Domain Name System (DNS):

The DNS coordinates the translation of hostnames to IP addresses and IP addresses into hostnames. The translation of IP addresses into hostnames is required if the remote machine performs logging or authentication. For example, the administrators of *ftp.cas.mcmaster.ca* do not want users from *www.badguys.com* to ftp into their machine. Then the FTP application at *ftp.cas.mcmaster.ca* would need to determine the hostname of the sending machine before granting access.

Routers forward packets based on the 32 bit destination address in the IP header. Instead, machines on the internet have host names, such as *opt-lab.cas.mcmaster.ca*, that consist of strings separated by periods. The reasons are the following:

1. Remembering a hostname is easier than remembering an IP address.
2. The IP address associated with the hostname may change over time. As an instance, a website's IP address may depend on what company is hosting the content; if a new company runs the webpage from a different location, its IP address will change. Thus, the URL of the webpage would have to change every time the IP address changes. Also, a user will now be unable to use the old URL from an existing set of bookmarks.
3. The same webpage may be available on multiple hosts, each with a different IP address. Identifying the site by name provides flexibility in deciding which server to contact. Thus, it is easier for you and me to contact *www.amazon.ca* rather than *www.amazon.com* because being in Canada, we assume that the former host machine will be closer (loyalties apart :)

Internet applications access DNS through a resolver, a software library linked with the application. A DNS resolver performs two main functions. The *gethostbyname()* function converts a hostname to an IP address, and the *gethostbyaddr()* function converts an IP address to a hostname. The resolver interacts with one or more DNS servers to perform these functions on behalf of the application. We explain the procedure in the DNS protocol section below.

DNS architecture:

DNS has a hierarchical name space with an unnamed root. The first layer of the tree contains the top-level domains. The top level domain includes the following:

1. **Generic or Organizational domains:** These include three character domains such as *.com* for commercial organizations, *.edu* for educational institutions in the U.S., *.gov*, *.net*, *.org* etc.

2. **Country domains:** These include two character domains ranging from Ascension Island *.ac* to Zimbabwe *.zw*. The United States is also listed as *.us*.

3. **ARPA domain:** A separate *.arpa* domain controls the translation of IP addresses into domain names.

The top-level domains are under the jurisdiction of the root server who knows all their IP addresses. Also, if the airline industry wants to be registered as the top-level domain *.aero* it has to obtain the permission of the root server to do so. The root server is run by the *Internet Corporation for Assigned Names and Numbers (ICANN).* Similarly the second-level domain servers are under the control of the first-level domain and so on. The root does not have to oversee the workings of the second-level domains. For example, if a new University called North Dakota State University was formed with the domain name *ndsu.edu* it would have to obtain the permission of the *.edu* server not the root.

The *in-addr.arpa* name space is a hierarchy based on the octets in the 32 bit IP address, starting with the beginning of the addressing hierarchy, i.e., the leftmost octet of the address. For example, finding the hostname associated with the IP address 12.34.56.78 would involve contacting a server for $12.in{-}addr.arpa$ first, then $12.34.in - arpa.arpa$ next and so on. Thus this translation proceeds in a manner reverse to *www.caam.rice.edu*, where one begins with *.edu* first, *rice.edu* next and so on. One reason being that in the latter case one does not know exactly how many labels are present in a hostname. For example, Rice University has two labels namely *rice.edu*, while Cambridge University in the U.K. is listed as *cam.ac.uk*.

DNS protocol:

For example, Kartik enters the URL *http://www.caam.rice.edu* on his web browser on *optlab.cas.mcmaster.ca*. The browser software would extract the domain name *www.caam.rice.edu* from the URL. To contact this web server, the browser must translate *www.caam.rice.edu* into an IP address. The browser software would invoke the *gethostbyname()* function, which would force the DNS resolver to contact one of the local DNS servers in McMaster University, to which the browser is configured. The local DNS server typically does not have the IP address of *www.caam.rice.edu*. This server contacts a root DNS server to learn the IP address of the *.edu* DNS server (which is under the jurisdiction of the root server; see the short discussion on DNS architecture below, and Sections 24.9-24.10 of Comer). The root server responds with this IP address. The local DNS server then sends the same query to the *.edu* server requesting the IP address of *www.caam.rice.edu*. Additional queries are sent by the local DNS server to the *rice.edu* and the *caam.rice.edu* servers in that order. Finally, the local DNS server learns the IP address of the *www.caam.rice.edu* server from the *caam.rice.edu* server, and it returns this IP address to the web browser via the DNS resolver. The DNS protocol operates in the reverse manner in translating an IP address to a hostname (see the discussion in the section of DNS architecture above).

DNS queries can be *recursive* or *iterative*. A recursive query requests that the receiving DNS server resolve the entire request itself. For example, the transaction between the DNS resolver and the local DNS resolver in the above example follows this pattern, since the local DNS server resolves the IP address of *www.caam.rice.edu* entirely on behalf of the DNS resolver. An iterative query requests that the receiving DNS respond directly to the DNS client with the IP address of the next DNS server in the hierarchy. In the above example, the transactions between the local DNS server and the various servers in the DNS hierarchy are iterative in nature. Iterative queries are preferable since one does not want to burden the root and the other servers in the hierarchy with the responsibility of completing the resolution process.

DNS servers also employ caching to reduce the latency in responding to queries, and to reduce the amount of DNS traffic in the internet. The local DNS server has a cache that stores the response sent to the resolver. In addition, the cache can store the IP address of each of the DNS servers involved in satisfying the query. For future queries, the local DNS server may be able to avoid contacting the root servers, and contact lower level servers in the DNS hierarchy, based on the cached information. Thus, the local DNS server first examines its cache to see if it has the necessary hostname to IP address binding, and carries out the DNS protocol only if it does not have one already. The resolver does not perform caching because the information would last the duration of the application, and besides this information could not be shared with other applications running on the same machine, or with other hosts on the same network. This caching is done based on a TTL (time to live) field. Each DNS response includes a TTL indicating the number of seconds that the response could be cached.

DNS primarily uses UDP for sending queries and responses (client-server model discussed in Chapter 21 of Comer), although TCP may also be used. Using UDP enables the resolver and DNS servers to communicate via single-packet messages without the overhead of establishing a connection. Most DNS queries and responses are short. In case, a UDP packet is lost, the client retransmits the query if no response is forthcoming within some time period.

DNS can also be used for accessing other kinds of information about hosts (not just their IP addresses); for instance if a host cannot receive email directly, but another machine will receive mail for it and pass it on, that information is communicated with a MX record in DNS.

The Telnet protocol:

We will cover Telnet with other applications such as FTP, HTTP, and electronic mail in one combined Lecture 7. The material on Telnet, although covered in 6th week of class, will not be part of the midterm exam.

Recommended Reading

1. Chapter 24 of Comer [1], and Chapter 14 of Stevens [2] for the DNS protocol. Section 5.3 of Krishnamurthy & Rexford [3] also has a good discussion of the DNS with respect to web protocols such as HTTP. Also, see Chapter 20 of Zwicky et al [4].

References

[1] D.E. Comer, *Internetworking with TCP/IP: Principles, Protocols, and Architectures*, 4th edition, Prentice Hall, NJ, 2000.

[2] W. Richard Stevens, *TCP/IP Illustrated, Volume I: The Protocols*, Addison Wesley Professional Computing Series, 1994.

[3] B. Krishnamurthy and J. Rexford, *Web Protocols and Practice*, Addison-Wesley, 2001.

[4] E.D. Zwicky, S. Cooper, and D.B. Chapman, *Building Internet Firewalls*, 2nd edition, O' Reilly, 2000.