Chapter 3

DNS PERFORMANCE ANALYSIS
3.1 Introduction

The domain system is intentionally extensible. Researchers continuously propose, implement and experiment with new data types, query types, classes, functions, etc. The applications on the Internet were getting more sophisticated and created need for general purpose name service. Sun Micro Systems developed the Domain Name System (DNS) in the early 1980's as an easier way to keep track of addresses. It has been the addressing system for the Internet ever since. The DNS establishes a hierarchy of domains, which are groups of computers on the Internet. The DNS gives each computer on the NET an Internet address or domain name, using easily recognizable letters and words instead of numbers. The domains at the top level of the hierarchy maintain lists and addresses of the domain just beneath them. These subordinate domains have similar responsibilities for the domains just beneath them, and so on. In this way, every computer on the Net gets an Internet address. The DNS also helps Internet computers send e-mail to the proper destination by converting textual Internet address into its IP numeric equivalent [28]. Zona Research reports that more than 2,700 B2B and B2C e-business and Web hosting firms are adding performance measurements as line item in their IT budgets [29, 30]. There are 171,638,297 Hosts advertised in the DNS (Jan 2003) [31].

Highly scalable, global directories make today's telecommunication networks function. To communicate with a Web site or send an e-mail, or use a cellular phone, before the application gets off the ground, there is some sort of directory lookup that is performed to gather the information that the application need to talk.
Domain Name System is responsible for translating domain name into IP address. The analysis of DNS performance is a must since almost 20% of Web site download time depends on the DNS conversion time. This study is carried out after studying different ISP DNS conversion time, also analyzing performance of more than thousand Web sites, and recorded normal DNS lookup time and reverse lookup time.

The Internet’s Domain Name System illustrates how to implement a large-scale distributed database system using the client/server program paradigm. A distributed database physically stores data on two or more computer systems. To program the distributed data, the geographical location of the computers is irrelevant [32, 33].

3.2 Background

Using directory services, a computer program searches for attributes of a different sort, such as access control, domain/group membership, authentication, network addresses, and port addresses. To manage the vast volume of data that it has to deal with, a directory service is typically implemented using a database stored on a server [34].

Figure 3.1 shows a typical directory service implementation. Clients connect to the directory service to query and update the directory service database. Some directory services can communicate information to and from other directory services. Directory service often provides a mapping between a human-friendly name and an address that a human would have trouble remembering. This is called a name service, and is one of many services that a directory service provides. Domain Name System is an example of an extremely popular name service.
A directory or name service can typically be represented as a tree, and it is true for DNS as well. Each entry in a DNS tree is called a node. Each node has a name. The node at the top of the tree is called the root. Each node can also be described by a domain name, also known as fully qualified domain name. A domain name is constructed by using the name of the node, and then appending the name of the parent of that node and all remaining parent nodes in the respective order to the root of the tree. DNS treats the domain name as case insensitive. There are sets of what are called "top-level domain names" (TLDs). These are the generic TLDs (EDU, COM, NET, ORG, GOV, MIL, INT), and the two letter country codes from ISO-3166 [35].

The most widely used nameserver implementation in the DNS is the Berkeley Internet Name Domain (BIND) [36, 37]. Other commercially available DNS management system is QuickDNS. QuickDNS is a DNS management solution that allows many users to simultaneously manage DNS servers on different platforms through an easy-to-use interface and provides control access for
users and groups to individual zones or servers, by configuring user’s rules [38].

3.3 Elements of the DNS

The DNS has three major components:

The DOMAIN NAME SPACE and RESOURCE RECORDS are specifications for a tree structured name space and data associated with the names. Conceptually, each node and leaf of the domain name space tree names a set of information, and query operations are attempts to extract specific types of information from a particular set. A query names the domain name of interest and describes the type of resource information that is desired. For example, the Internet uses some of its domain names to identify hosts; queries for address resources return Internet host addresses.

NAME SERVERS are server programs, which hold information about the domain tree's structure and set information. A name server may cache structure or set information about any part of the domain tree, but in general a particular name server has complete information about a subset of the domain space, and pointers to other name servers that can be used to lead to information from any part of the domain tree. Name servers know the parts of the domain tree for which they have complete information; a name server is said to be an AUTHORITY for these parts of the name space. Authoritative information is organized into units called Zones, and these zones can be automatically distributed to the name servers, which provide redundant service for the data in a zone.

Resolvers are programs that interface user programs to domain name servers. In the simplest case, a resolver receives a request from a user program
(e.g., mail programs, TELNET, FTP) in the form of subroutine call, system call etc., and returns the desired information in a form compatible with the local host's data formats. The resolver is located on the same machine as the program that requests the resolver's services, but it may need to consult name servers on other hosts. Because a resolver may need to consult several name servers, or may have the requested information in a local cache, the amount of time that a resolver will take to complete can vary from milliseconds to several seconds. A very important goal of the resolver is to eliminate network delay and name server load from most requests by answering them from its cache of prior results. Caches, which are shared by multiple processes, users, machines, etc., are more efficient than non-shared caches [39].

Queries are messages, which may be sent to a name server to provoke a response. In the Internet, queries are carried in UDP datagrams or over TCP connections. The response by the name server either answers the question posed in the query, refers the requester to another set of name servers, or signals some error condition. Clients query DNS servers by means of resolvers. A resolver is typically a piece of code running on DNS client that carries out the DNS queries. The resolver sends queries to the server and cache replies. Each query is sent using a User Datagram Protocol. Clients can issue two types of queries: recursive and iterative. A recursive query from a client causes a DNS server to contact another DNS server when the first server is unable to satisfy a client request. This recursive can be cascaded so that the second server can contact a third server and so on until the request is satisfied. Clients most typically issue an iterative query, wherein a server that does not have the required information returns pointers to other servers that might be able to provide the information. Iterative queries are less stressful for servers. A resolver typically caches the values returned by a DNS server in an effort to improve performance and lower the load on the DNS server.
3.3.1 Resource Records

DNS defines among other things a distributed database. The database resides on servers that administer various domains and subdomains. The database is organized into resource records. A resource record contains an IP address, the name of a host willing to accept mail for this domain and other information [40]. Figure 3.2 shows DNS resource record.

<table>
<thead>
<tr>
<th>NAME</th>
<th>Type</th>
<th>Class</th>
<th>TTL</th>
<th>RDLength</th>
<th>Rdata</th>
</tr>
</thead>
</table>

- The Name field is a string that identifies the domain that owns this resource record.
- The Type field is a 16-bit value that identifies the type of resource record.
- The Class field is 16 bits long and identifies the protocol family of the resource record type.
- The Time To Live (TTL) field contains a signed 32-bit value that indicates the time in seconds that the resource record retains in a cache.
- The RDLength field specifies the length of the Rdata field in bytes.
- The RData field is of variable size, depending on the type of the resource record.
In general, the user does not generate queries directly, but instead makes a request to a resolver, which in turn sends one or more queries to name servers and deals with the error conditions and referrals that may result. Of course, the possible questions which can be asked in a query does shape the kind of service a resolver can provide.

The DNS name space is divided into zones. A zone is created by logically cutting the link between a node and its parent. The resulting isolated subtree is the zone. A server has full authority over its zone. When a server in a zone is presented with a query, the server returns either the requested information or returns a referral to a server that has authority over the subdomain and refers that query to a hypothetical DNS name space divided into zones. Figure 3.3 shows Domain Name Resolution configuration [41].

![Figure 3.3 Domain Name Resolution Configuration](image-url)
3.4 UDP usage

Messages are sent using UDP user server port 53 (decimal). Messages carried by UDP are restricted to 512 bytes (not counting the IP or UDP headers). Longer messages are truncated and the TC bit is set in the header [42]. UDP is not acceptable for zone transfers, but is the recommended method for standard queries in the Internet. Queries sent using UDP may be lost, and hence a retransmission strategy is required. Queries or their responses may be reordered by the network, or by processing in name servers, so resolvers should not depend on them being returned in order.

The optimal UDP retransmission policy will vary with the performance of the Internet and the needs of the client, but the following are recommended:

- The client should try other servers and server addresses before repeating a query to a specific address of a server.
- The retransmission interval should be based on prior statistics if possible. Depending on how well connected the client is to its expected servers, the minimum retransmission interval should be 2-5 seconds.

3.5 TCP usage

The message is prefixed with a two byte length field which gives the message length, excluding the two byte length field. This length field allows the low-level processing to assemble a complete message before beginning to parse it. Several connection management policies recommended are as follows:

- The server should not block other activities waiting for TCP data.
- The server should support multiple connections.
The server should assume that the client will initiate connection closing, and should delay closing its end of the connection until all outstanding client requests have been satisfied.

If the server needs to close a dormant connection to reclaim resources, it should wait until the connection has been idle for a period on the order of two minutes. Since the server would be unable to answer queries anyway, a unilateral close or reset may be used instead of a graceful close.

3.6 Cache usage

Caching is one of the most powerful features of DNS. It allows the local server to dynamically build up a supply of answers to its most frequently asked questions. The local server has all the information it needs. This creates a very efficient service.

The scalability and performance of the DNS largely depends on the caching of resource records across intermediate name-servers. Caching is controlled by the Time To Live (TTL) value, which in turn depends on the frequency with which administrators expect the data to change. For example, Internet RFC 1912 recommends minimum Time To Live (TTL) values around 1 to 5 days [43]. Earlier documentation had recommended 1 day as the minimum TTL for most servers and around 4 days for top-level domains. These values are now considered too small. Once a domain stabilizes, values on the order of three or more days are recommended. A recent study shows, however, that a majority of name servers use a default Time To Live (TTL) value of 86400 seconds (or 1 day) for their domain [44, 45]. In general, a resolver has to cache all data which it receives in responses, since it may be useful in answering future client requests. A mapping is simply an association between two things, in this case a
machine name, like ftp.linux.org, and the machine's IP number (or address) 199.249.150.4. DNS also contains mappings the other way, from the IP number to the machine name; this is called a "reverse mapping". Converting a number to a name is the reverse of what is done in a normal DNS query when a name is converted to a number, which makes mapping number to a name a reverse lookup.

3.7 DNS Performance Analysis

There are several areas of research and standardization efforts relating to DNS. Anees Shaikh [37] draws attention to two of the main issues in using DNS: 1) the negative effects of reducing or eliminating the cache lifetimes of DNS information, and 2) the implicit assumption that client name servers are indicative of actual client location and performance. Anees Shaikh et al., quantify the impact of reducing DNS Time To Live (TTL) values on Web access latency and show that it can increase name resolution latency by two orders of magnitude. As the embedded objects served from multiple sources increases, name lookup overheads can grow to nearly 50%.

The general problem of determining distance between Internet hosts or networks has received a great deal of recent attention. [46]. For example, the IDMaps architecture attempts to provide a service in which traceroute measurements are distributed over the Internet using IP multicast [47, 48]. The SONAR service provides an interface between applications and proximity estimation services [49]. Related to proximity measurement is the question of which metrics provide the best indication of actual latency. Recent work has considered network hops, AS hops, and RTT metrics, along with various means of collecting them, including active probing or passive participation in BGP peering sessions [50, 51]. Several modifications to DNS have been
proposed, both to provide additional location information about hosts, and specifically to facilitate server selection. The LOC resource record allows geographic location information to be expressed in the DNS as latitude and longitude [52]. Similarly, the GL resource record encodes location information in terms of hierarchical locator (country code, postal code) and a textual address [53]. The SRV-DNS resource record is a proposed standard which specifies the identity of servers that provide a specific service (e.g., LDAP) using a specified protocol (e.g., TCP), in a specified domain (e.g., service.com) [54]. Earlier work suggests using the existing DNS zone transfer mechanism as a way to add flexible load-balancing capability to a name server [55]. Finally, some recent work has proposed new mechanisms to reduce client latency related to name resolution using prefetching or proactive cache management techniques [56]. This work further affirms that DNS caching plays a crucial role in determining client-perceived latency.

This section quantifies the DNS conversion time by collecting more than 1000 DNS domain names and IP addresses using two Internet Service Providers namely A-Team online and Dishnet. Analysis of DNS lookup directory performance and reverse DNS lookup performance of two ISP’s has been attempted. Assessment of performance improvement by using local DNS cache and using DNS server in the ISP setup was also analyzed.

The following sections present the results from several tests performed using a DNS-JAVA tool developed by the researcher. Several experiments were performed to verify the above hypotheses and the results from these experiments have been presented.

To get initial feeling for the relevance of the approach the researcher has analyzed more than 1000 domain IP addresses using JAVA tool. Figure 3.4
shows DNS mapping performance analysis of two ISPs namely A-Team Online and Dishnet. Noteworthy here is the dominance of the conversion time between 500msec and 1200 msec. This is because client nameserver queries root nameserver and receives the IP address (Figure 3.8). If the DNS latency exceeds 2000 msec, the queries made by the client namserver are transferred to the authoritative nameserver (Figure 3.9). This process increases DNS conversion latency.

Figure 3.5 reveals reverse DNS mapping performance of A-team Online versus Dishnet ISPs. The dominance of reverse DNS mapping latency is between 1000 msec to 3500 msec. In reverse DNS mapping the latency is more than 2 times as compared to normal DNS mapping. This increase in latency time is due to the fact that the structure of IP address is the opposite of the structure of domain name and hence the address must be reversed, when it is treated as a reverse name.

Figure 3.6 shows DNS Caching performance analysis of A-Team Online versus Dishnet ISPs. A-Team Online uses 64 kb cache memory in the DNS server, while Dishnet uses 512 kb cache memory in the DNS server. The data for DNS conversion latency, reverse DNS latency and DNS caching latency are given in Appendix I.
Figure 3.4 A-Team Versus Dishnet DNS performance
Figure 3.5 A-Team Versus Dishnet reverse DNS Mapping
Figure 3.6 A-team Versus Dishnet DNS Caching Performance
3.8 Statistical Analysis

Table 3.1 shows the mean (M), standard deviation (S.D), Standard error (S.E), test statistics (for large samples) Z, probability (P), for two ISPs. Let $\xi_1$ be the mean of a random sample of size $n_1$ from a population (A-Team Online) with mean $\mu_1$ and variance $\sigma_1^2$ and let $\xi_2$ be the mean of an independent random sample of size $n_2$ from another population (Dishnet) with mean $\mu_2$ and variance $\sigma_2^2$.

**Under the null hypothesis:** $H_0 : \mu_1 = \mu_2$, i.e., there is no significant difference between the average conversion time of two ISP’s.

**Under alternative hypothesis:** DNS and Reverse DNS: $H_0 : \mu_1 < \mu_2$, i.e., average conversion time of A-Team online is less.

DNS Caching: $H_1: \mu_1 > \mu_2$, i.e., average conversion time of Dishnet is less. The test statistic is

$$Z = \frac{(\xi_1 - \xi_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \sim N(0,1)$$

When $\sigma_1^2$ and $\sigma_2^2$ are not known, then the estimates from sample values, for large samples, $(n1-1) = n1$, and $(n2-1) = n2$, and $\sigma_1^2 = S_1^2$ and $\sigma_2^2 = S_2^2$.

$$Z = \frac{(\xi_1 - \xi_2)}{\sqrt{\frac{S_1^2}{n1} + \frac{S_2^2}{n2}}} \sim N(0,1)$$

As per Table 3.1 the P value is 0.0014, for DNS performance of two ISPs, which is not significant, implies $H_0 : \mu_1 = \mu_2$, is true. The average performance does not differ significantly. The reverse DNS performance of two ISPs the P value is 0.0418 which is significant at 5% level, $H_1 : \mu_1 < \mu_2$. 


is true. The average DNS performance of A-Team Online is less. The DNS Cache performance for the two ISPs the P value is 0 which is significant at 1%. The average DNS caching time is less. Table 3.2 shows the DNS minimum and maximum time conversion performance of two ISPs.

### Table 3.1
Two ISPs responses

<table>
<thead>
<tr>
<th>ISP</th>
<th>MEAN (msec)</th>
<th>S.D</th>
<th>S.E</th>
<th>Z</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td>A-Team Online</td>
<td>664</td>
<td>737.8</td>
<td>54.10</td>
<td>3.25</td>
</tr>
<tr>
<td></td>
<td>Dishnet</td>
<td>1035</td>
<td>1364</td>
<td>100</td>
<td>3.25</td>
</tr>
<tr>
<td>Reverse DNS</td>
<td>A-Team Online</td>
<td>2140</td>
<td>1367</td>
<td>91.77</td>
<td>4.24</td>
</tr>
<tr>
<td></td>
<td>Dishnet</td>
<td>2725</td>
<td>1537</td>
<td>103.1</td>
<td>4.24</td>
</tr>
<tr>
<td>DNS Caching</td>
<td>A-Team Online</td>
<td>448.7</td>
<td>488.7</td>
<td>15.95</td>
<td>23.96</td>
</tr>
<tr>
<td></td>
<td>Dishnet</td>
<td>20.92</td>
<td>161.3</td>
<td>11.27</td>
<td>23.96</td>
</tr>
</tbody>
</table>

### Table 3.2
Various DNS responses for two ISPs

<table>
<thead>
<tr>
<th>ISP</th>
<th>DNS (msec)</th>
<th>Reverse DNS (msec)</th>
<th>DNS Cache (msec)</th>
<th>Local DNS Caching (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-Team Online</td>
<td>min</td>
<td>10</td>
<td>220</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>2750</td>
<td>8620</td>
<td>1102</td>
</tr>
<tr>
<td>Dishnet</td>
<td>min</td>
<td>10</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>8422</td>
<td>12808</td>
<td>10</td>
</tr>
</tbody>
</table>
3.9 Basic DNS Operation

DNS is a rooted, hierarchical structure. The bulk of DNS database information is distributed among the tens of thousands of servers that are responsible for lower level domains. Building such a server and creating the database information for that server is one of the responsibilities of a domain administrator.

As shown in Figure 3.7, every authoritative server is responsible for a piece of the domain name space. A server is said to be authoritative when it can answer a query with complete accuracy. It can answer with authority because the server has the complete database for that part of the domain name space. The part of the domain name space of an authoritative server that is responsible for it is referred to as that server’s zone of authority. The root domains and the top-level domains are basically composed of pointers to the servers for lower level domains. Below the top-level domains are the second-level domains that are assigned to organizations around the Internet. There are 13 servers that provide root service.

Figure 3.7 illustrates how a client typically finds the address of a service using DNS. The client application uses a resolver, usually implemented as a set of operating system library routines, to make a recursive query to its local nameserver. The local nameserver may be configured statically (e.g., in a system file), or dynamically using protocols like DHCP or PPP. After making the request, the client waits as the local nameserver iteratively tries to resolve the name (www.service.com in this example). The local nameserver first sends an iterative query to the root to resolve the name (steps 1 and 2), but since the subdomain service.com has been delegated, the root server responds with the address of the authoritative nameserver for the sub-domain, i.e., ns.service.com
(step 3). The client’s nameserver then queries ns.service.com and receives the IP address of www.service.com (steps 4 and 5). Finally the nameserver returns the address to the client (step 6) and the client is able to connect to the server (step 7) [2].

Figure 3.7 Basic DNS Operation
3.9.1 Modified DNS Technique

The designers of DNS wanted to avoid the storage of an up-to-date name resolution file on each computer. Instead the DNS places name resolution data on one or more special servers. DNS simplifies maintenance and controls the size of individual database files by delegating complete authority over individual domain. The advantage of DNS distributed database is that it allows the network to grow more naturally.

The modified DNS operation is shown in Figure 3.8. The client application uses a resolver, to make a recursive query to its local nameserver. After making the request, the client waits as the local nameserver iteratively tries to resolve the name (www.service.com) in this example). The first thing the local nameserver did was see if it could directly answer the query. It did this by checking its cache for answers (step 1). If it has no information about the remote domain, it sends an iterative query to the root server. The local nameserver always knows how to find a root server. If the domain name is resolved at the root server (step 2) then the root server sends the client address to the remote Web server (step 3). Then the required Web page is transferred from Web server to the client machine. In basic DNS operation 7 hops are required for IP address translation (Figure 3.7). But by using modified DNS operation only 3 hops will be required.
As shown in figure 3.9, if the domain name is not available at the root server, since the sub domain has been delegated, the root server responds with the address of authoritative nameserver and received the IP address (step 4). Instead of transferring Web server IP address to the client machine, the root server transfers the IP address of client machine to the origin Web server (step 5). Then the Web page is transferred from origin Web server to the client machine (Figure 3.9). Thus 2 hops distance will be reduced as compared to the normal DNS operation. The average topological distance of client nameserver will be 7.6 hops with a median of 8 [2].

**Figure 3.8 Modified DNS operation – root nameserver**
The average client to nameserver round-trip latency was 234 msec, though this was dominated by the average first-hop latency which was 188 msec. These results show that even when considering direct distances, clients and nameservers are often topologically quite far apart [31]. By using modified DNS technique, 2 to 8 hops will be reduced, thereby reducing greatly the DNS conversion delay.
3.10 Discussion

Initial measurements of caching, using the local DNS caching, indicate, that using local DNS caching is effective in reducing DNS conversion delay. However, the measurement of local DNS caching performance is only an indication of the effectiveness of local DNS caching. However, researcher’s expectation based on the past success of other commercial systems that have employed local DNS caching, is that local DNS caching will be very effective.

Two different DNS Servers were studied: the DNS Server for A-Team Online and a DISHNET POP (POINT OF PRESENCE). These DNS Servers reflect the progression from a District level (Class A) to a top level (National level—Class C) ISPs. By using the ISP level DNS Server instead of POP, DNS performance and reverse DNS performance will be increased considerably. By using powerful DNS Server DNS caching performance improved considerably.

By using modified DNS, IP address resolution technique 2 to 8 hops delay will be reduced. DNS address translation occurs thousands of times a day on the Internet. So, DNS latency will be reduced by adopting new DNS address translation technique. This tidy scenario is complicated by some additional features of the modern network, including address caching, DHCP, and dynamic DNS. However the functionality of most TCP/IP networks depends on this form of DNS name resolution.

The next chapter discusses Distributed Caching Architecture analysis, using different protocols, like TCP, UDP and Multicasting and proposes which protocol is suitable for Distributed Caching Architecture, to solve flash-crowds problem.