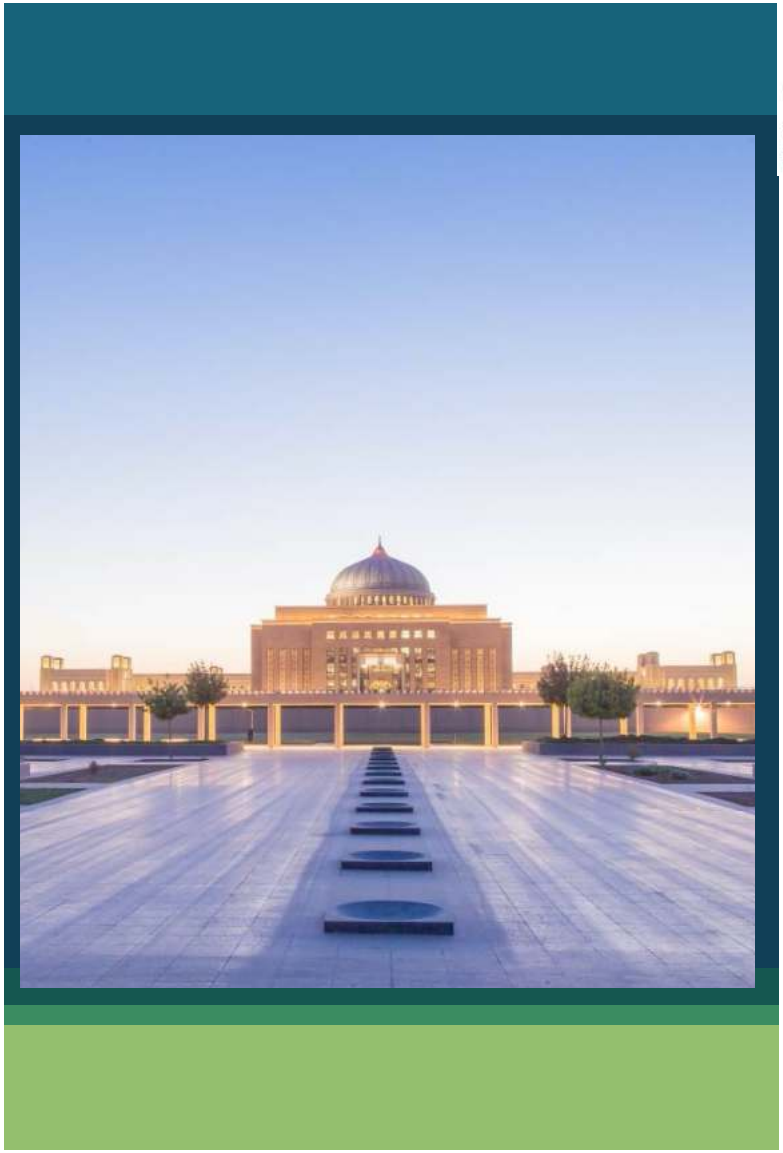


Faculty of Computer and Information Sciences

Information Technology Department

CCIS
كلية علوم الحاسب والمعلومات
College of Computer and Information Sciences

Network Protocols
Net323D

Chapter 5: Application Layer (DNS)

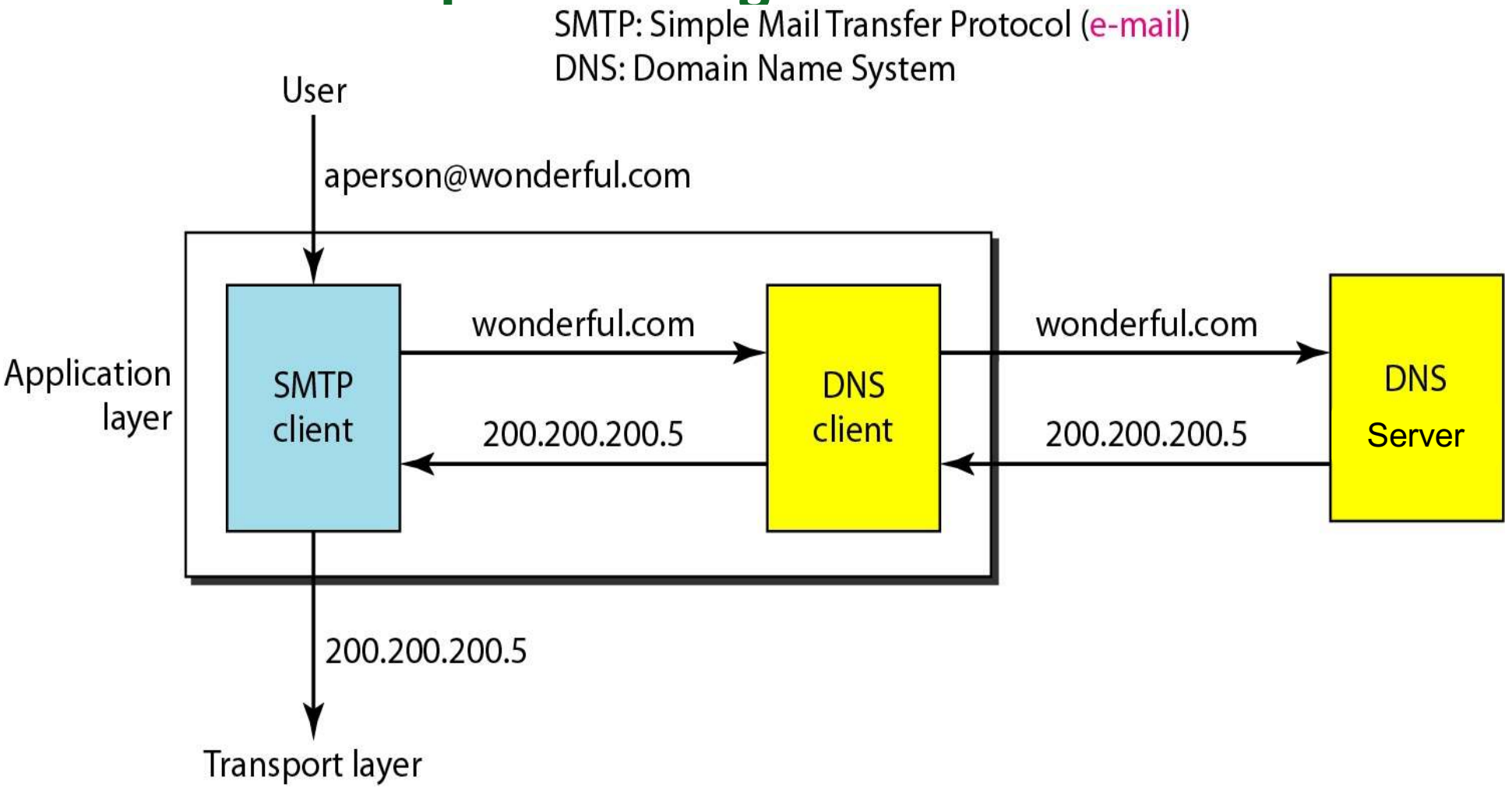
Introduction

- 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**:
  - Application that **directly used** by the user, such as **e-mail**,
  - Application that **support other** application programs.

- **The Domain Name System (DNS)** is a **supporting program** that is used by other programs such as e-mail.

Example of using the DNS service



Example of using the DNS service

- The Figure 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
- The **DNS client program** **sends** a **request** to a DNS **server** to **map** the **e-mail address** to the **corresponding** **IP** **address**
- After that **DNS** will **respond** to the **client** with the **IP address**
- The **well-known** port number of **DNS** is **53**

Introduction cont.

- To identify an entity, **TCP/IP** 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**.
- Which is the **DNS server**

Introduction cont.

- 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**.

Introduction cont.

- When a program or a user wanted to map a name to an address, the host consulted the host file and found the mapping

- **But now it is difficult**

- **The solution is to use : DNS server**

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.

Topics discussed in this section

- Flat Name Space
- Hierarchical Name Space

1- Flat Name 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 main disadvantage of a **flat** namespace 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**.

2- Hierarchical Name 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 namespaces can be **decentralized**

DOMAIN NAMESPACE

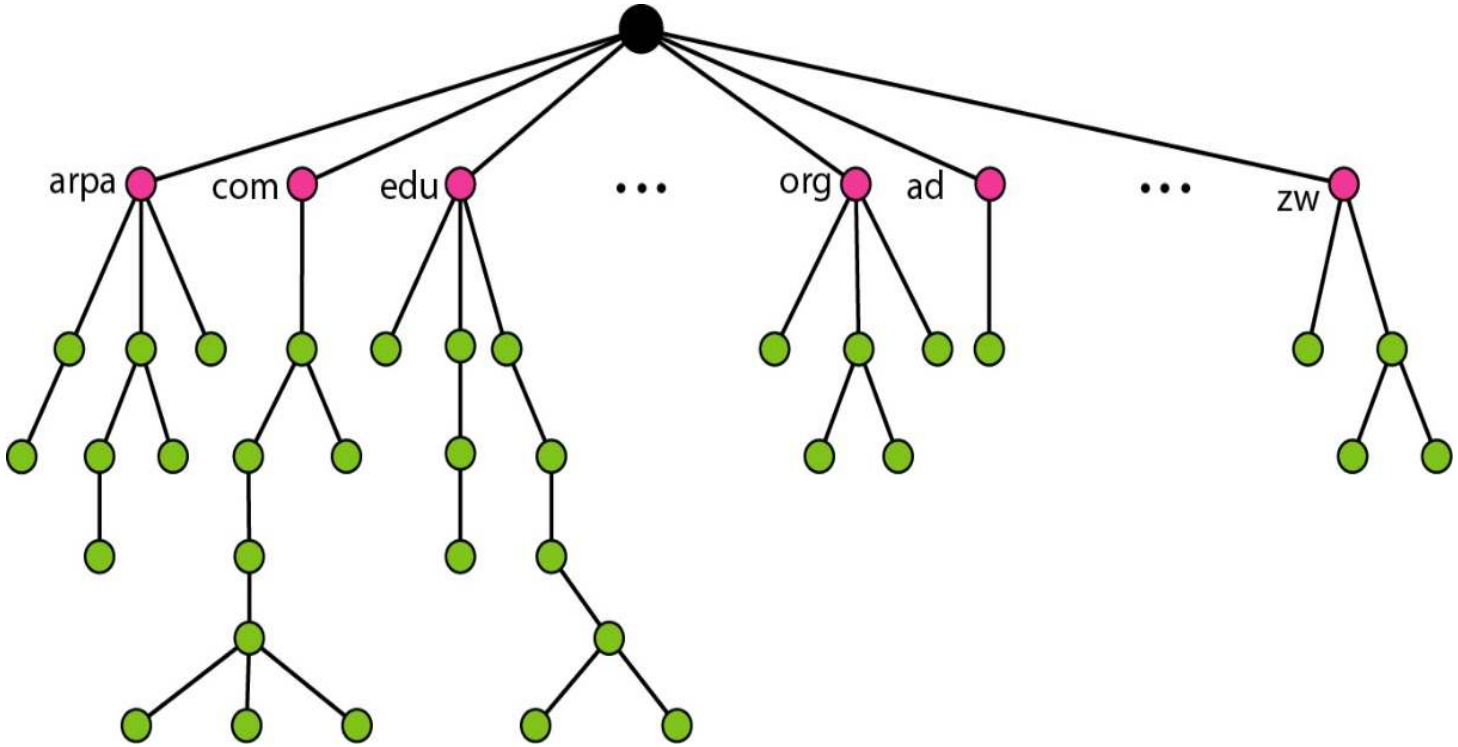
To have a hierarchical name space, a domain namespace was designed. In this design the names are defined in an **inverted-tree** structure with the **root** at the **top**. The tree can have only **128** levels: **level 0** (root) to **level 127**.

Topics discussed in this section:

- **Label**
- **Domain Name**
- **Domain**

DOMAIN NAMESPACE cont.

Figure 25.2 ***Domain name space***



DOMAIN NAMESPACE cont.

- **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).

- **Domain Name**
  - Each **node** in the **tree** has a **domain name**. A full domain name is a sequence of **labels separated** by **dots**

DOMAIN NAMESPACE cont.

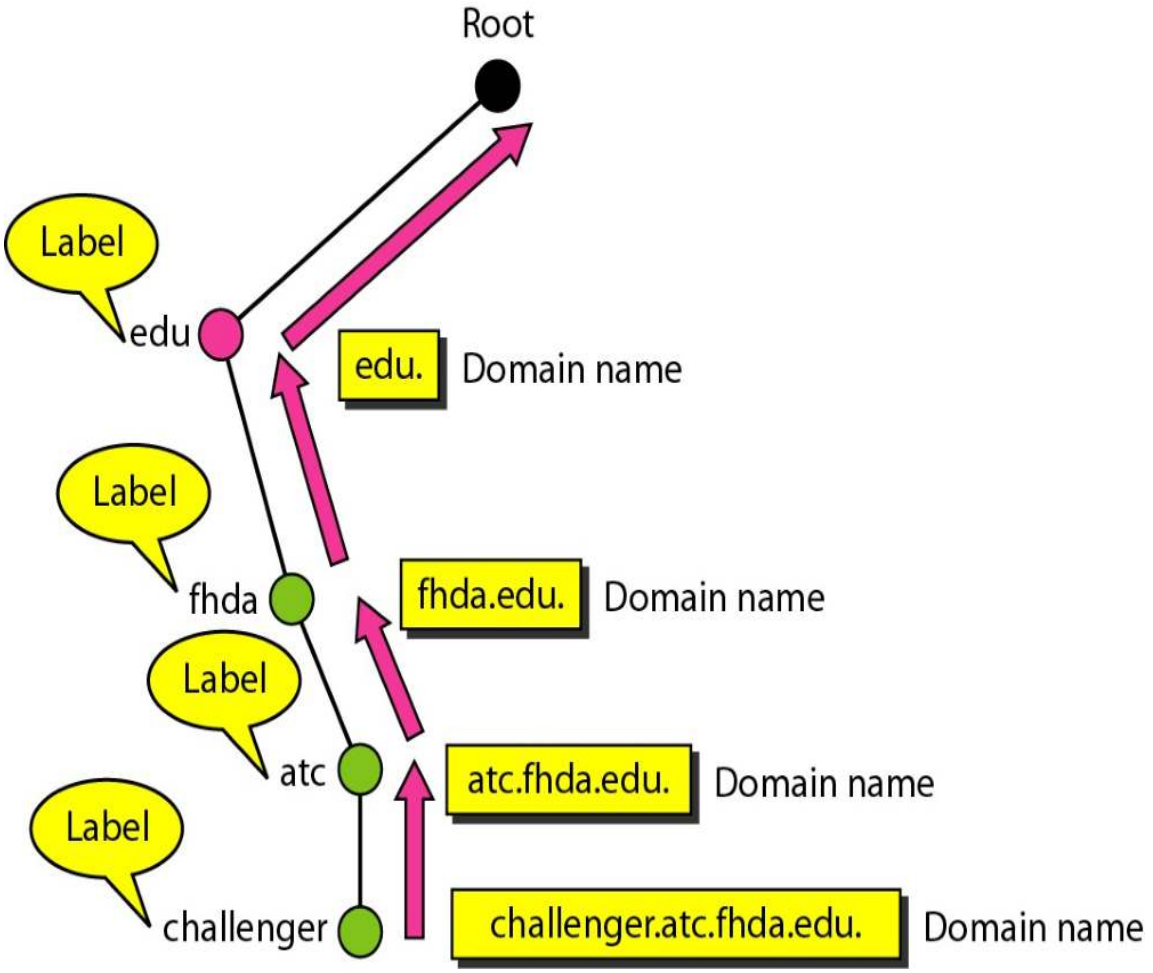


Figure 25.3

Domain names and labels

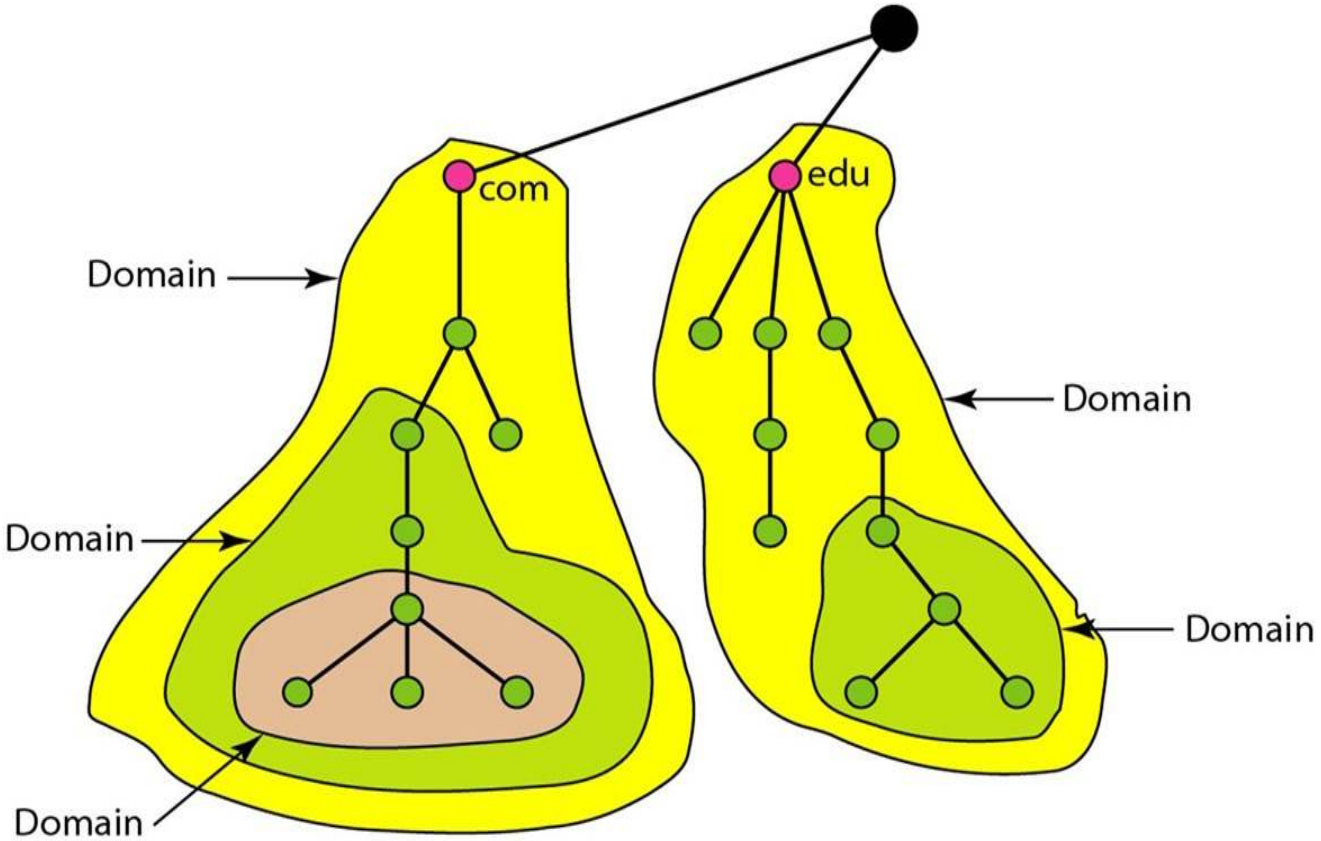
DOMAIN NAMESPACE cont.

Domain :

- A **domain** is a **sub tree** of the **domain namespace**.
- The name of the domain is the domain name of the node at the top of the subtree.

DOMAIN NAMESPACE cont.

Figure 25.5 ***Domains***



DISTRIBUTION OF NAMESPACE

- The information contained in the domain namespace must be stored.
- However, it is **very inefficient** and also **unreliable** to have **just one computer** store such a **huge amount** of information.
- In this section, we discuss the distribution of the domain name space.

Topics discussed in this section:

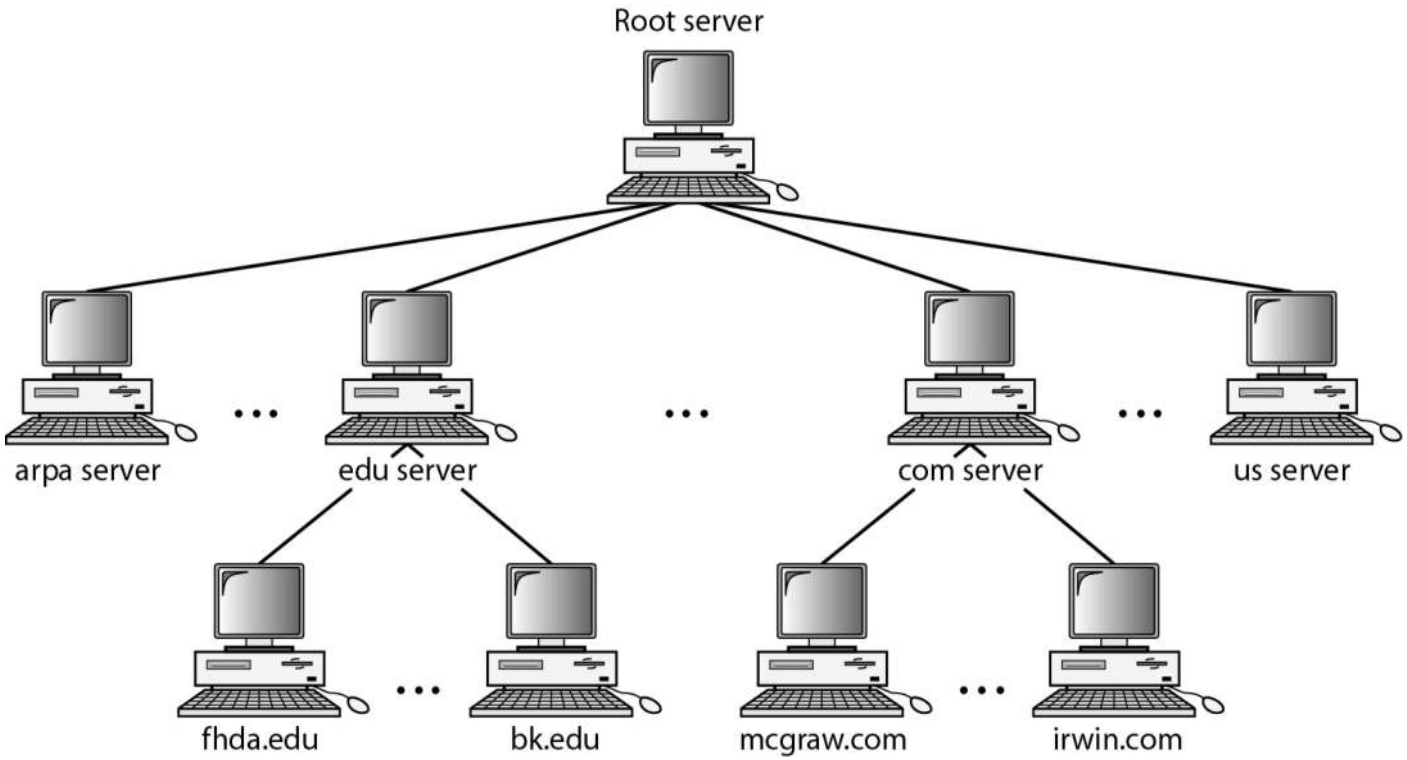
- **Hierarchy of Name Servers**
- **Zone**
- **Root Server**
- **Primary and Secondary Servers**

DISTRIBUTION OF NAME SPACE

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**.
- 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**)

Figure 25.6 *Hierarchy of name servers*

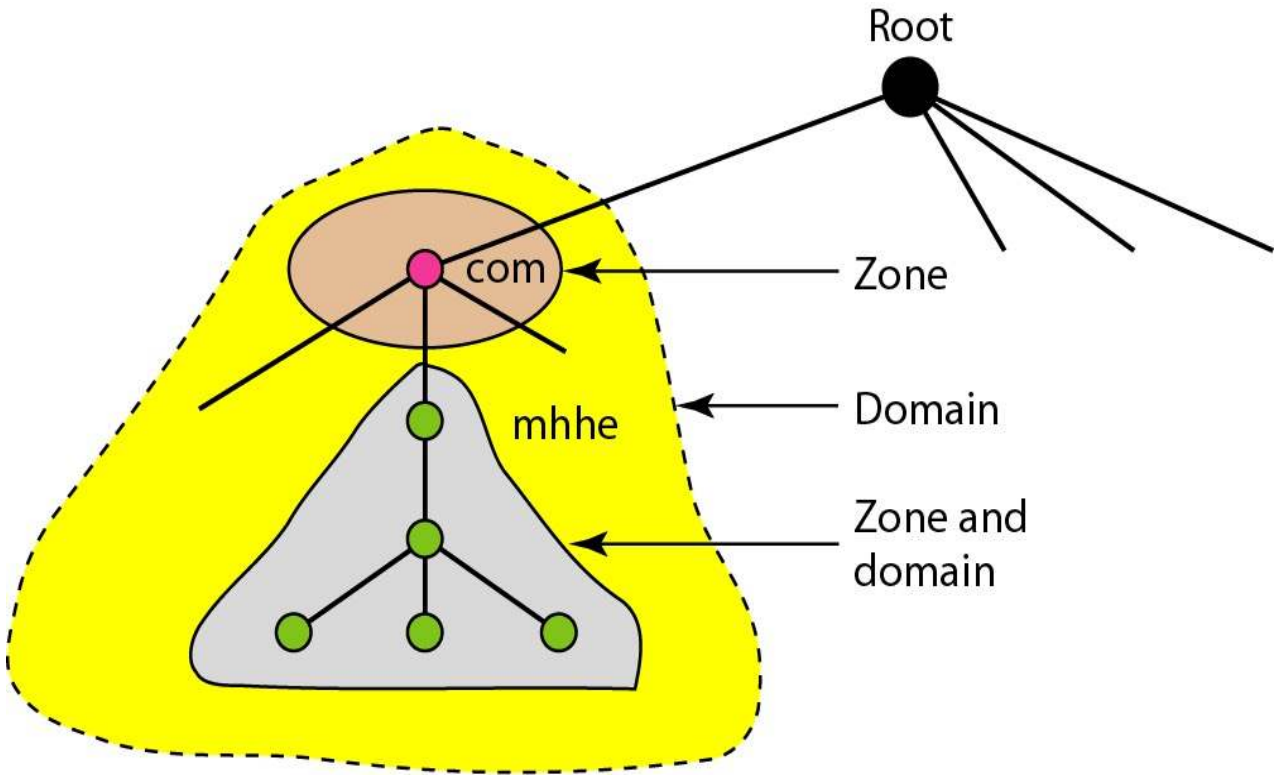


DISTRIBUTION OF NAME SPACE cont.

Zone :

- The server is responsible for or has authority over is called **a zone**
- The server makes a **database** called a ***zone file*** and keeps all the **information** for **every node** under that **domain**
- If a **server divides** its domain into **subdomains** and **delegates part** of its **authority** to **other servers**, ***domain*** and ***zone*** refer to **different** things

Figure 25.7 *Zones and domains*



DISTRIBUTION OF NAME SPACE cont.

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 namespace**

DISTRIBUTION OF NAME SPACE cont.

Primary and Secondary Servers

Primary server:

- 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**

DISTRIBUTION OF NAME SPACE cont.

Secondary server:

- 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 **updating** is required, it must be **done** by the **primary server**, which **sends** the **updated version** to the **secondary**.

Note

- 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**.

Note

- 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.**

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**.

Topics discussed in this section:

- **Generic Domains**
- **Country Domains**
- **Inverse Domain**

DNS IN THE INTERNET

Generic Domains

- The **generic domains** define **registered hosts** according to their generic behavior.

- **Each node** in the **tree defines** a **domain**

- **Eaxmple :** .com , .gov

Figure 25.8 *DNS IN THE INTERNET*

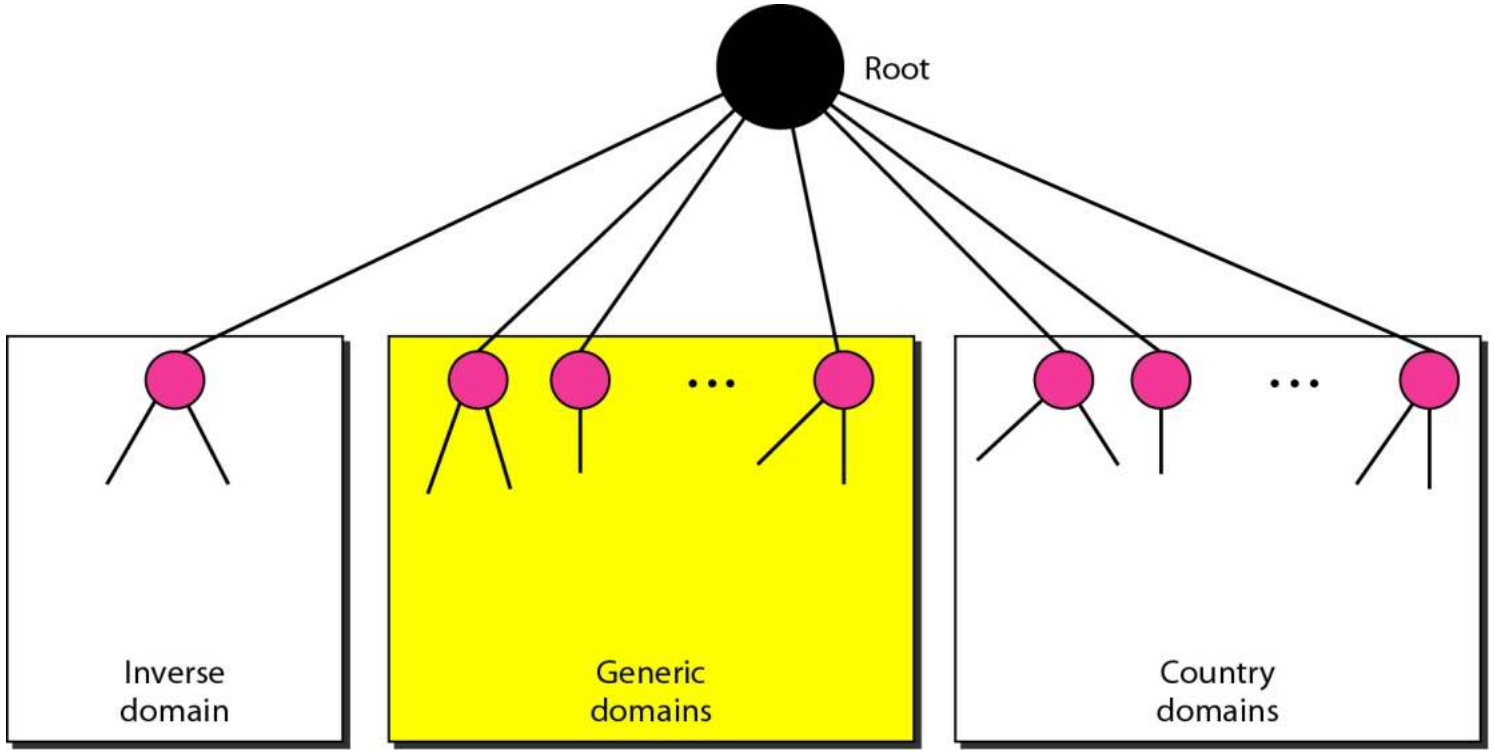


Figure 25.9 *Generic domains*

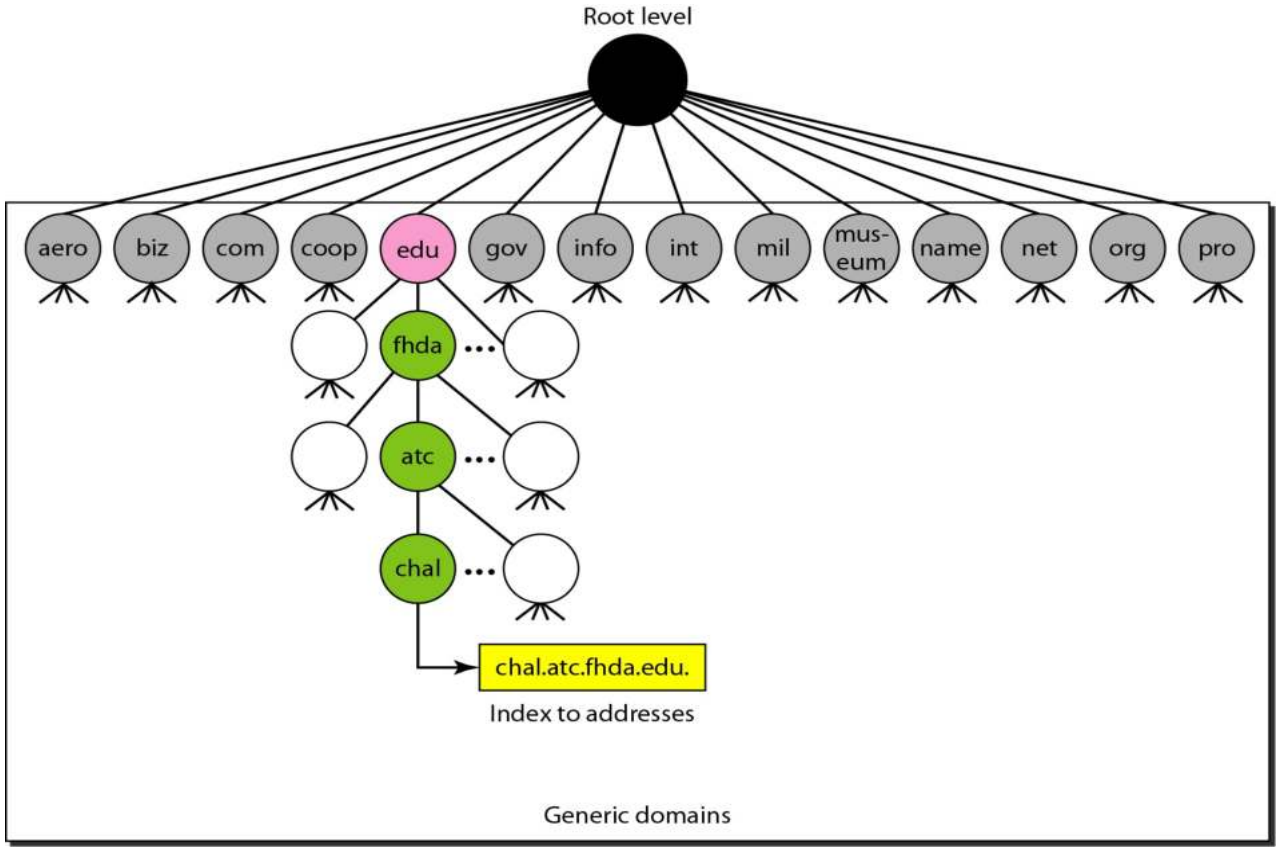


Table 25.1 *Generic domain labels*

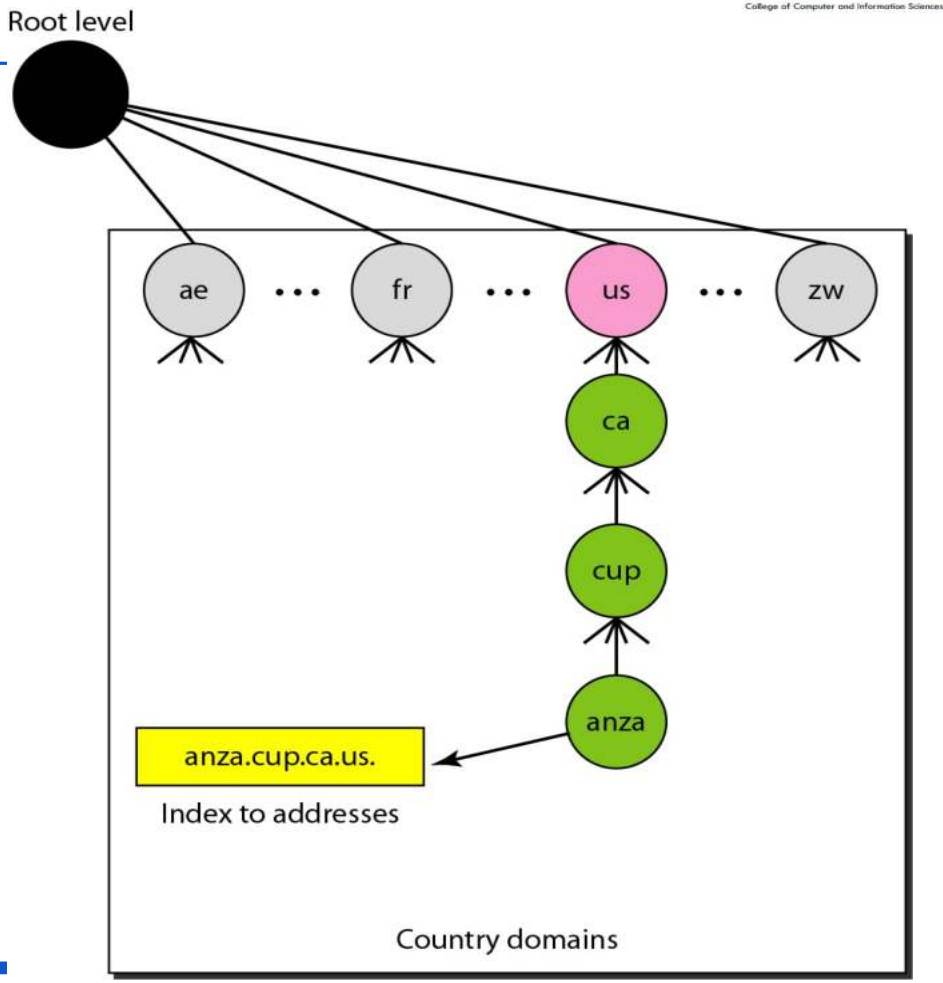
Label	*Description*
aero	Airlines and aerospace companies
biz	Businesses or firms (similar to "com")
com	Commercial organizations
coop	Cooperative business organizations
edu	Educational institutions
gov	Government institutions
info	Information service providers
int	International organizations
mil	Military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

DNS IN THE INTERNET cont.

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 *Country domains*



RESOLUTION

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

Topics discussed in this section:

- **Resolver**
- **Mapping Names to Addresses**
- **Mapping Addresses to Names**
- **Caching**

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 information .

1-Mapping Names to Addresses

- 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.

2-Mapping Addresses to Names

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.

2-Mapping Addresses to Names

Inverse domain

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 namespace 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**.

2-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.**

DNS uses the inverse domain.

- 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 IP 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**.

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**.

Book Chapter/ References or Other materials: Data Communications and Networking, Fourth Edition , 2007 The McGraw-Hill Companies, Inc. Chapter 25

THANK YOU