Application Level

Network core: routers, network of networks

Network edge: applications and hosts

end systems (hosts):

run application programs

e.g., WWW, email

client/server model:

client host requests, receives service from server

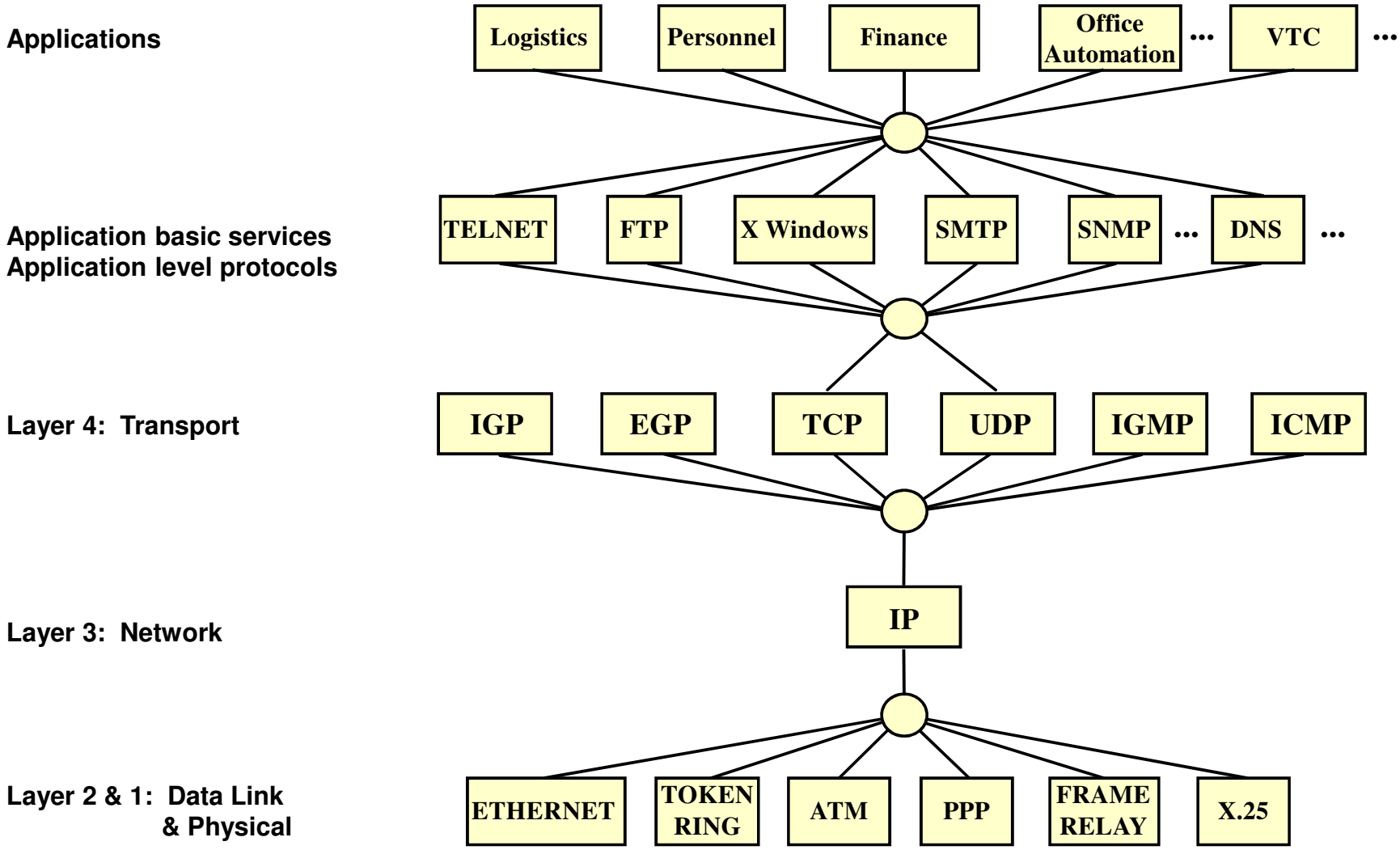
e.g., WWW client (browser)/ server; email client/server

peer-peer model:

host interaction symmetric

e.g.: teleconferencing

Everything over IP & IP over Everything



Internet protocols provide:

- General-purpose facility for reliable data transfer
- Mechanism for contacting hosts

Application-level protocols provide high-level services:

- DNS, Electronic mail, Remote login,FTP, World Wide Web

All of these applications use *client-server* architecture

Application programs

- Use internet protocols to contact other applications
- Application must interact with protocol software *before* contact is made
- *Listening* application informs local protocol software that it is ready to accept incoming messages
- *Connecting* application uses internet protocol to contact listener
- Applications exchange messages through resulting connection
- Provide *user-level* services

Application Level

Enterprise Systems:

- Engineering/Manufacturing Systems
- Business/Office Systems

Application Systems:

- User Interfaces
- Processing Programs
- Databases and files

Application Support Services:

- Client/Server support
- Distributed OS

Application: Transport Service Requirements

Data loss

- –some apps (e.g., audio) can tolerate some loss
- –other apps (e.g., file transfer, telnet) require 100% reliable data transfer

Timing

- –some apps (e.g., Internet telephony, interactive games) require low delay to be “effective”

Bandwidth

- –some apps (e.g., multimedia) require minimum amount of bandwidth to be “effective”
- –other apps (“elastic apps”) make use of whatever bandwidth they get

Transport Service Requirements of Common Apps

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	loss-tolerant	elastic	no
real-time audio/video	loss-tolerant	audio: 5Kb-1Mb video:10Kb-5Mb	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few Kbps up	yes, 100's msec
financial apps	no loss	elastic	yes and no

Services Provided by Internet Transport Protocols

TCP service:

- *connection-oriented:* setup required between client, server
- Client establishes connection to server
- Client terminates connection
- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not providing:* timing, minimum bandwidth guarantees

UDP service:

- unreliable data transfer between sending and receiving process
- does not provide: connection setup, reliability, flow control, congestion control, timing, or bandwidth guarantee
- Message must fit in one UDP datagram

Some services use both

- DNS

Internet Applications: their protocols and transport protocols

Application	Application layer protocol	Underlying transport protocol
e-mail	smtp [RFC 821]	TCP
remote terminal access	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
file transfer	ftp [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
remote file server	NFS	TCP or UDP
Internet telephony	proprietary (e.g., Vocaltec)	typically UDP

Traditional Distributed Applications

Those based on Internet: distributed applications

Have the following features:

Application logic

Application protocol support code

Example: The X protocol for transmitting graphical images

Transport interface code

Makes the appropriate network calls to send and receive the messages that make up the application protocol over a specific network transport

Usually divided into transport-independent and transport-dependent parts

Middleware provides transparency of the transport interface code

Software between application programs and OS/network.

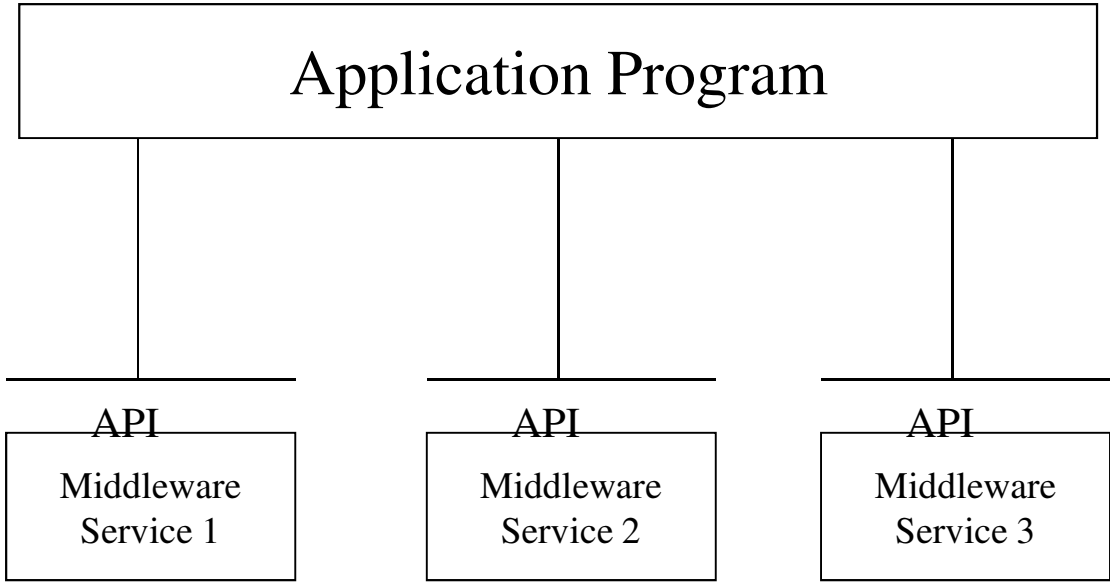
Provides a set of higher-level distributed computing capabilities and a set of standards-based interfaces.

Interfaces allow applications to be distributed and to take advantage of other services provided over the network.

Middleware can be viewed as a set of (transport) services that are accessible to application programmers through an API (Application Programming Interface)

Example: Sockets, RPC.

Middleware & API



API: Application Programming Interface

Defines interface between application and transport layer

Socket API: specific Internet API

Two processes communicate by sending data into socket, reading data out of socket

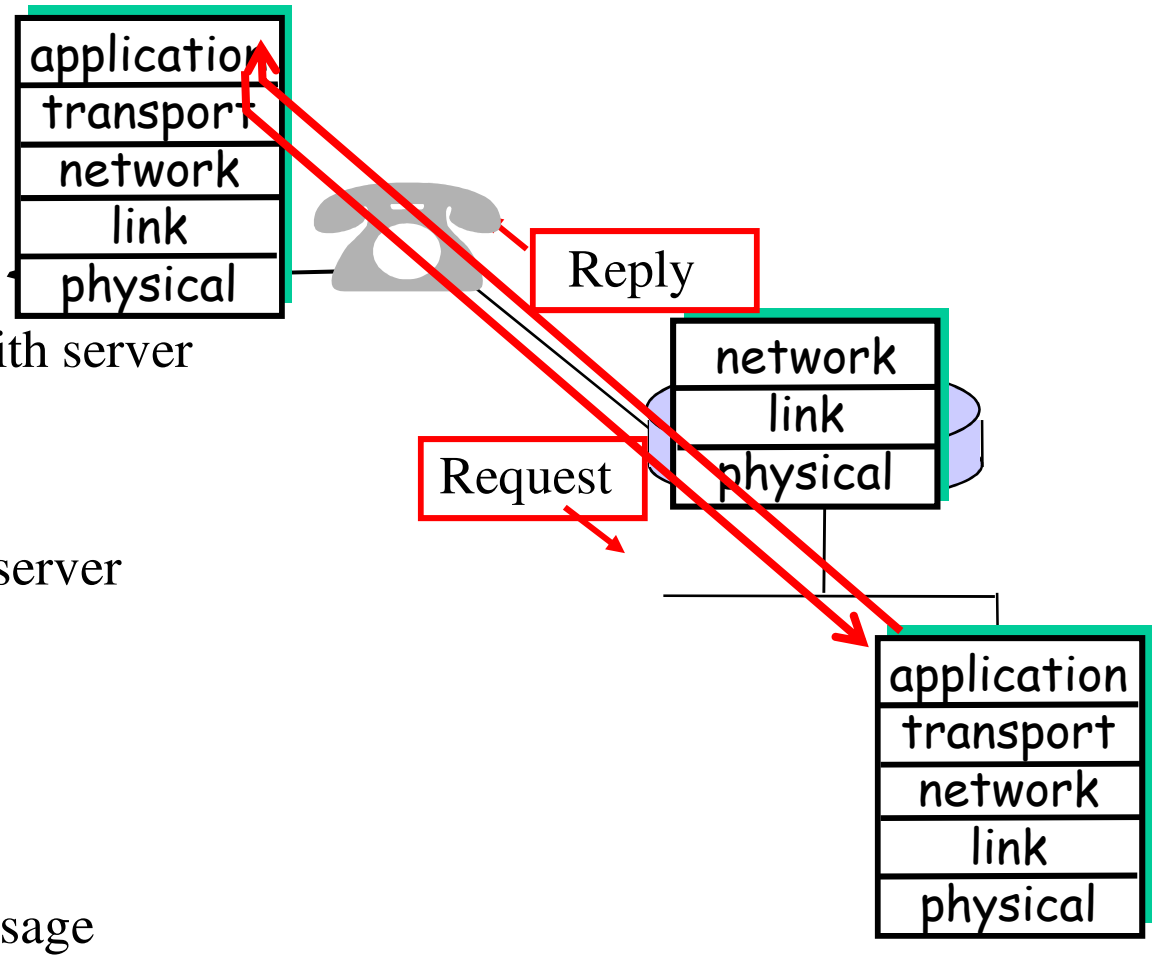
- Defined by programming/operating system
- Includes collection of procedures for application program
- Protocols do not typically specify API
- API defined by programming system
- Allows greatest flexibility - compatibility with different programming systems
- Originated with Berkeley BSD UNIX, now available on Windows 95 and Windows NT, Solaris, etc.

Client-server model

Typical network application has two pieces: *client* and *server*

Client application

- initiates contact (connection) with server ("speaks first")
- Sends message to server
- typically requests service from server
- e.g.: request WWW page
- Waits for return message



Server application is ``listener''

- Waits for incoming message
- Performs service
- Returns results
- e.g., sends requested WWW page.

Domain Name System

The **Domain Name System** (DNS) provides translation between symbolic names for IP hosts and their IP addresses

–People: use many identifiers: name, Passport #, …

–Internet hosts:

- IP address (32 bit) - used for addressing in IP datagrams
- "name", e.g., www.iitb.eirnet.ie - used by humans

Provides logical hierarchical view of the Internet

– DNS: globally *distributed database* implemented in hierarchy of many *name servers*

Functioning:

–*application-layer protocol* to communicate to *resolve* names (address/name translation): application calls *resolver*

–A client/server interaction

clients: query servers to resolve names (*nslookup* function)

servers: run name server daemons, reply to queries *(bind, named*)

–gethostbyname: UNIX based resolver library call that can be invoked from an application program

DNS: Example

Host named: xyz.iitb.ronet.ro wants IP address of web server: www.ibm.com

Resolver Steps:

1. After checking locally, contacts its local DNS server:

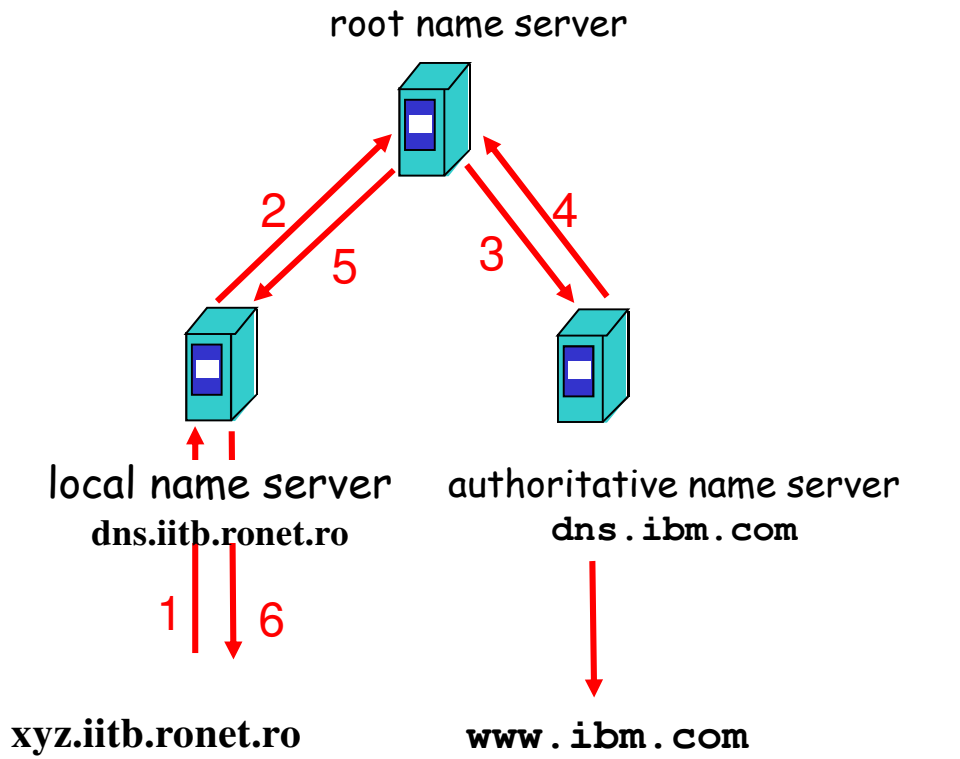
   dns.iitb.ronet.ro

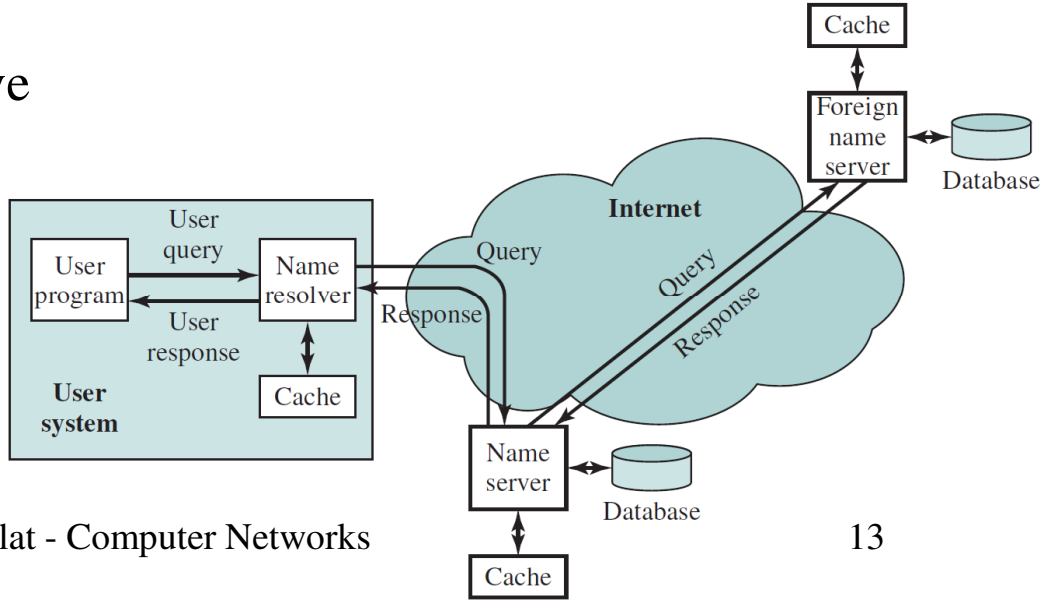
2. dns.iitb.ronet.ro contacts root name server, if necessary

3. root name server contacts authoritative name server (responsible server):

   dns.ibm.com, if necessary

4,5,6 Return of info





DNS Name Servers

Centralized DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance
- doesn't *scale!*

Distributed DNS:

- no server has all name-to-IP address mappings

Local Name Servers:

- each organization/ISP has *local (default) name server*
- host DNS query first goes to local name server

Authoritative Name Server:

- for a host: stores that host's IP address & name
- can perform name/address translation for that host's name

Root Name Server:

contacts authoritative name server if name mapping not known

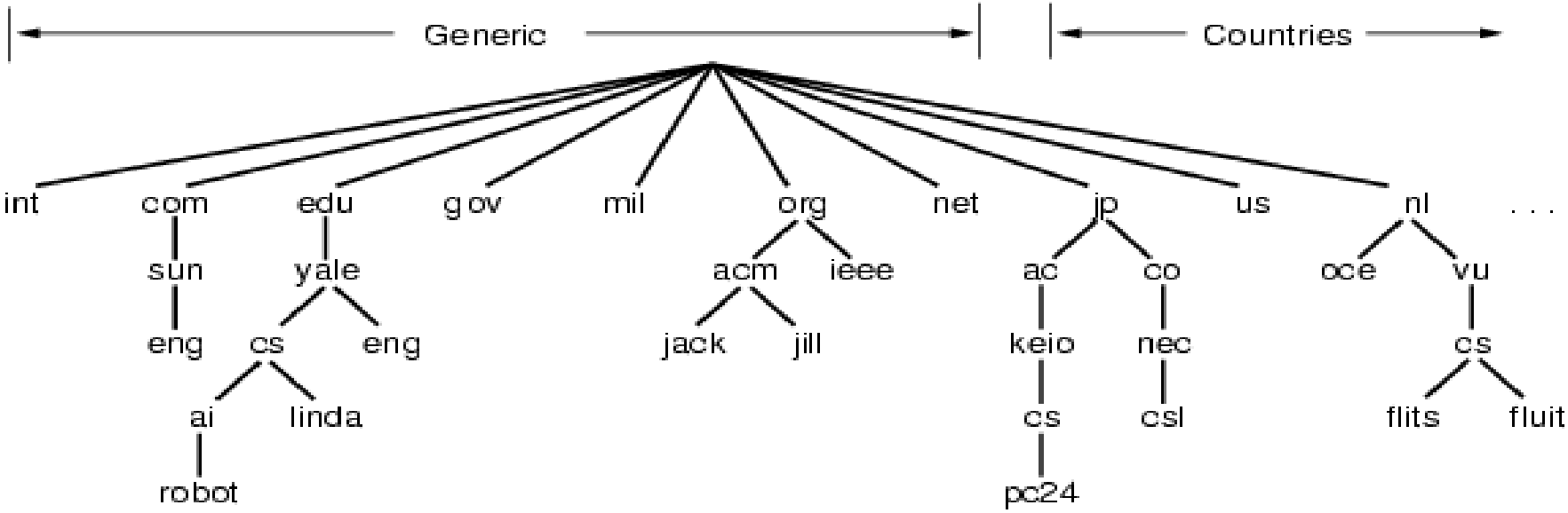
gets mapping

returns mapping to local name server

Several root name servers worldwide

Structure of DNS names

- Each name consists of a sequence of alphanumeric components separated by periods; domain names are case insensitive; each component name up to 63 characters long, entire path up to 255 characters
- Examples: www.eg.bucknell.edu www.netbook.cs.purdue.edu charcoal.eg.bucknell.edu
- Names are hierarchical, with most-significant component on the right
- Left-most component is the computer name



A portion of the Internet domain name space.

DNS naming structure

Top level domains (right-most components; also known as *TLD*s) defined by global authority (only Internet authority, what will be for the future?)

com	Commercial organization
edu	Educational institution
gov	Government organization
mil	Military organization

Organizations apply for names in a top-level domain:
bucknell.edu macdonalds.com
Organizations determine own internal structure
csis.ul.ie cs.purdue.edu

Geographic structure

Top-level domains are US-centric (e.g., cs.yale.edu could be accessed as cs.yale.ct.us), each organization in US is under a generic domain
Geographic TLDs are used for organizations in other countries:

TLD	Country
.uk	United Kingdom
.fr	France
.ch	Switzerland
.ie	Ireland

Countries define their own internal hierarchy: ac.uk and .edu.au are used for academic organizations in the United Kingdom and Australia; ac.uk or co.uk are mirrors for .edu or .com
Romania makes not (yet) that distinction, all organizations are under .ro

Choosing DNS server architecture

- Small organizations can use a single DNS server
  - Easy to administer
  - Inexpensive
- Large organizations often use multiple servers
  - Reliability through redundancy
  - Improved response time through load-sharing
  - Delegation of naming authority
- Locality of reference applies - users will most often look up names of computers within same organization

Domain names within an organization

Organizations can create any internal DNS hierarchy

- Uniqueness of TLD and organization name guarantee uniqueness of any internal name (much like file names in your directories)
- All but the left-most component of a domain name, is called the *domain* for that name:

Name	Domain
www.netbook.cs.purdue.edu	netbook.cs.purdue.edu
regulus.eg.bucknell.edu	eg.bucknell.edu
coral.bucknell.edu	bucknell.edu

Authority for creating new *subdomains* is delegated to each domain

- Administrator of bucknell.edu has authority to create eg.bucknell.edu and need not contact any central naming authority

DNS names and physical location

- DNS domains are logical concepts and need not correspond to physical location of organizations
- DNS domain for an organization can span multiple networks
  - utcluj.ro covers all networks at UTCN
  - cs.utcluj.ro all from Computer Science department
  - laptop.cs.utcluj.ro could be connected to a network in … California

Abbreviations

- May be convenient to use abbreviations for local computers; e.g. coral for coral.bucknell.edu
- Abbreviations are handled in the *resolver*; DNS servers only know *full-qualified domain names* (FQDNs)
- Local resolver is configured with list of suffixes to append
- Suffixes are tried sequentially until match found

DNS: Name Resolution

Root name server:

–may not know authoritative name server

–may know *intermediate name server* to contact to find authoritative name server

–Two ways of action:

Recursive queries:

–puts burden of name resolution on contacted name server (each server, having not requested information, goes & finds it, reporting back)

–not scalable under heavy load

Iterated queries:

–contacted server replies only with the name of next server to contact

DNS: Name Resolution (continued)

- Resolver software typically available as library procedures
  - Implement DNS application protocol
  - Configured for local servers
  - Example - UNIX *gethostbyname*
- Calling program is *client*
  - Constructs DNS protocol message - a *DNS request*
  - Sends message to local DNS server
- DNS *server* resolves name
  - Constructs DNS protocol message - a *DNS reply*
  - Sends message to client program and waits for next request

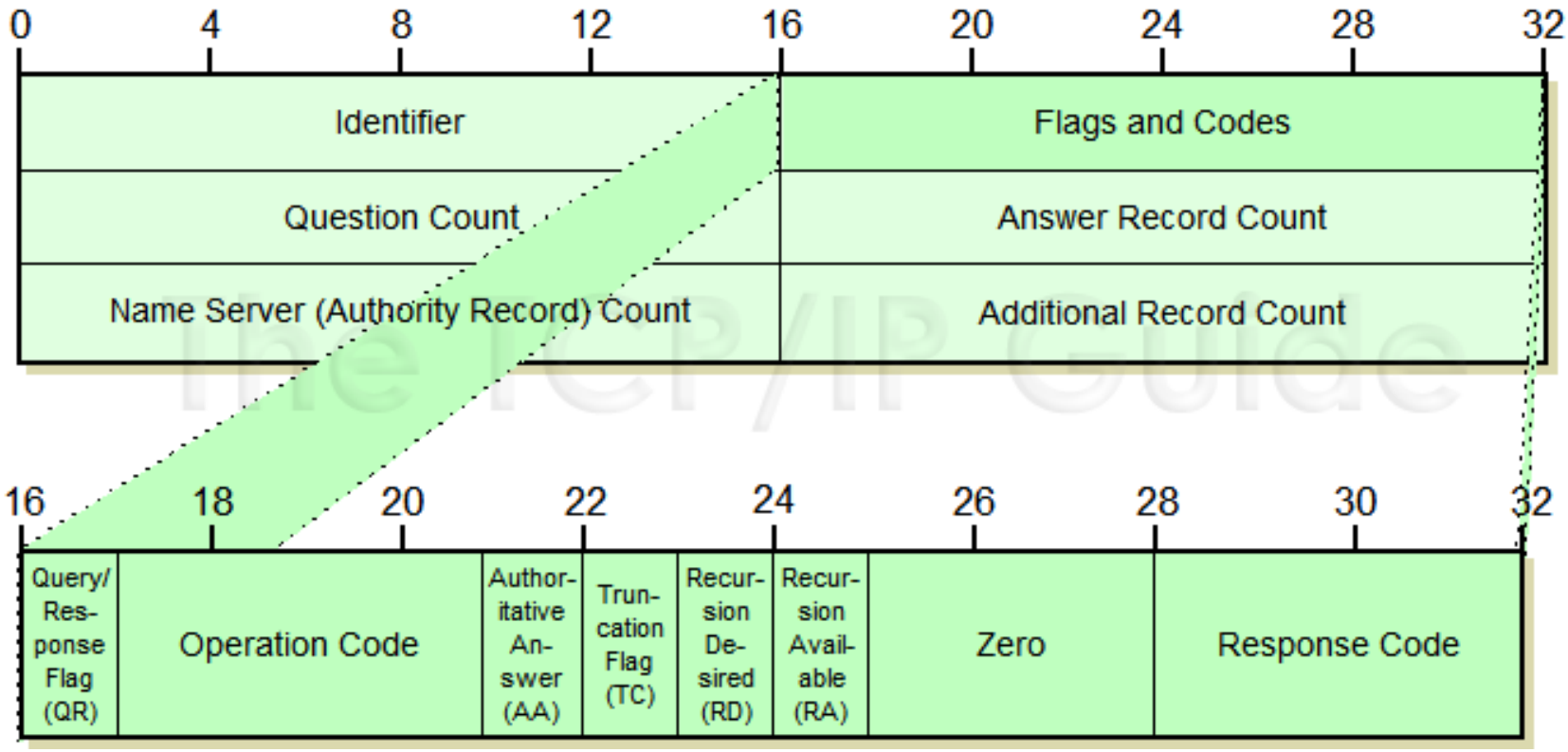
DNS: Database

- Contains:
  - ***resource records (RRs)*** that include the name, IP address, and other information about hosts
- Key features:
  - *Variable-depth hierarchy for names*: DNS allows essentially unlimited levels and uses the period (.) as the level delimiter in printed names
  - *Distribution controlled by the database*: DNS database is divided into thousands of separately managed zones, which are managed by separate administrators. The database software controls distribution and update of records.

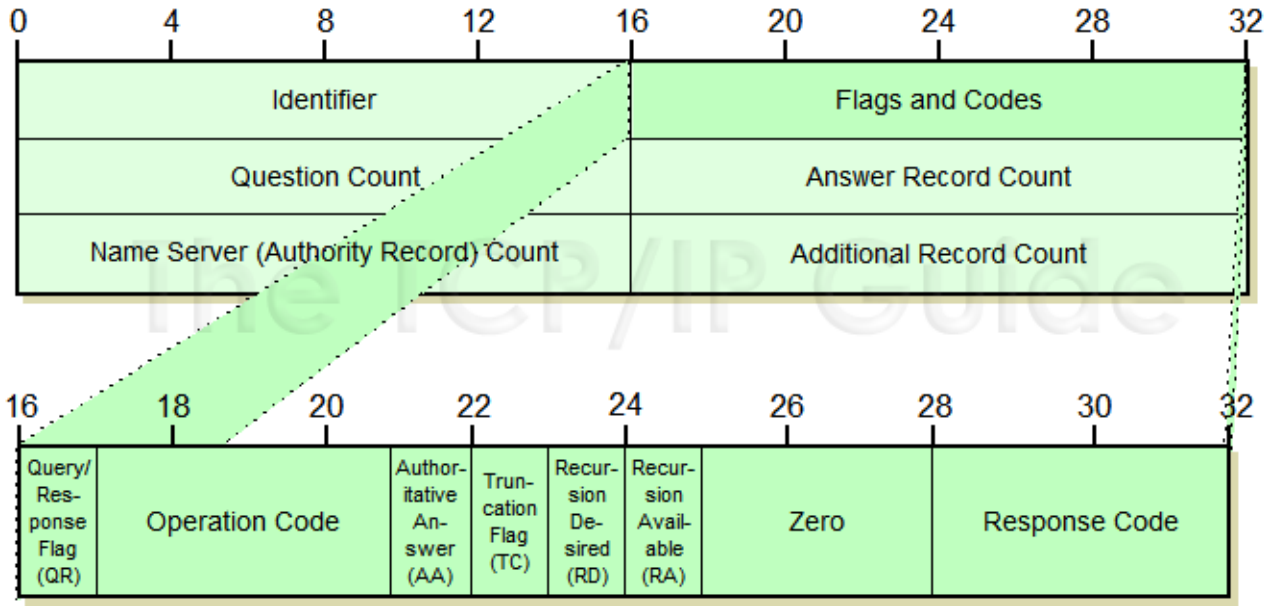
Type	Description
A	A host address. This RR type maps the name of a system to its IPv4 address. Some systems (e.g., routers) have multiple addresses, and there is a separate RR for each.
AAAA	Similar to A type, but for IPv6 addresses.
CNAME	Canonical name. Specifies an alias name for a host and maps this to the canonical (true) name.
HINFO	Host information. Designates the processor and operating system used by the host.
MINFO	Mailbox or mail list information. Maps a mailbox or mail list name to a host name.
MX	Mail exchange. Identifies the system(s) via which mail to the queried domain name should be relayed.
NS	Authoritative name server for this domain.
PTR	Domain name pointer. Points to another part of the domain name space.
SOA	Start of a zone of authority (which part of naming hierarchy is implemented). Includes parameters related to this zone.
SRV	For a given service, provides name of server or servers in domain that provide that service.
TXT	Arbitrary text. Provides a way to add text comments to the database.
WKS	Well-known services. May list the application services available at this host.

DNS messages (main idea)

DNS request contains name to be resolved
DNS reply contains IP address for the name in the request



Identifier: A 16-bit identification field generated by the device that creates the DNS query. It is copied by the server into the response, so it can be used by that device to match that query to the corresponding reply received from a DNS server.
Question Count: Specifies the number of questions in the *Question* section of the message.
Answer Record Count: Specifies the number of resource records in the *Answer* section of the message.
Authority Record Count: Specifies the number of resource records in the *Authority* section of the message
Additional Record Count: Specifies the number of resource records in the *Additional* section of the message.



www.tcpipguide.com/