PART 6

Application Layer

Objectives

The application layer enables the user, whether human or software, to access the network. It provides user interfaces and support for services such as electronic mail, file access and transfer, access to system resources, surfing the world wide web, and network management.

The application layer is responsible for providing services to the user.

In this part, we briefly discuss some applications that are designed as a client/server pair in the Internet. The client sends a request for a service to the server; the server responds to the client.

Part 6 of the book is devoted to the application layer and the services provided by this layer.

Chapters

This part consists of five chapters: Chapters 25 to 29.

Chapter 25

Chapter 25 discusses the Domain Name System (DNS). DNS is a client/server application that provides name services for other applications. It enables the use of application-layer addresses, such as an email address, instead of network layer logical addresses.

Chapter 26

Chapter 26 discusses three common applications in the Internet: remote login, electronic mail, and file transfer. A remote login application allows the user to remotely access the resources of a system. An electronic mail application simulates the tasks of snail mail at a much higher speed. A file transfer application transfers files between remote systems.

Chapter 27

Chapter 27 discuss the ideas and issues in the famous world wide web (WWW). It also briefly describes the client/server application program (HTTP) that is commonly used to access the web.

Chapter 28

Chapter 28 is devoted to network management. We first discuss the general idea behind network management. We then introduce the client/server application, SNMP, that is used for this purpose in the Internet. Although network management can be implemented in every layer, the Internet has decided to use a client/server application.

Chapter 29

Chapter 29 discusses multimedia and a set of wiely-used application programs. These programs have generated new issues such as the need for new protocols in other layers to handle the specific problems related to multimedia. We briefly discuss these issues in this chapter.

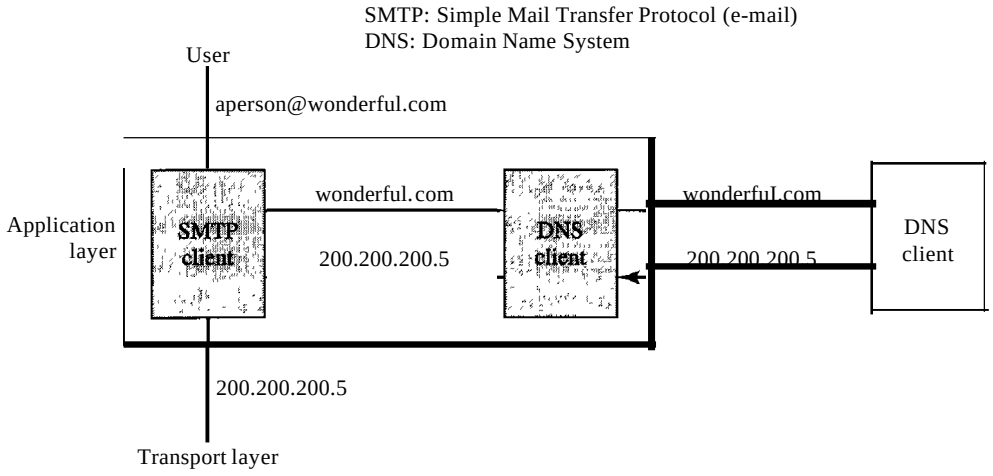
CHAPTER 25

Domain Name System

There are several applications in the application layer of the Internet model that follow the client/server paradigm. The client/server programs can be divided into two categories: those that can be directly used by the user, such as e-mail, and those that support other application programs. The Domain Name System (DNS) is a supporting program that is used by other programs such as e-mail.

Figure 25.1 shows an example of how a DNS client/server program can support an e-mail program to find the IP address of an e-mail recipient. A user of an e-mail program may know the e-mail address of the recipient; however, the IP protocol needs the IP address. The DNS client program sends a request to a DNS server to map the e-mail address to the corresponding IP address.

Figure 25.1 *Example of using the DNS service*



To identify an entity, TCPIIP protocols use the IP address, which uniquely identifies the connection of a host to the Internet. However, people prefer to use names instead of numeric addresses. Therefore, we need a system that can map a name to an address or an address to a name.

When the Internet was small, mapping was done by using a host file. The host file had only two columns: name and address. Every host could store the host file on its disk and update it periodically from a master host file. When a program or a user wanted to map a name to an address, the host consulted the host file and found the mapping.

Today, however, it is impossible to have one single host file to relate every address with a name and vice versa. The host file would be too large to store in every host. In addition, it would be impossible to update all the host files every time there was a change.

One solution would be to store the entire host file in a single computer and allow access to this centralized information to every computer that needs mapping. But we know that this would create a huge amount of traffic on the Internet.

Another solution, the one used today, is to divide this huge amount of information into smaller parts and store each part on a different computer. In this method, the host that needs mapping can contact the closest computer holding the needed information. This method is used by the Domain Name System (DNS). In this chapter, we first discuss the concepts and ideas behind the DNS. We then describe the DNS protocol itself.

25.1 NAME SPACE

To be unambiguous, the names assigned to machines must be carefully selected from a name space with complete control over the binding between the names and IP addresses. In other words, the names must be unique because the addresses are unique. A name space that maps each address to a unique name can be organized in two ways: fiat or hierarchical.

Flat Narne Space

In a flat name space, a name is assigned to an address. A name in this space is a sequence of characters without structure. The names mayor may not have a common section; if they do, it has no meaning. The main disadvantage of a fiat name space is that it cannot be used in a large system such as the Internet because it must be centrally controlled to avoid ambiguity and duplication.

Hierarchical Narne Space

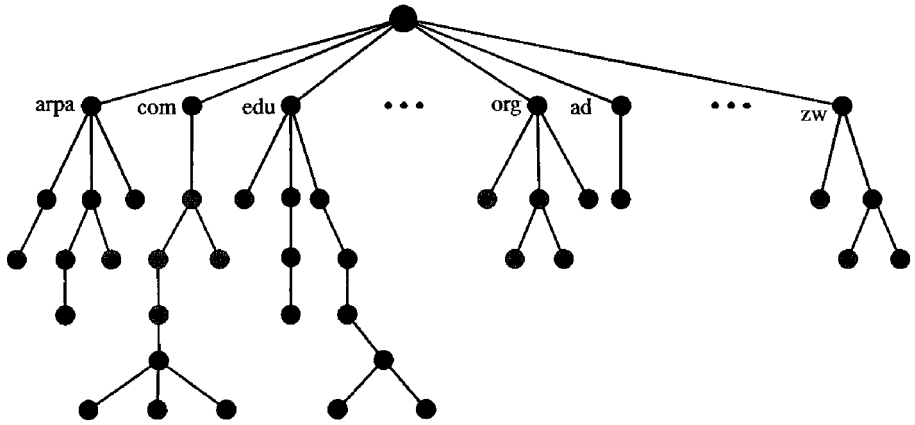
In a hierarchical name space, each name is made of several parts. The first part can define the nature of the organization, the second part can define the name of an organization, the third part can define departments in the organization, and so on. In this case, the authority to assign and control the name spaces can be decentralized. A central authority can assign the part of the name that defines the nature of the organization and the name of the organization. The responsibility of the rest of the name can be given to the organization itself. The organization can add suffixes (or prefixes) to the name to define its host or resources. The management of the organization need not worry that the prefix chosen for a host is taken by another organization because, even if part of an address is the

same, the whole address is different. For example, assume two colleges and a company call one of their computers *challenger.* The first college is given a name by the central authority such as *jhda.edu,* the second college is given the name *berkeley.edu,* and the company is given the name *smart.com.* When these organizations add the name *challenger* to the name they have already been given, the end result is three distinguishable names: *challenger.jhda.edu, challenger.berkeley.edu,* and *challenger.smart.com.* The names are unique without the need for assignment by a central authority. The central authority controls only part of the name, not the whole.

25.2 DOMAIN NAME SPACE

To have a hierarchical name space, a domain name space was designed. In this design the names are defined in an inverted-tree structure with the root at the top. The tree can have only 128 levels: level 0 (root) to level 127 (see Figure 25.2).

Figure 25.2 *Domain name space*



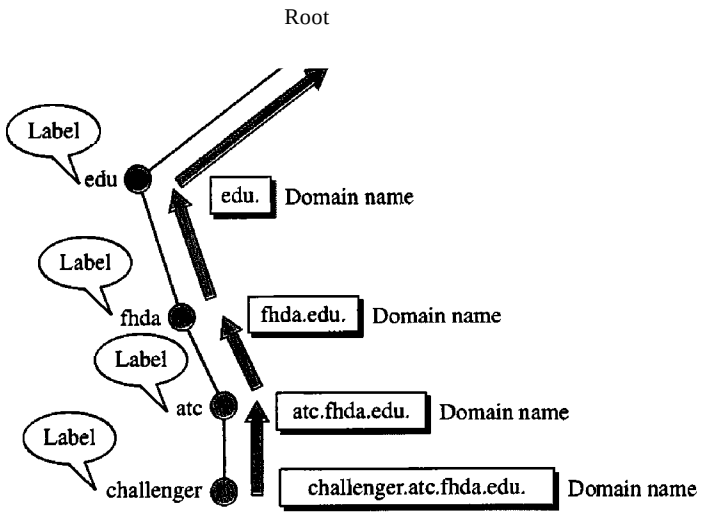
Label

Each node in the tree has a label, which is a string with a maximum of 63 characters. The root label is a null string (empty string). DNS requires that children of a node (nodes that branch from the same node) have different labels, which guarantees the uniqueness of the domain names.

Domain Name

Each node in the tree has a domain name. A full domain name is a sequence of labels separated by dots (.). The domain names are always read from the node up to the root. The last label is the label of the root (null). This means that a full domain name always ends in a null label, which means the last character is a dot because the null string is nothing. Figure 25.3 shows some domain names.

Figure 25.3 *Domain names and labels*



Fully Qualified Domain Name

If a label is terminated by a null string, it is called a fully qualified domain name (FQDN). An FQDN is a domain name that contains the full name of a host. It contains all labels, from the most specific to the most general, that uniquely define the name of the host. For example, the domain name

challenger.ate.tbda.edu.

is the FQDN of a computer named *challenger* installed at the Advanced Technology Center (ATC) at De Anza College. A DNS server can only match an FQDN to an address. Note that the name must end with a null label, but because null means nothing, the label ends with a dot (.).

Partially Qualified Domain Name

If a label is not terminated by a null string, it is called a partially qualified domain name (PQDN). A PQDN starts from a node, but it does not reach the root. It is used when the name to be resolved belongs to the same site as the client. Here the resolver can supply the missing part, called the suffix, to create an FQDN. For example, if a user at the *jhda.edu.* site wants to get the IP address of the challenger computer, he or she can define the partial name

challenger

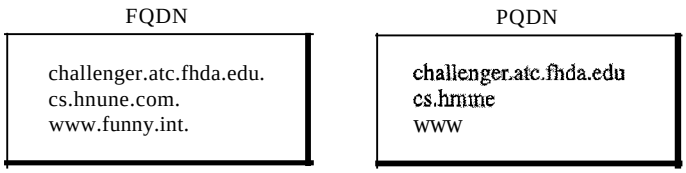
The DNS client adds the suffix *atc.jhda.edu.* before passing the address to the DNS server.

The DNS client normally holds a list of suffixes. The following can be the list of suffixes at De Anza College. The null suffix defines nothing. This suffix is added when the user defines an FQDN.

atc.fhda.edu
fhda.edu
null

Figure 25.4 shows some FQDNs and PQDNs.

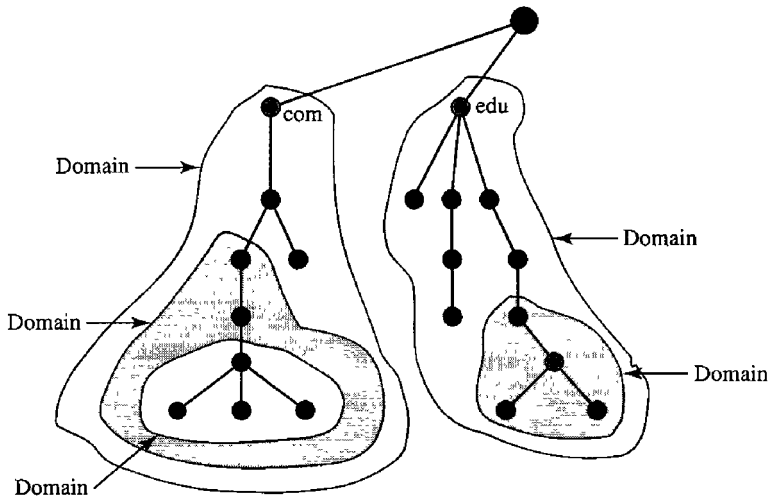
Figure 25.4 *FQDN and PQDN*



Domain

A **domain** is a subtree of the domain name space. The name of the domain is the domain name of the node at the top of the subtree. Figure 25.5 shows some domains. Note that a domain may itself be divided into domains (or **subdomains** as they are sometimes called).

Figure 25.5 *Domains*



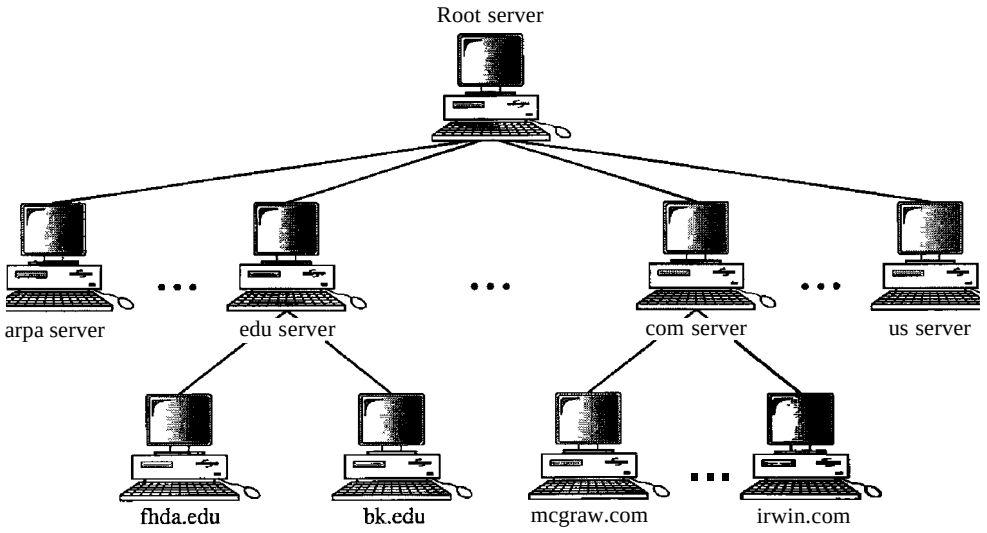
25.3 DISTRIBUTION OF NAME SPACE

The information contained in the domain name space must be stored. However, it is very inefficient and also unreliable to have just one computer store such a huge amount of information. It is inefficient because responding to requests from all over the world places a heavy load on the system. It is not unreliable because any failure makes the data inaccessible.

Hierarchy of Name Servers

The solution to these problems is to distribute the information among many computers called DNS servers. One way to do this is to divide the whole space into many domains based on the first level. In other words, we let the root stand alone and create as many domains (subtrees) as there are first-level nodes. Because a domain created in this way could be very large, DNS allows domains to be divided further into smaller domains (subdomains). Each server can be responsible (authoritative) for either a large or a small domain. In other words, we have a hierarchy of servers in the same way that we have a hierarchy of names (see Figure 25.6).

Figure 25.6 *Hierarchy of name servers*

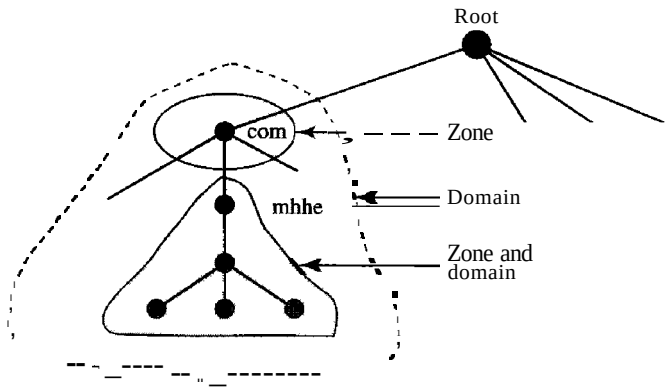


Zone

Since the complete domain name hierarchy cannot be stored on a single server, it is divided among many servers. What a server is responsible for or has authority over is called a zone. We can define a zone as a contiguous part of the entire tree. If a server accepts responsibility for a domain and does not divide the domain into smaller domains, the *domain* and the *zone* refer to the same thing. The server makes a database called a *zone file* and keeps all the information for every node under that domain. However, if a server divides its domain into subdomains and delegates part of its authority to other servers, *domain* and *zone* refer to different things. The information about the nodes in the subdomains is stored in the servers at the lower levels, with the original server keeping some sort of reference to these lower-level servers. Of course the original server does not free itself from responsibility totally: It still has a zone, but the detailed information is kept by the lower-level servers (see Figure 25.7).

A server can also divide part of its domain and delegate responsibility but still keep part of the domain for itself. In this case, its zone is made of detailed information for the part of the domain that is not delegated and references to those parts that are delegated.

Figure 25.7 *Zones and domains*



Root Server

A root server is a server whose zone consists of the whole tree. A root server usually does not store any information about domains but delegates its authority to other servers, keeping references to those servers. There are several root servers, each covering the whole domain name space. The servers are distributed all around the world.

Primary and Secondary Servers

DNS defines two types of servers: primary and secondary. A primary server is a server that stores a file about the zone for which it is an authority. It is responsible for creating, maintaining, and updating the zone file. It stores the zone file on a local disk.

A secondary server is a server that transfers the complete information about a zone from another server (primary or secondary) and stores the file on its local disk. The secondary server neither creates nor updates the zone files. If updaJing is required, it must be done by the primary server, which sends the updated version to the secondary.

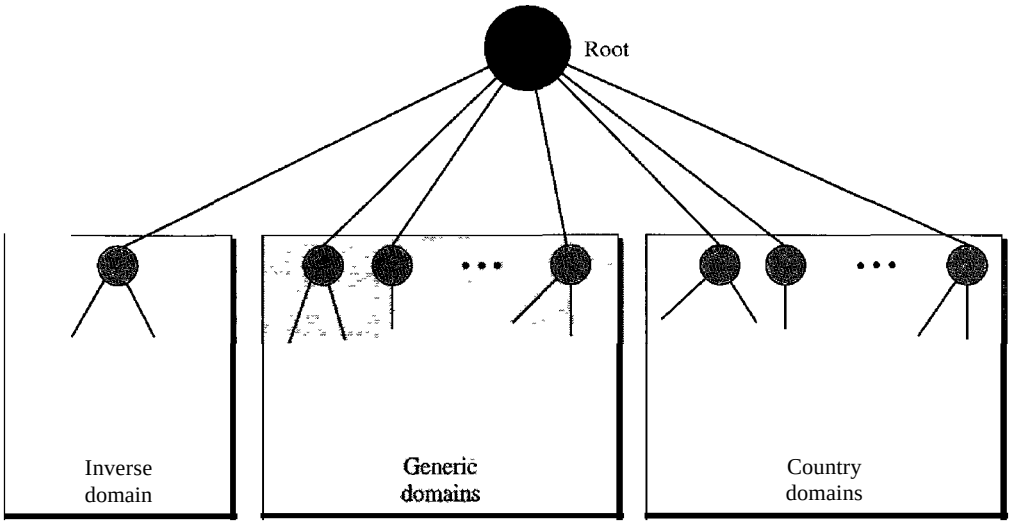
The primary and secondary servers are both authoritative for the zones they serve. The idea is not to put the secondary server at a lower level of authority but to create redundancy for the data so that if one server fails, the other can continue serving clients. Note also that a server can be a primary server for a specific zone and a secondary server for another zone. Therefore, when we refer to a server as a primary or secondary server, we should be careful to which zone we refer.

> A primary server loads all information from the disk file; the secondary server loads all information from the primary server. When the secondary downloads information from the primary, it is called zone transfer.

25.4 DNS IN THE INTERNET

DNS is a protocol that can be used in different platforms. In the Internet, the domain name space (tree) is divided into three different sections: generic domains, country domains, and the inverse domain (see Figure 25.8).

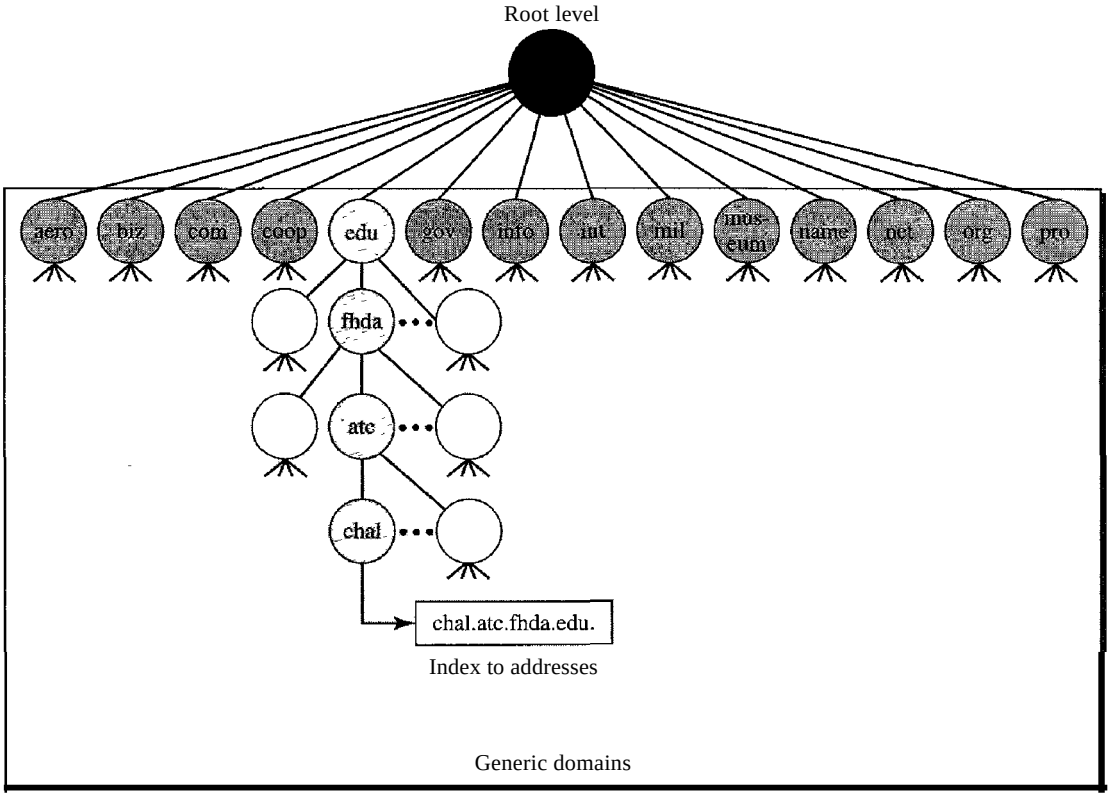
Figure 25.8 *DNS used in the Internet*



Generic Domains

The **generic domains** define registered hosts according to their generic behavior. Each node in the tree defines a domain, which is an index to the domain name space database (see Figure 25.9).

Figure 25.9 *Generic domains*



Looking at the tree, we see that the first level in the generic domains section allows 14 possible labels. These labels describe the organization types as listed in Table 25.1.

Table 25.1 *Generic domain labels*

Label	*Description*
aero	Airlines and aerospace companies
biz	Businesses or firms (similar to "com")
com	Commercial organizations
coop	Cooperative business organizations
edu	Educational institutions
gov	Government institutions
info	Information service providers
int	International organizations
mil	Military groups
museum	Museums and other nonprofit organizations
name	Personal names (individuals)
net	Network support centers
org	Nonprofit organizations
pro	Professional individual organizations

Country Domains

The country domains section uses two-character country abbreviations (e.g., us for United States). Second labels can be organizational, or they can be more specific, national designations. The United States, for example, uses state abbreviations as a subdivision of us (e.g., ca.us.).

Figure 25.10 shows the country domains section. The address *anza.cup.ca.us* can be translated to De Anza College in Cupertino, California, in the United States.

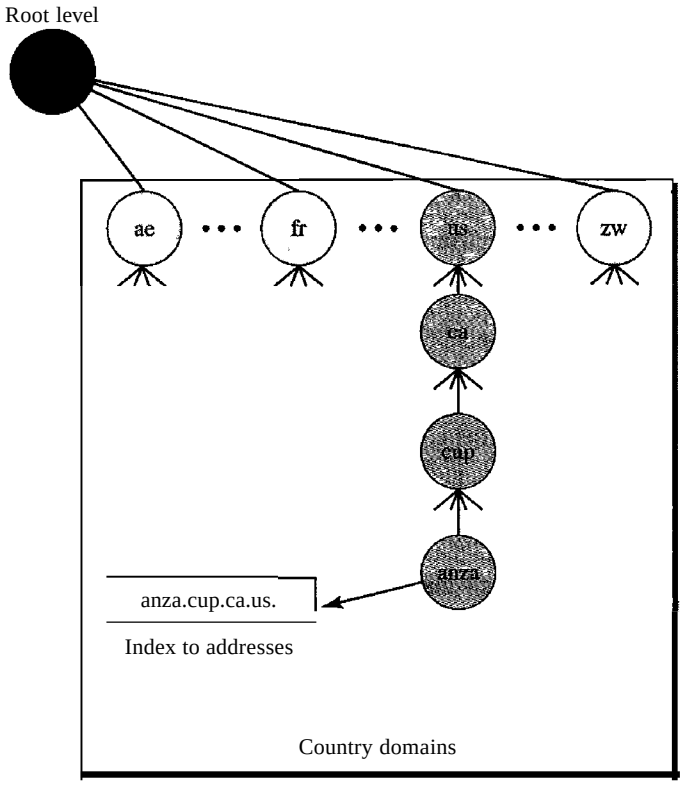
Inverse Domain

The inverse domain is used to map an address to a name. This may happen, for example, when a server has received a request from a client to do a task. Although the server has a file that contains a list of authorized clients, only the IP address of the client (extracted from the received IP packet) is listed. The server asks its resolver to send a query to the DNS server to map an address to a name to determine if the client is on the authorized list.

This type of query is called an inverse or pointer (PTR) query. To handle a pointer query, the inverse domain is added to the domain name space with the first-level node called *arpa* (for historical reasons). The second level is also one single node named *in-addr* (for inverse address). The rest of the domain defines IP addresses.

The servers that handle the inverse domain are also hierarchical. This means the netid part of the address should be at a higher level than the subnetid part, and the subnetid part

Figure 25.10 *Country domains*



higher than the hostid part. In this way, a server serving the whole site is at a higher level than the servers serving each subnet. This configuration makes the domain look inverted when compared to a generic or country domain. To follow the convention of reading the domain labels from the bottom to the top, an IF address such as 132.34.45.121 (a class B address with netid 132.34) is read as 121.45.34.132.in-addr. arpa. See Figure 25.11 for an illustration of the inverse domain configuration.

25.5 RESOLUTION

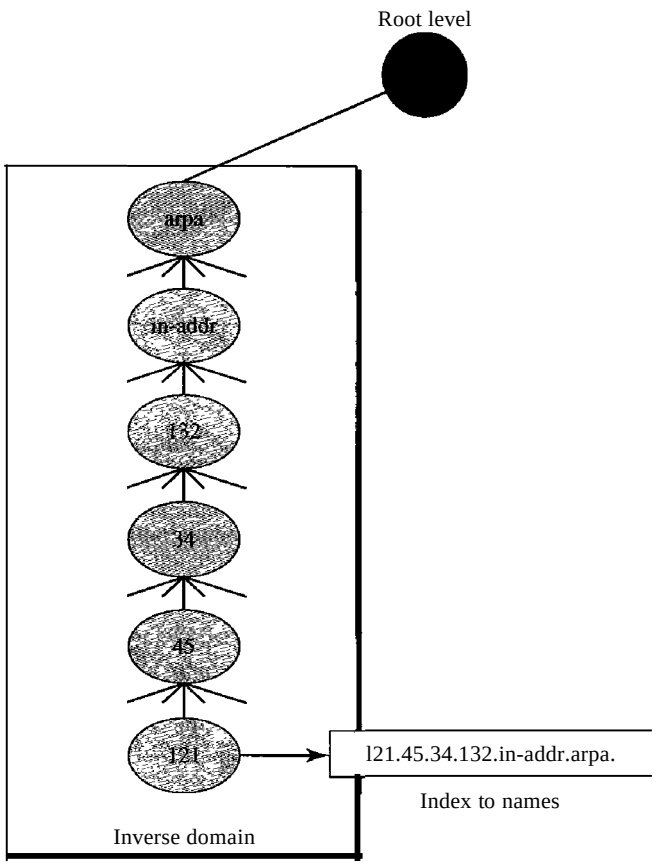
Mapping a name to an address or an address to a name is called *name-address resolution.*

Resolver

DNS is designed as a client/server application. A host that needs to map an address to a name or a name to an address calls a DNS client called a resolver. The resolver accesses the closest DNS server with a mapping request. If the server has the information, it satisfies the resolver; otherwise, it either refers the resolver to other servers or asks other servers to provide the infonnation.

After the resolver receives the mapping, it interprets the response to see if it is a real resolution or an error, and finally delivers the result to the process that requested it.

Figure 25.11 *Inverse domain*



Mapping Names to Addresses

Most of the time, the resolver gives a domain name to the server and asks for the corresponding address. In this case, the server checks the generic domains or the country domains to find the mapping.

If the domain name is from the generic domains section, the resolver receives a domain name such as *"chal.atc.jhda.edu."*. The query is sent by the resolver to the local DNS server for resolution. If the local server cannot resolve the query, it either refers the resolver to other servers or asks other servers directly.

If the domain name is from the country domains section, the resolver receives a domain name such as *"ch.jhda.cu.ca.us."*. The procedure is the same.

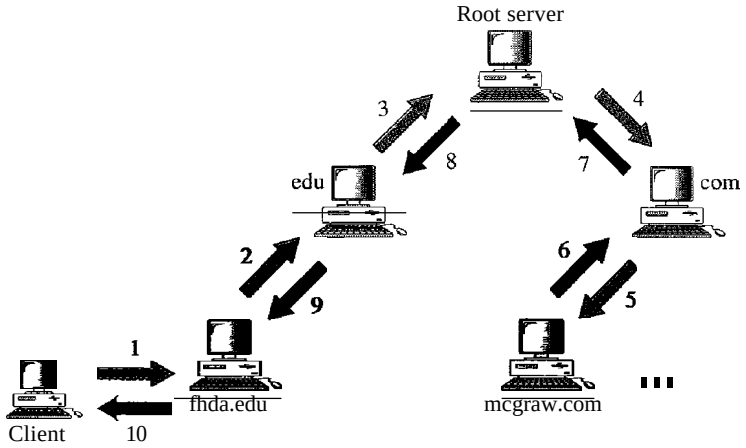
Mapping Addresses to Names

A client can send an IP address to a server to be mapped to a domain name. As mentioned before, this is called a PTR query. To answer queries of this kind, DNS uses the inverse domain. However, in the request, the IP address is reversed and the two labels *in-addr* and *arpa* are appended to create a domain acceptable by the inverse domain section. For example, if the resolver receives the IF address 132.34.45.121, the resolver first inverts the address and then adds the two labels before sending. The domain name sent is *"121.45.34.132.in-addr.arpa."* which is received by the local DNS and resolved.

Recursive Resolution

The client (resolver) can ask for a recursive answer from a name server. This means that the resolver expects the server to supply the final answer. If the server is the authority for the domain name, it checks its database and responds. If the server is not the authority, it sends the request to another server (the parent usually) and waits for the response. If the parent is the authority, it responds; otherwise, it sends the query to yet another server. When the query is finally resolved, the response travels back until it finally reaches the requesting client. This is called recursive resolution and is shown in Figure 25.12.

Figure 25.12 *Recursive resolution*



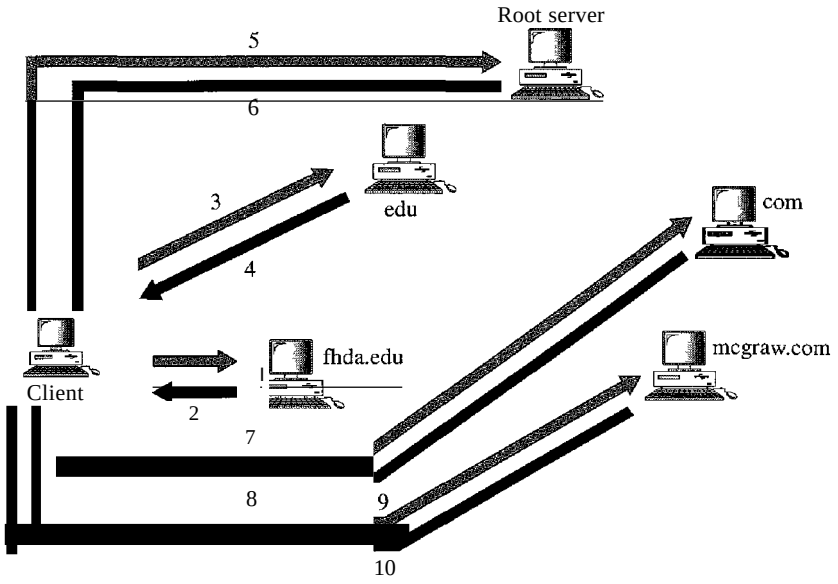
Iterative Resolution

If the client does not ask for a recursive answer, the mapping can be done iteratively. If the server is an authority for the name, it sends the answer. If it is not, it returns (to the client) the IP address of the server that it thinks can resolve the query. The client is responsible for repeating the query to this second server. If the newly addressed server can resolve the problem, it answers the query with the IP address; otherwise, it returns the IP address of a new server to the client. Now the client must repeat the query to the third server. This process is called iterative resolution because the client repeats the same query to multiple servers. In Figure 25.13 the client queries four servers before it gets an answer from the mcgraw.com server.

Caching

Each time a server receives a query for a name that is not in its domain, it needs to search its database for a server IP address. Reduction of this search time would increase efficiency. DNS handles this with a mechanism called caching. When a server asks for a mapping from another server and receives the response, it stores this information in its cache memory before sending it to the client. If the same or another client asks for the same mapping, it can check its cache memory and solve the problem. However, to

Figure 25.13 *Iterative resolution*



inform the client that the response is coming from the cache memory and not from an authoritative source, the server marks the response as *unauthoritative.*

Caching speeds up resolution, but it can also be problematic. If a server caches a mapping for a long time, it may send an outdated mapping to the client. To counter this, two techniques are used. First, the authoritative server always adds information to the mapping called *time-to-live* (TTL). It defines the time in seconds that the receiving server can cache the information. After that time, the mapping is invalid and any query must be sent again to the authoritative server. Second, DNS requires that each server keep a TTL counter for each mapping it caches. The cache memory must be searched periodically, and those mappings with an expired TTL must be purged.

25.6 DNS MESSAGES

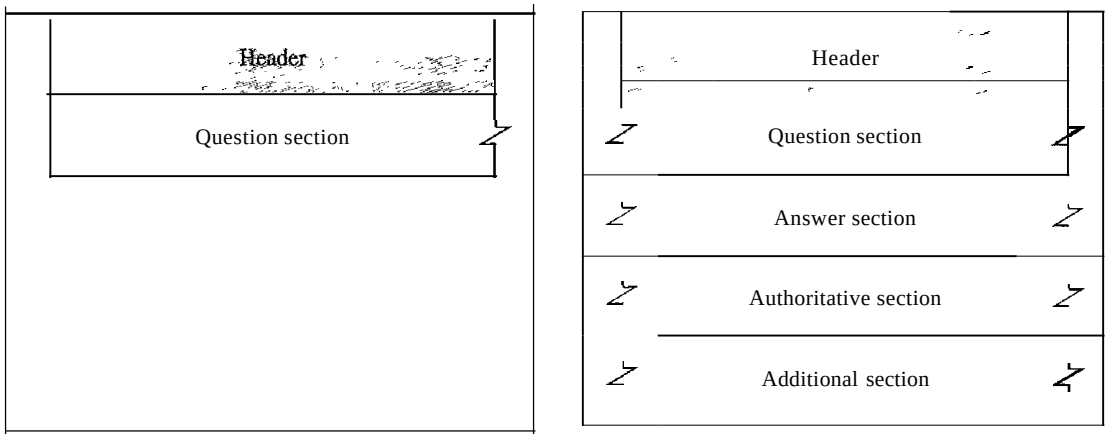
DNS has two types of messages: query and response. Both types have the same format. The query message consists of a header and question records; the response message consists of a header, question records, answer records, authoritative records, and additional records (see Figure 25.14).

Header

Both query and response messages have the same header format with some fields set to zero for the query messages. The header is 12 bytes, and its format is shown in Figure 25.15.

The *identification* subfield is used by the client to match the response with the query. The client uses a different identification number each time it sends a query. The server duplicates this number in the corresponding response. The *flags* subfield is a collection of

Figure 25.14 *Query and response messages*



a. Query b. Response

Figure 25.15 *Header format*

Identification	Flags
Number of question records	Number of answer records (all 0s in query message)
Number of authoritative records (all 0s in query message)	Number of additional records (all 0s in query message)

subfields that define the type of the message, the type of answer requested, the type of desired resolution (recursive or iterative), and so on. The *number of question records* subfield contains the number of queries in the question section of the message. The *number of answer records* subfield contains the number of answer records in the answer section of the response message. Its value is zero in the query message. The *number of authoritative records* subfield contains the number of authoritative records in the authoritative section of a response message. Its value is zero in the query message. Finally, the *number of additional records* subfield contains the number additional records in the additional section of a response message. Its value is zero in the query message.

Question Section

This is a section consisting of one or more question records. It is present on both query and response messages. We will discuss the question records in a following section.

Answer Section

This is a section consisting of one or more resource records. It is present only on response messages. This section includes the answer from the server to the client (resolver). We will discuss resource records in a following section.

Authoritative Section

This is a section consisting of one or more resource records. It is present only on response messages. This section gives infonnation (domain name) about one or more authoritative servers for the query.

Additional Information Section

This is a section consisting of one or more resource records. It is present only on response messages. This section provides additional infonnation that may help the resolver. For example, a server may give the domain name of an authoritative server to the resolver in the authoritative section, and include the IP address of the same authoritative server in the additional information section.

25.7 TYPES OF RECORDS

As we saw in Section 25.6, two types of records are used in DNS. The question records are used in the question section of the query and response messages. The resource records are used in the answer, authoritative, and additional information sections of the response message.

Question Record

A question record is used by the client to get information from a server. This contains the domain name.

Resource Record

Each domain name (each node on the tree) is associated with a record called the resource record. The server database consists of resource records. Resource records are also what is returned by the server to the client.

25.8 REGISTRARS

How are new domains added to DNS? This is done through a registrar, a commercial entity accredited by ICANN. A registrar first verifies that the requested domain name is unique and then enters it into the DNS database. A fee is charged.

Today, there are many registrars; their names and addresses can be found at

http://www.intenic.net

To register, the organization needs to give the name of its server and the IP address of the server. For example, a new commercial organization named *wonderful* with a server named *ws* and IP address 200.200.200.5 needs to give the following information to one of the registrars:

Domain name: ws.wonderful.com
IP address: 200.200.200.5

25.9 DYNAMIC DOMAIN NAME SYSTEM (DDNS)

When the DNS was designed, no one predicted that there would be so many address changes. In DNS, when there is a change, such as adding a new host, removing a host, or changing an IP address, the change must be made to the DNS master file. These types of changes involve a lot of manual updating. The size of today's Internet does not allow for this kind of manual operation.

The DNS master file must be updated dynamically. The Dynamic Domain Name System (DDNS) therefore was devised to respond to this need. In DDNS, when a binding between a name and an address is determined, the information is sent, usually by DHCP (see Chapter 21) to a primary DNS server. The primary server updates the zone. The secondary servers are notified either actively or passively. In active notification, the primary server sends a message to the secondary servers about the change in the zone, whereas in passive notification, the secondary servers periodically check for any changes. In either case, after being notified about the change, the secondary requests information about the entire zone (zone transfer).

To provide security and prevent unauthorized changes in the DNS records, DDNS can use an authentication mechanism.

25.10 ENCAPSULATION

DNS can use either UDP or TCP. In both cases the well-known port used by the server is port 53. UDP is used when the size of the response message is less than 512 bytes because most UDP packages have a 512-byte packet size limit. If the size of the response message is more than 512 bytes, a TCP connection is used. In that case, one of two scenarios can occur:

- If the resolver has prior knowledge that the size of the response message is more than 512 bytes, it uses the TCP connection. For example, if a secondary name server (acting as a client) needs a zone transfer from a primary server, it uses the TCP connection because the size of the information being transferred usually exceeds 512 bytes.
- If the resolver does not know the size of the response message, it can use the UDP port. However, if the size of the response message is more than 512 bytes, the server truncates the message and turns on the TC bit. The resolver now opens a TCP connection and repeats the request to get a full response from the server.

DNS can use the services of UDP or TCP using the well-known port 53.

25.11 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [...] refer to the reference list at the end of the text.

Books

DNS is discussed in [AL98], Chapter 17 of [For06], Section 9.1 of [PD03], and Section 7.1 of [Tan03].

Sites

The following sites are related to topics discussed in this chapter.

- www.intenic.net/Information about registrars
- www.ietf.org/rfc.html Information about RFCs

RFCs

The following RFCs are related to DNS:

799, 811, 819, 830, 881, 882, 883, 897, 920, 921, 1034,1035,1386,1480,1535,1536,1537, 1591, 1637, 1664, 1706, 1712, 1713, 1982, 2065, 2137, 2317, 2535, 2671

25.12 KEY TERMS

caching
country domain
DNS server
domain
domain name
domain name space
Domain Name System (DNS)
Dynamic Domain Name System (DDNS)
flat name space
fully qualified domain name (FQDN)
generic domain
hierarchical name space
host file
inverse domain
iterative resolution
label
name space
partially qualified domain name (PQDN)
primary server
query message
question record
recursive resolution
registrar
resolver
resource record
response message
root server
secondary server
subdomain
suffix
zone

25.13 SUMMARY

- The Domain Name System (DNS) is a client/server application that identifies each host on the Internet with a unique user-friendly name.
- DNS organizes the name space in a hierarchical structure to decentralize the responsibilities involved in naming.

- DNS can be pictured as an inverted hierarchical tree structure with one root node at the top and a maximum of 128 levels.
- Each node in the tree has a domain name.
- A domain is defined as any subtree of the domain name space.
- The name space information is distributed among DNS servers. Each server has jurisdiction over its zone.
- A root server's zone is the entire DNS tree.
- A primary server creates, maintains, and updates information about its zone.
- A secondary server gets its information from a primary server.
- The domain name space is divided into three sections: generic domains, country domains, and inverse domain.
- There are 14 generic domains, each specifying an organization type.
- Each country domain specifies a country.
- The inverse domain finds a domain name for a given IP address. This is called address-to-name resolution.
- Name servers, computers that run the DNS server program, are organized in a hierarchy.
- The DNS client, called a resolver, maps a name to an address or an address to a name.
- In recursive resolution, the client sends its request to a server that eventually returns a response.
- In iterative resolution, the client may send its request to multiple servers before getting an answer.
- Caching is a method whereby an answer to a query is stored in memory (for a limited time) for easy access to future requests.
- A fully qualified domain name (FQDN) is a domain name consisting of labels beginning with the host and going back through each level to the root node.
- A partially qualified domain name (PQDN) is a domain name that does not include all the levels between the host and the root node.
- There are two types of DNS messages: queries and responses.
- There are two types of DNS records: question records and resource records.
- Dynamic DNS (DDNS) automatically updates the DNS master file.
- DNS uses the services of UDP for messages of less than 512 bytes; otherwise, TCP is used.

25.14 PRACTICE SET

Review Questions

1. What is an advantage of a hierarchical name space over a flat name space for a system the size of the Internet?
2. What is the difference between a primary server and a secondary server?
3. What are the three domains of the domain name space?

4. What is the purpose of the inverse domain?
5. How does recursive resolution differ from iterative resolution?
6. What is an FQDN?
7. What is a PQDN?
8. What is a zone?
9. How does caching increase the efficiency of name resolution?
10. What are the two main categories of DNS messages?
11. Why was there a need for DDNS?

Exercises

12. Determine which of the following is an FQDN and which is a PQDN.
    a. xxx
    b. xxx.yyy.
    c. xxx.yyy.net
    d. zzz.yyy.xxx.edu.
13. Determine which of the following is an FQDN and which is a PQDN.
    a. mil.
    b. edu.
    c. xxx.yyy.net
    d. zzz.yyy.xxx.edu
14. Which domain is used by your system, generic or country?
15. Why do we need a DNS system when we can directly use an IP address?
16. To find the IP address of a destination, we need the service of DNS. DNS needs the service of UDP or TCP. UDP or TCP needs the service of IP. IP needs an IP destination address. Is this a vicious cycle here?
17. If a DNS domain name is *voyager.fhda.edu,* how many labels are involved here? How many levels of hierarchy?
18. Is a PQDN necessarily shorter than the corresponding FQDN?
19. A domain name is *hello.customer.info.* Is this a generic domain or a country domain?
20. Do you think a recursive resolution is normally faster than an interactive one? Explain.
21. Can a query message have one question section but the corresponding response message have several answer sections?