Kilnam Chon (AUGUST 2009), “DNS Resolution with Renewal Using Piggyback”, JOURNAL OF COMMUNICATIONS AND NETWORKS, VOL. 11, NO. 4,

BIND 9 Administrator Reference Manual (2005), Internet Systems Consortium, Inc.

Chipps Kenneth M. “How to Use OPNET IT Guru Academic Version”, 2006-2008

Danzig P., Kumar A., Miller S., Neuman C. and Postel J (Oct. 1993). “Common DNS implementation errors and suggested fixes,”, RFC 1536.

Deb S.K., Pavan A., Srinivasan S. (2008), “An improved DNS server selection algorithm for faster lookups”, Communication Systems Software and Middleware and Workshops, Bell Labs Res., Murray Hill, NJ.

DNS Basics Books .(2011). Available: http://www.zytrax.com/books/dns/info/minimum-ttl.html

ERNET India (2011). Available: http://www.eis.ernet.in/index.htmHow Stuff Works.(2011). Available: http://www.howstuffworks.com/

Ethereal: A Network Protocol Analyzer.(2011). Available : http://www.ethereal.com/Location of Root Servers in the World (2011). Available: http://www.root-servers.org/

Farrera Paredes, Fleury Martin, Jammeh Emmanuel, Lucio Flores Gilberto, Reed J. Martin. “OPNET Modeler and Ns-2: Comparing the Accuracy Of Network Simulators for Packet-Level Analysis using a Network Testbed”. Electronic Systems Engineering Department, University of Essex, Colchester, Essex C04 3SQ, United Kingdom.

Forouzan, B.A. (2006) Chapter 25 - Domain Name System ,Chapter 26-Remote Logging, Electronic Mail, and File Transfer in Data Communications and Networking. (Boston):McGraw-Hill Publications.

How to configure a DNS server. (2011). Available: http://support.microsoft.com/kb/323380

Lioy Antonio, Maino Fabio, Marian Marius, Mazzocchi Daniele (2000). "DNS Security" Terena Networking Conference, Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino (Italy), 1-2.

Pro DNS and BIND. (2011). Available: http://www.zytrax.com/books/dns/info/soarecords.html

Xinjie Chang (1999), “NETWORK SIMULATIONS WITH OPNET”, Winter Simulation Conference Phoenix, AZ, USA