Application Layer Protocols

SNMP - Simple Network Management Protocol

What is SNMP?

- SNMP is an application–layer protocol for exchanging management information between network devices.
- It is a part of Transmission Control Protocol⁄Internet Protocol (TCP⁄IP) protocol suite.

Concept:

- SNMP uses the concept of manager and agent.
- That is, a manager, usually a host, controls and monitors a set of agents, usually routers or servers.

Managers & Agents:

- A management station, called a manager, is a host that runs the SNMP client program.
- A managed station, called an agent, is a router (or a host) that runs the SNMP server program.
- Management is achieved through simple interaction between a manager and an agent.

Managers & Agents:

- The agent keeps performance information in a database.
- The manager has access to the values in the database.
- For example, a router can store in appropriate variables the number of packets received and forwarded.
- The manager can fetch and compare the values of these two variables to see if the router is congested or not.

Managers & Agents:

- The manager can also make the router perform certain actions.
- For example, a router periodically checks the value of a reboot counter to see when it should reboo titself.
- It reboots itself, for example, if the value of the counter is 0.
- The manager can use this feature to reboot the agent remotely at any time. It simply sends a packet to force a 0 value in the counter.

Managers & Agents:

- Agents can also contribute to the management process.
- The server program running on the agent can check the environment and, if it notices something unusual, it can send a warning message (called a trap) to the manager.

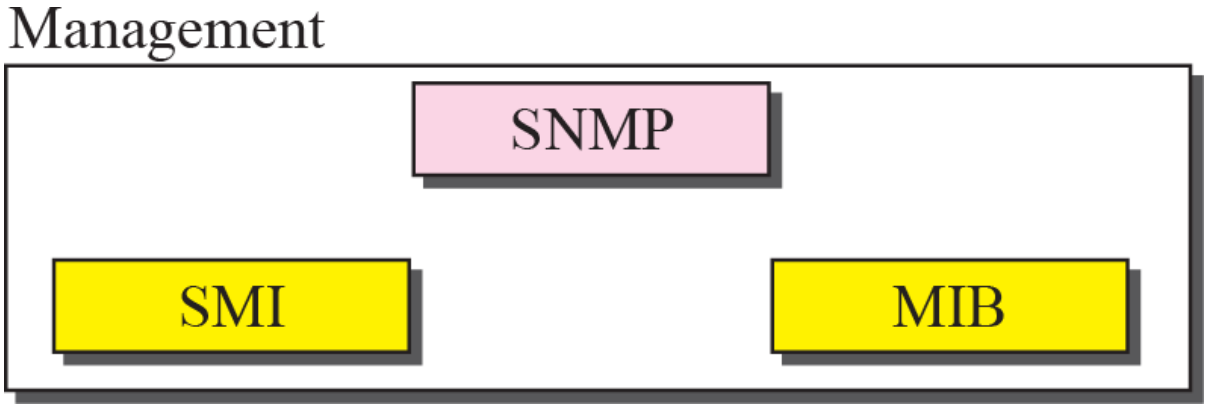
Managers & Agents:

- In other words, management with SNMP is based on three basic ideas:

1. A manager checks an agent by requesting information that reflects the behavior of the agent.
2. A manager forces an agent to perform a task by resetting values in the agent database.
3. An agent contributes to the management process by warning the manager of an unusual situation.

Management Components:

- To do management tasks, SNMP uses two other protocols:
  - Structure of Management Information (SMI)
  - Management Information Base (MIB).
- In other words, management on the Internet is done through the cooperation of three protocols: SNMP, SMI, and MIB



Role of SMI

- SMI defines the general rules for naming objects, defining object types (including range and length), and showing how to encode objects and values.

Role of MIB

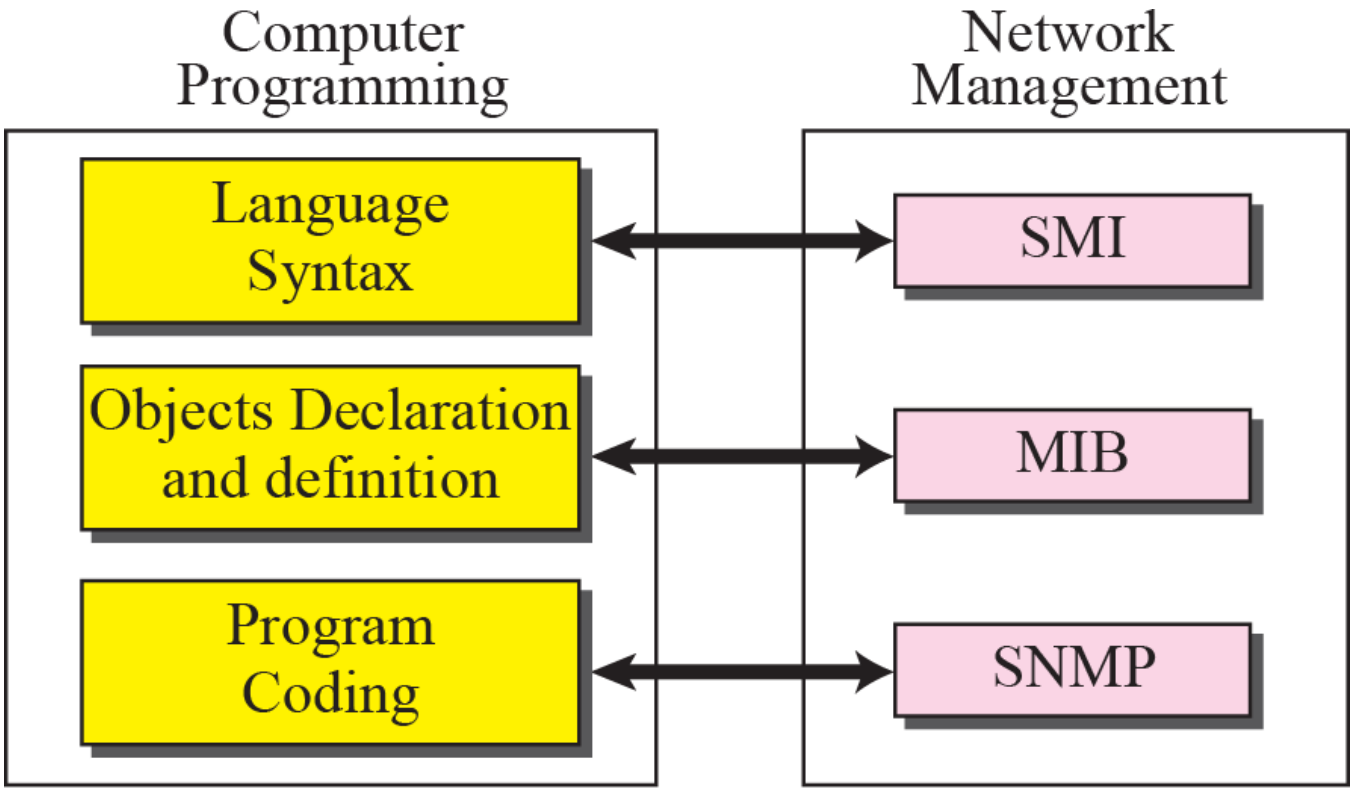
- MIB creates a collection of named objects, their types, and their relationships to each other in an entity to be managed.

Role of SNMP

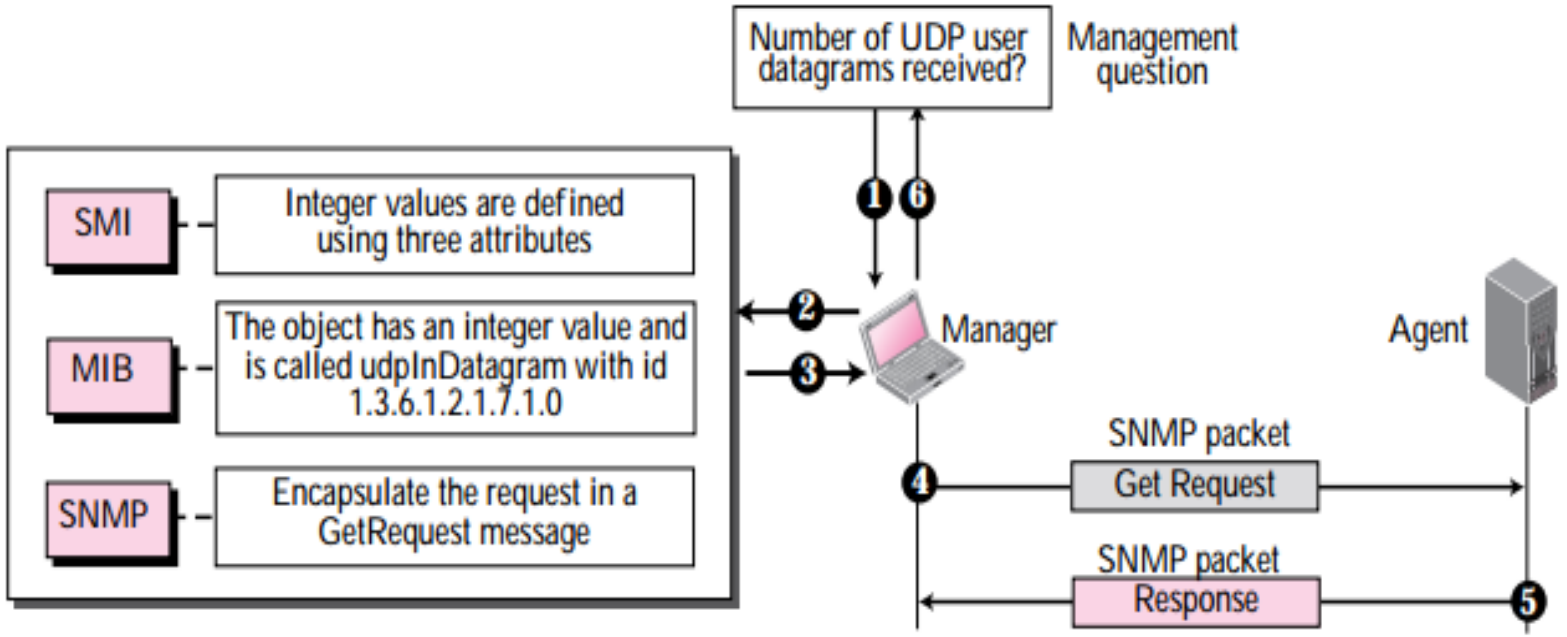
- SNMP defines the format of packets exchanged between a manager and an agent.

- It reads and changes the status of objects (values of variables) in SNMP packets.

Analogy


Computer Programming
Network Management
Language Syntax
Objects Declaration and definition
Program Coding
SMI
MIB
SNMP

Management Overview



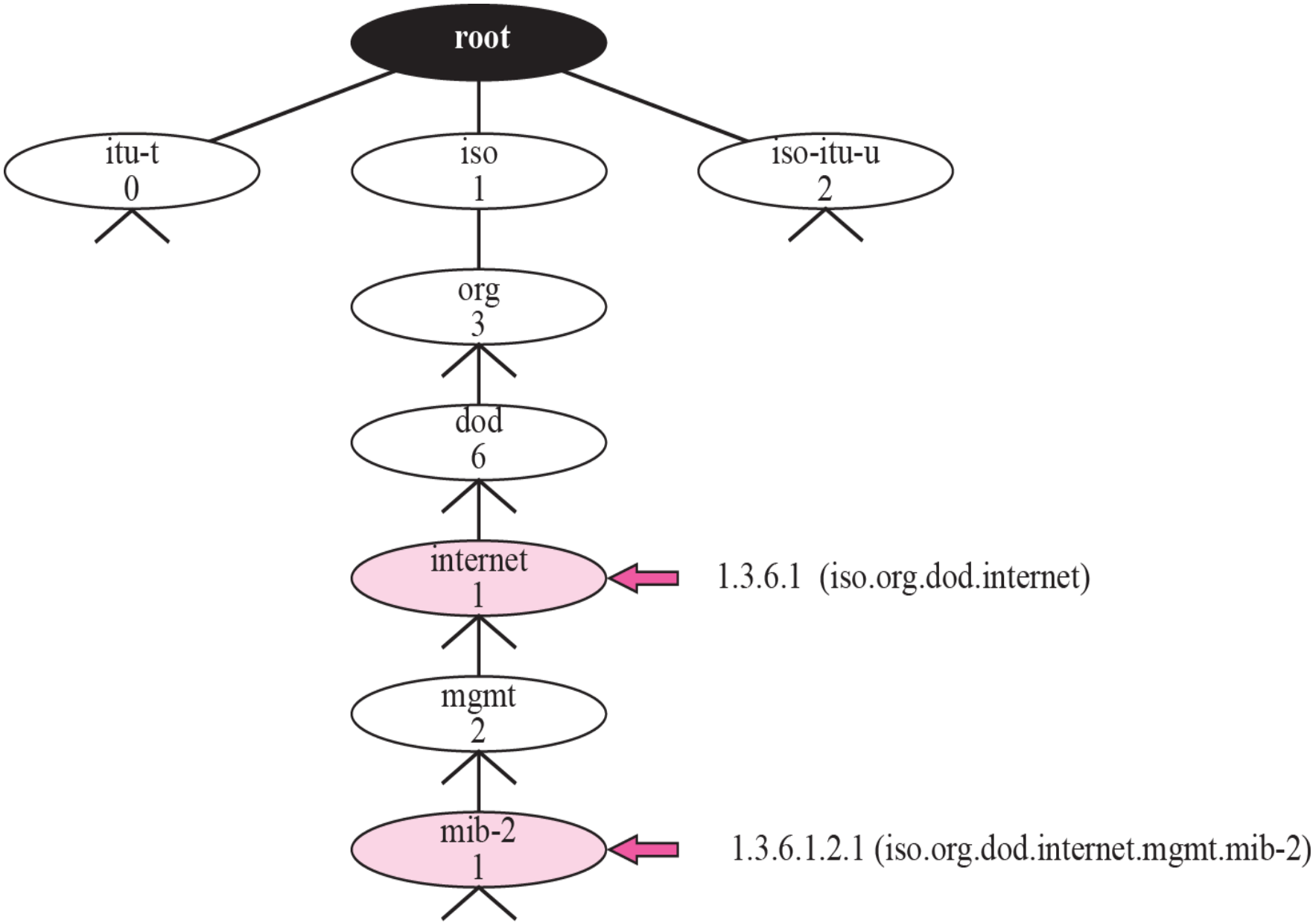
Functions of SMI :

1. To name objects.
2. To define the type of data that can be stored in an object.
3. To show how to encode data for transmission over the network.

- SMI is a guideline for SNMP.

Object Identifier

- All objects managed by SNMP are given an object identifier.
- The object identifier always starts with 1.3.6.1.2.1.


root
itu-t
0
iso
1
iso-itu-u
2
org
3
dod
6
internet
1
1.3.6.1 (iso.org.dod.internet)
mgmt
2
mib-2
1
1.3.6.1.2.1 (iso.org.dod.internet.mgmt.mib-2)

Data types

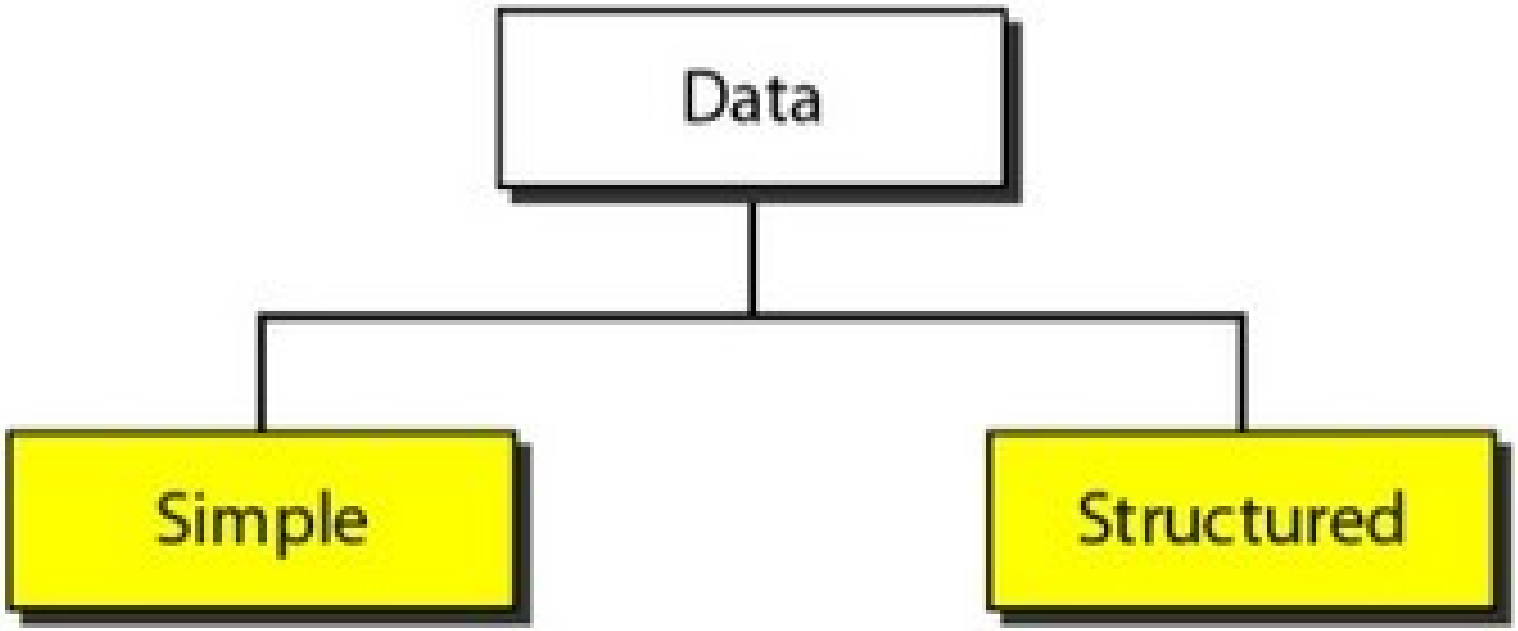
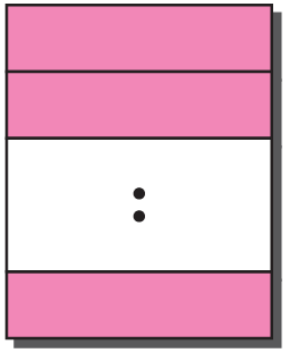


Table 24.1 *Data Types*

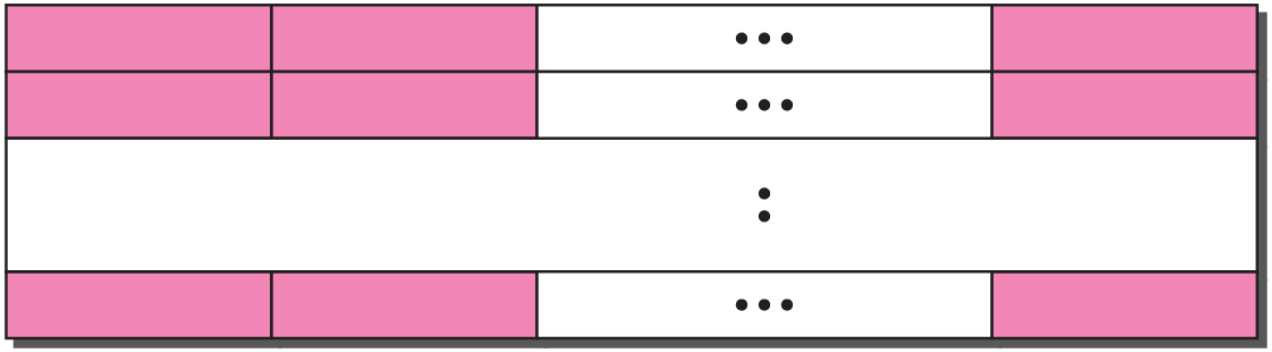
Type	*Size*	*Description*
INTEGER	4 bytes	An integer with a value between -2^{31} and $2^{31}-1$
Integer32	4 bytes	Same as INTEGER
Unsigned32	4 bytes	Unsigned with a value between 0 and $2^{32}-1$
OCTET STRING	Variable	Byte-string up to 65,535 bytes long
OBJECT IDENTIFIER	Variable	An object identifier
IPAddress	4 bytes	An IP address made of four integers
Counter32	4 bytes	An integer whose value can be incremented from zero to 2^{32}; when it reaches its maximum value it wraps back to zero
Counter64	8 bytes	64-bit counter
Gauge32	4 bytes	Same as Counter32, but when it reaches its maximum value, it does not wrap; it remains there until it is reset
TimeTicks	4 bytes	A counting value that records time in 1/100ths of a second
BITS		A string of bits
Opaque	Variable	Uninterpreted string

a. Simple variable

c. Sequence



b. Sequence of
(simple variables)

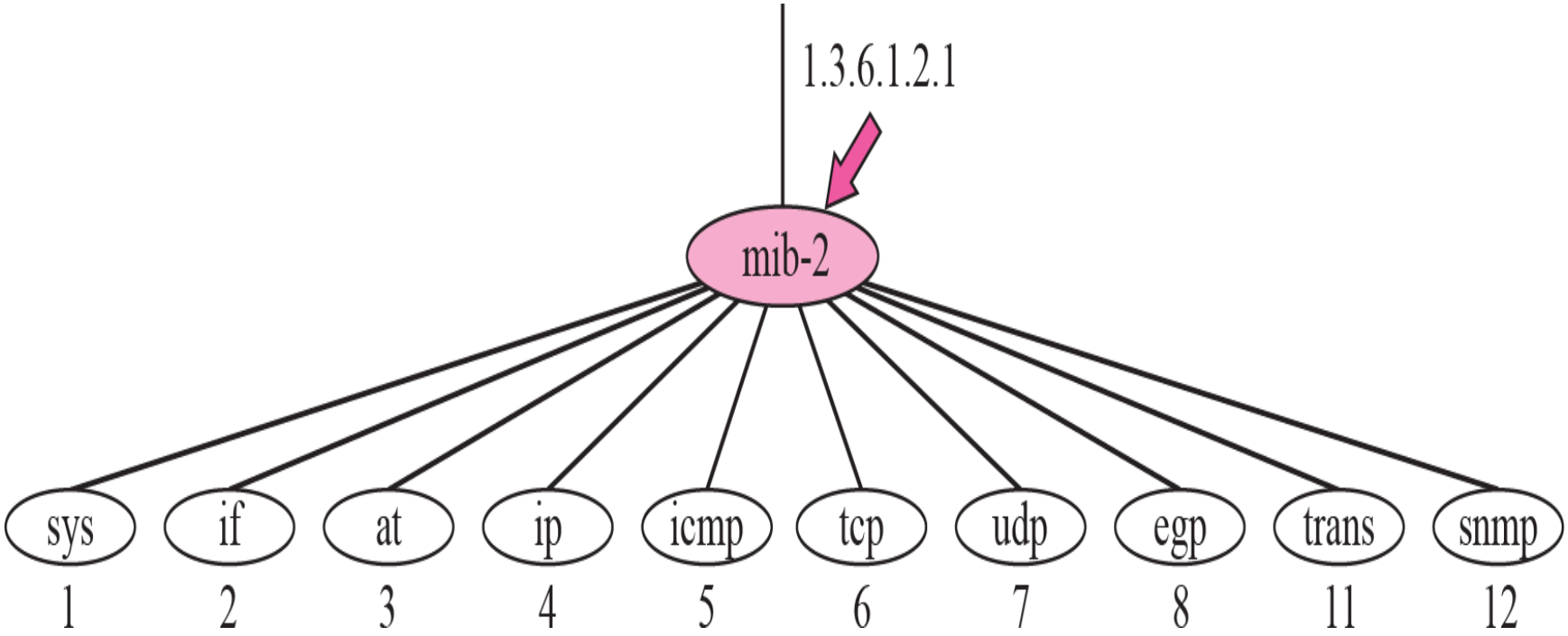


d. Sequence of
(sequences)

MIB

- The Management Information Base, version 2 (MIB2) is the second component used in network management.
- Each agent has its own MIB2, which is a collection of all the objects that the manager can manage.
- The objects in MIB2 are categorized under 10 different groups: system, interface, address translation, ip, icmp, tcp, udp, egp, transmission, and snmp.
- These groups are under the mib-2 object in the object identifier tree.

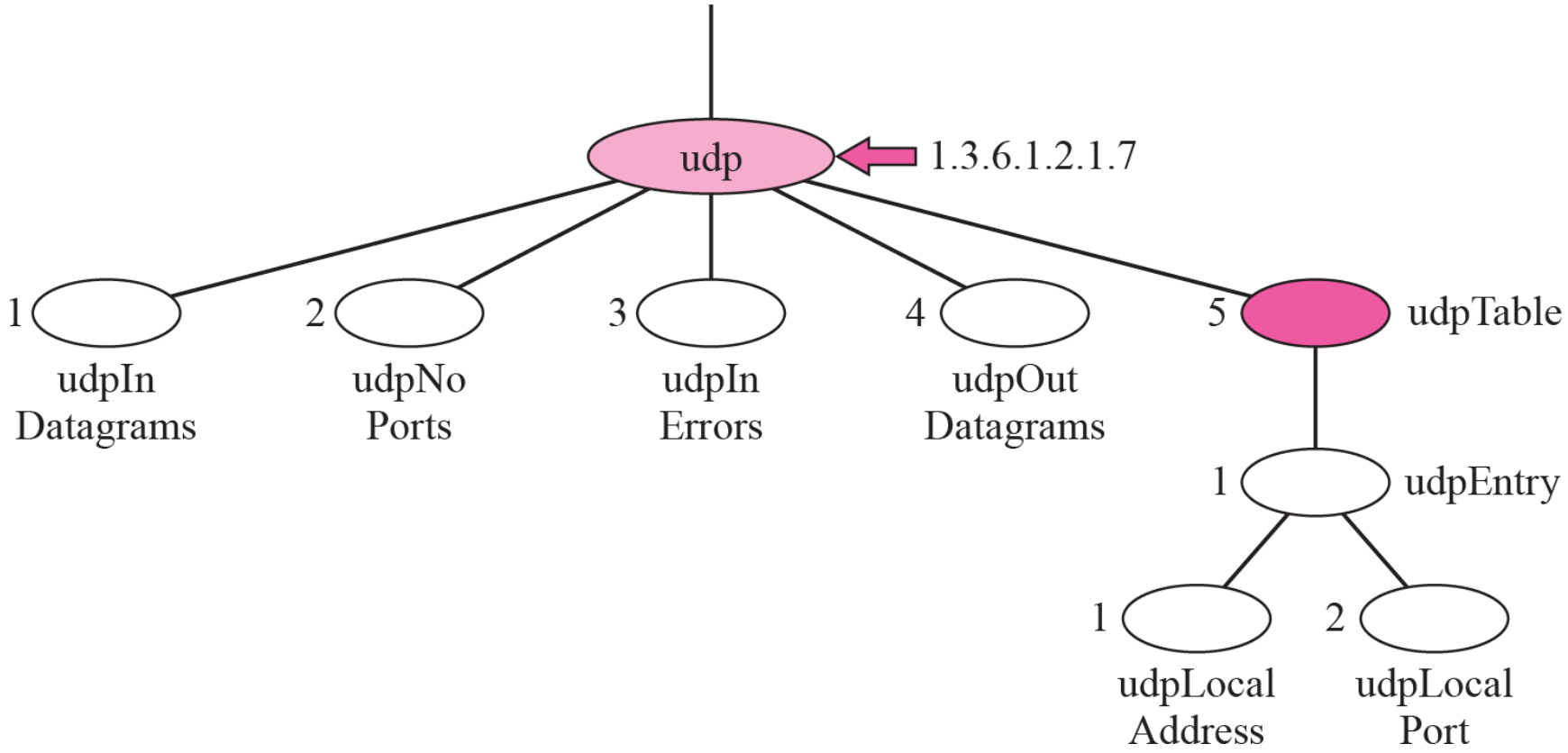
MIB


1.3.6.1.2.1
mib-2
sys
if
at
ip
icmp
tcp
udp
egp
trans
snmp
1
2
3
4
5
6
7
8
11
12

MIB

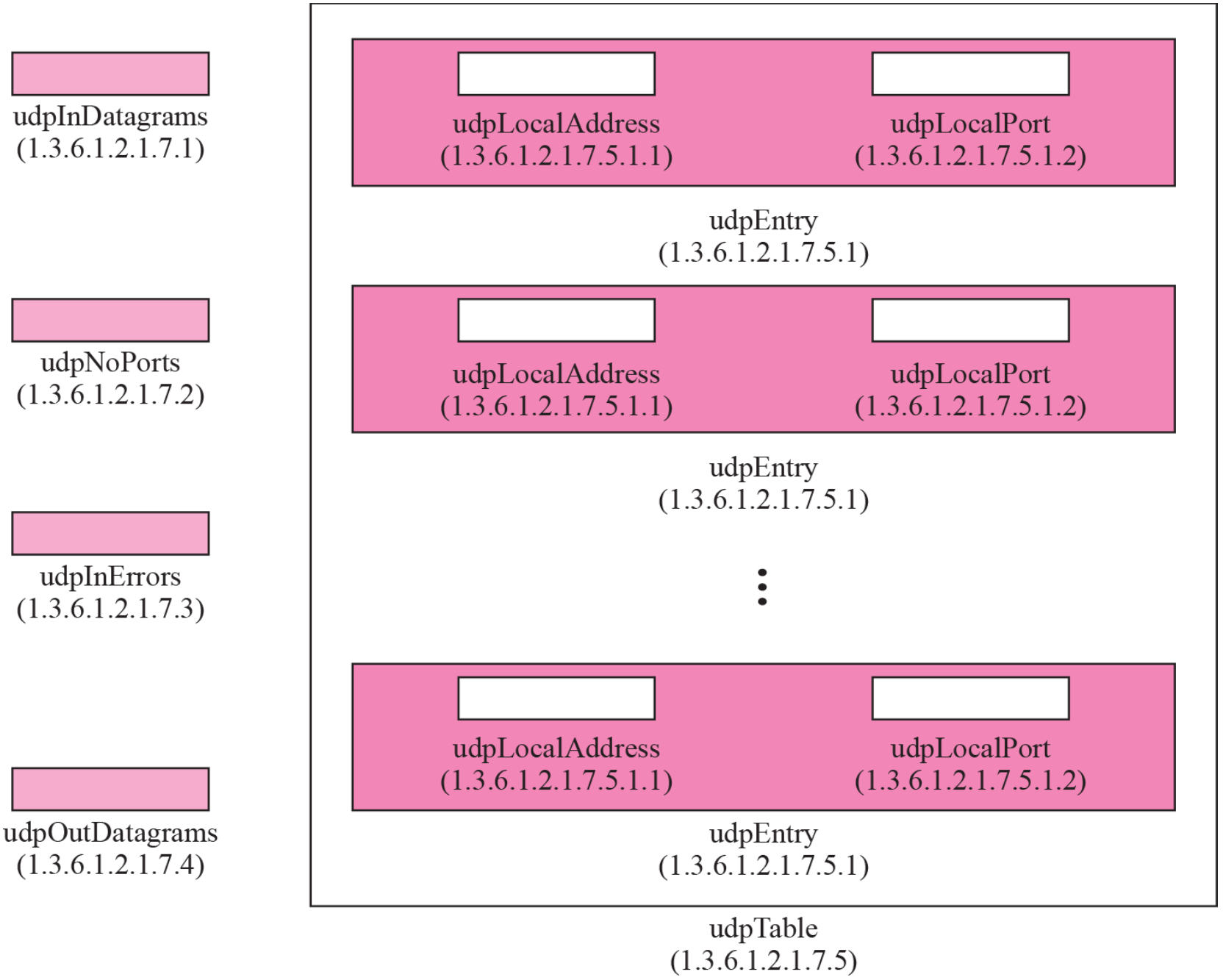
- **sys** This object (*system*) defines general information about the node (system), such as the name, location, and lifetime.
- **if** This object (*interface*) defines information about all of the interfaces of the node including interface number, physical address, and IP address.
- **at** This object (*address translation*) defines the information about the ARP table.
- **ip** This object defines information related to IP, such as the routing table and the IP address.
- **icmp** This object defines information related to ICMP, such as the number of packets sent and received and total errors created.
- **tcp** This object defines general information related to TCP, such as the connection table, time-out value, number of ports, and number of packets sent and received.
- **udp** This object defines general information related to UDP, such as the number of ports and number of packets sent and received.
- **snmp** This object defines general information related to SNMP itself.

An Example of Accessing MIB Variables: UDP group

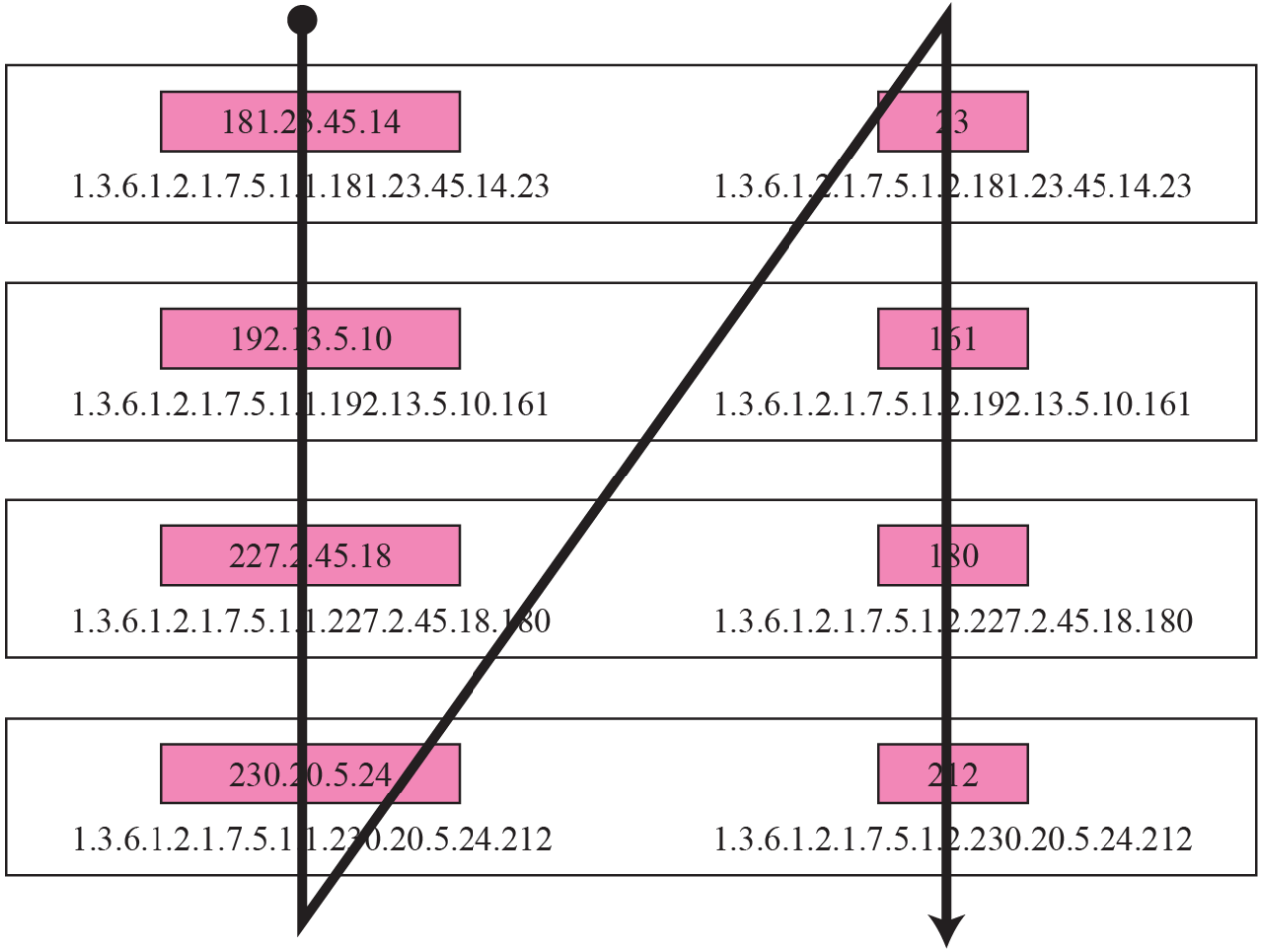


Accessing MIB variables

- To access any simple variables we use the id of the group (1.3.6.1.2.1.7) followed by id of the variable.
  - udpInDatagrams = 1.3.6.1.2.1.7.1,
  - udpNoPorts = 1.3.6.1.2.1.7.2 etc.
- But these object identifiers define variables and not the instance (contents).
- To show an instance of the above variables, we use the following:
  - udpInDatagrams.O = 1.3.6.1.2.1.7.1.0,
  - udpNoPorts.O=1.3.6.1.2.1.7.2.0


udpInDatagrams
(1.3.6.1.2.1.7.1)
udpNoPorts
(1.3.6.1.2.1.7.2)
udpInErrors
(1.3.6.1.2.1.7.3)
udpOutDatagrams
(1.3.6.1.2.1.7.4)
udpLocalAddress
(1.3.6.1.2.1.7.5.1.1)
udpLocalPort
(1.3.6.1.2.1.7.5.1.2)
udpEntry
(1.3.6.1.2.1.7.5.1)
udpLocalAddress
(1.3.6.1.2.1.7.5.1.1)
udpLocalPort
(1.3.6.1.2.1.7.5.1.2)
udpEntry
(1.3.6.1.2.1.7.5.1)
udpLocalAddress
(1.3.6.1.2.1.7.5.1.1)
udpLocalPort
(1.3.6.1.2.1.7.5.1.2)
udpEntry
(1.3.6.1.2.1.7.5.1)
udpTable
(1.3.6.1.2.1.7.5)

Lexicographic ordering



HTTP – Hyper Text Transfer Protocol

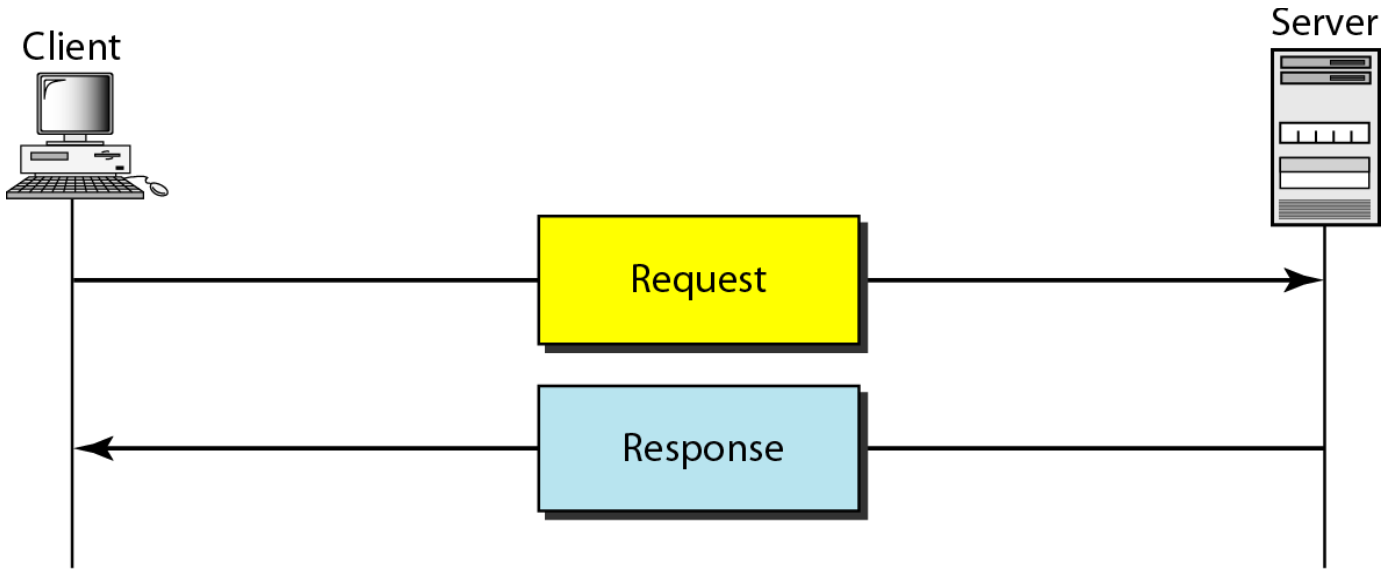
HTTP:

- The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the World Wide Web.

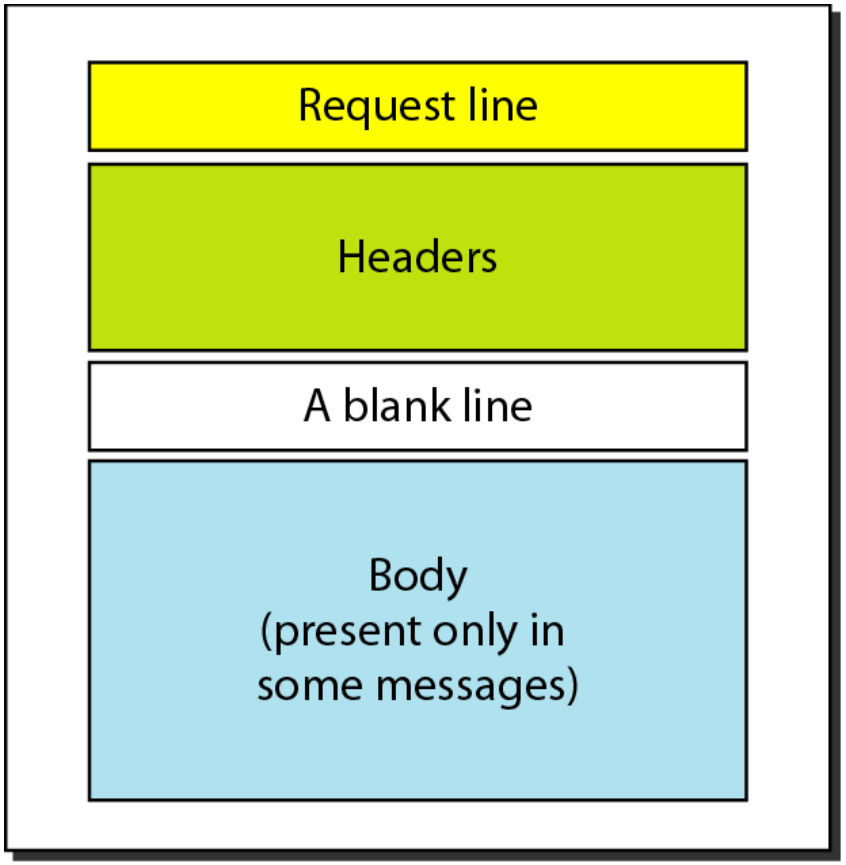
Basics of HTTP:

- HTTP uses the services of TCP on well-known port 80.

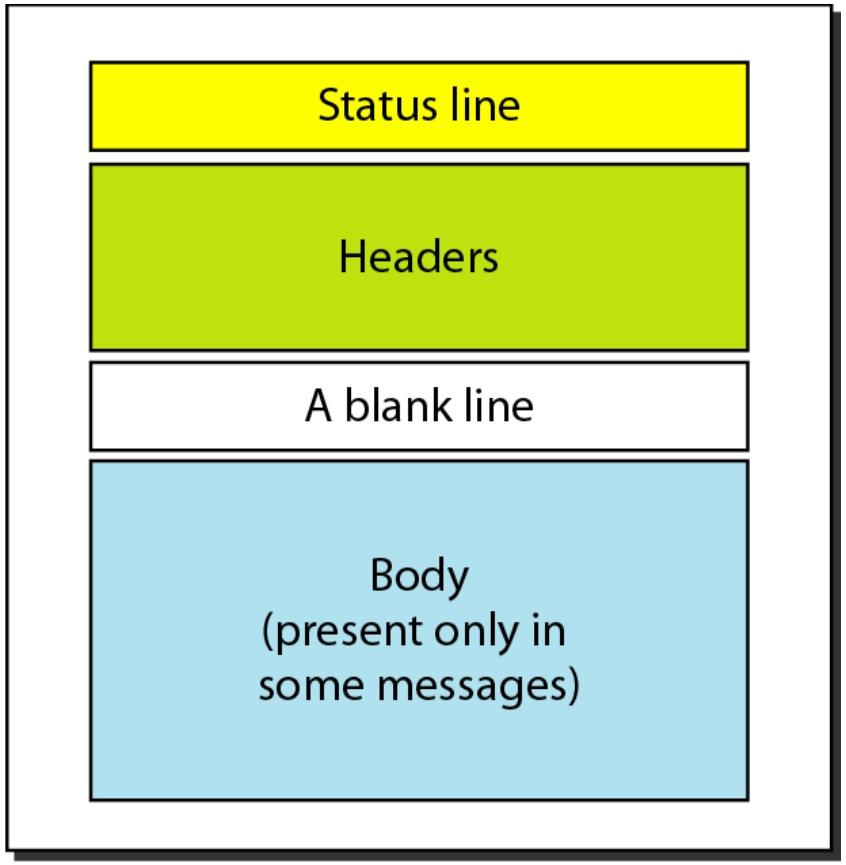
HTTP transaction:



Request and response messages

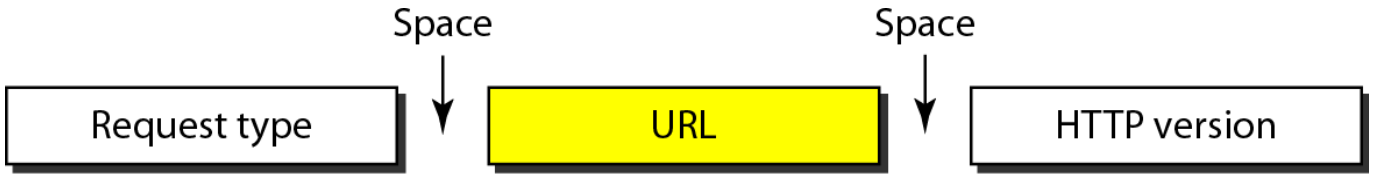


Request message



Response message

Request line: Request message

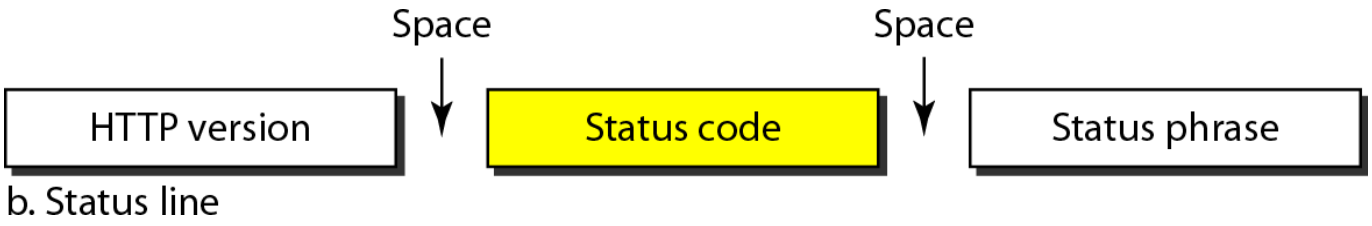


a. Request line

Request type:

Method	*Action*
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
POST	Sends some information from the client to the server
PUT	Sends a document from the server to the client

Status line: Response message



b. Status line

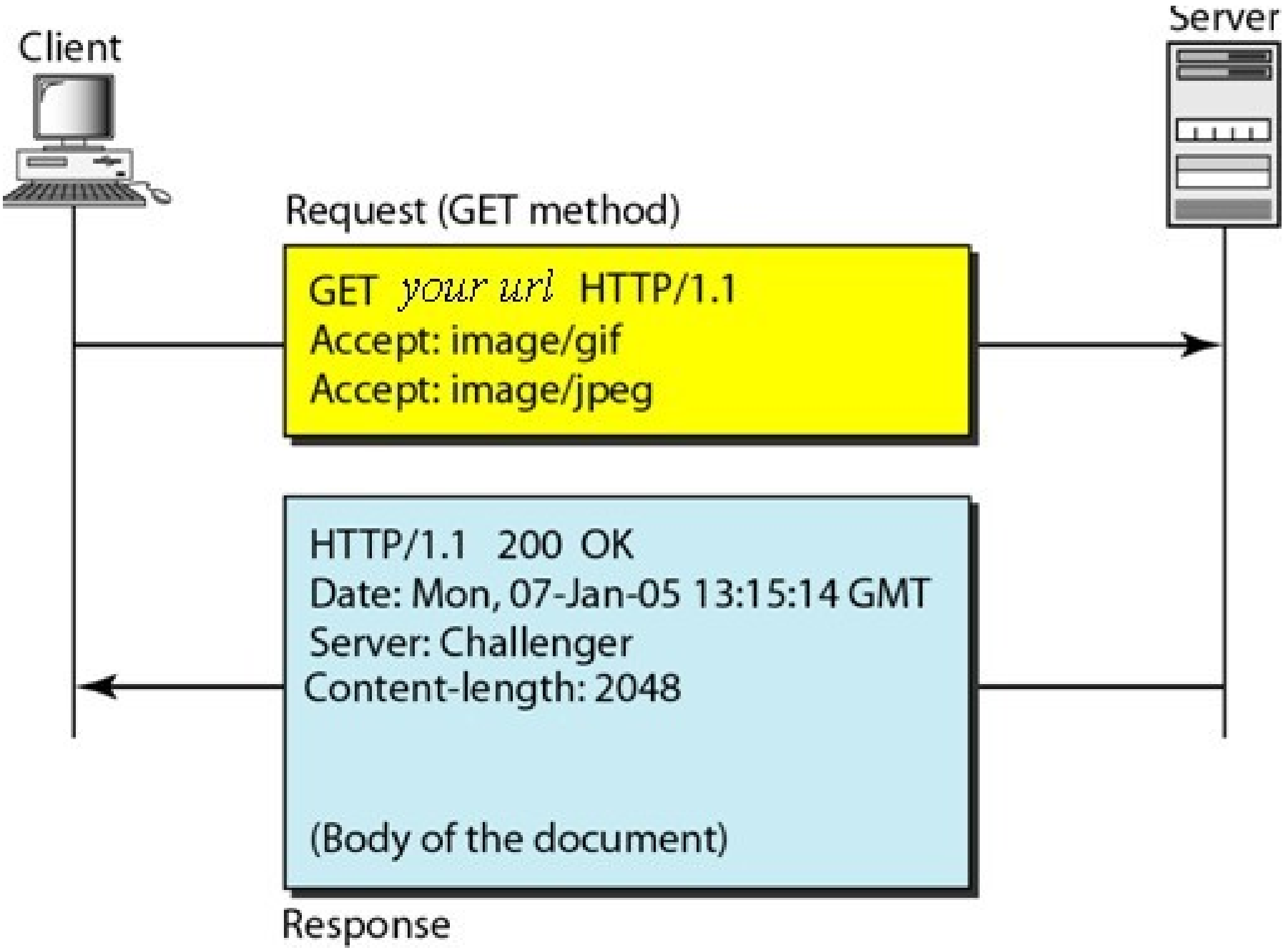
Code	*Phrase*	*Description*
200	OK	The request is successful.
400	Bad request	There is a syntax error in the request.
401	Unauthorized	The request lacks proper authorization.
404	Not found	The document is not found.
500	Internal server error	There is an error, such as a crash, at the server site.

Headers:

Header categories	Type of Message	Example
General header: General information	Response/ Request	Date, upgrade etc
Request header	Request	Accept, user agent, accept language, accept encoding, if match etc
Response header	Response	Age (age of the document), server(server name, version number) etc
Entity header: Information about the body of document	Request/Response (The message with body)	Content length, content language, last modified etc.

Body

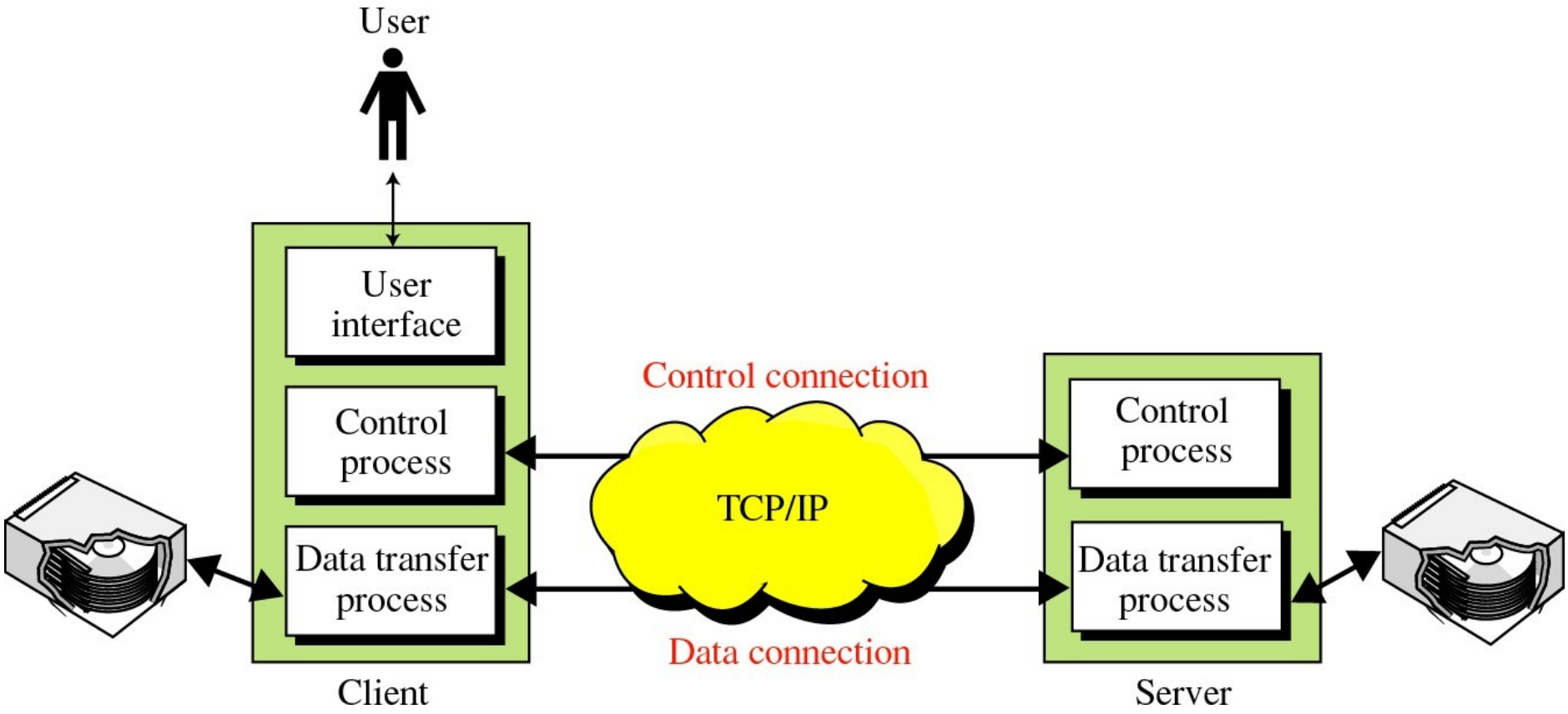
- The body can be present in request or response message.
- Usually, it contains the document to be sent or received.


Client
Server
Request (GET method)
GET your url HTTP/1.1
Accept: image/gif
Accept: image/jpeg
HTTP/1.1 200 OK
Date: Mon, 07-Jan-05 13:15:14 GMT
Server: Challenger
Content-length: 2048
(Body of the document)
Response

FTP-File transfer protocol

Basics of FTP:

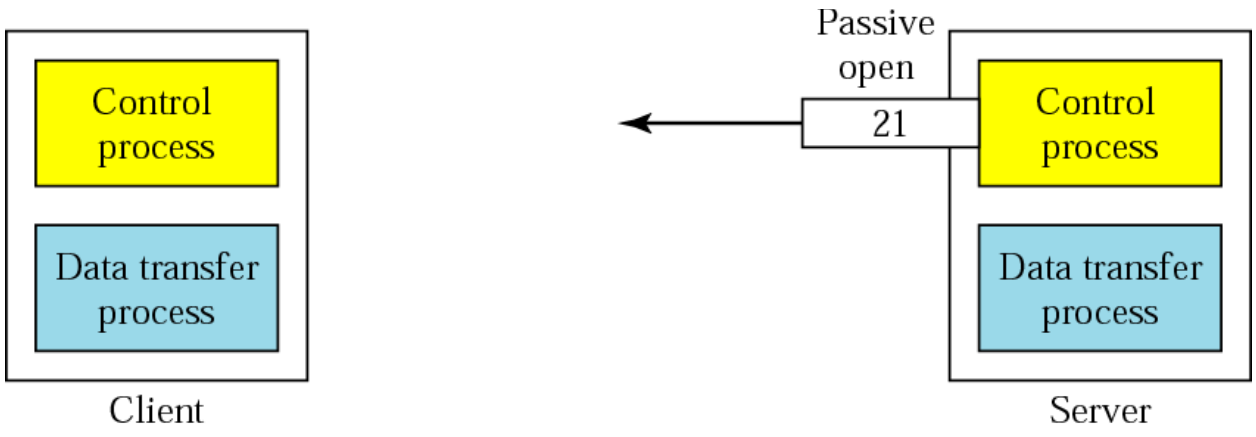
- A standard mechanism to copy file from one host to another.
- Uses services of TCP.
- Needs two TCP connections:
  - one for control : well-known port 21
  - one for data transfers: well-known port 20


User
User interface
Control process
Data transfer process
Client
Control connection
TCP/IP
Data connection
Control process
Data transfer process
Server

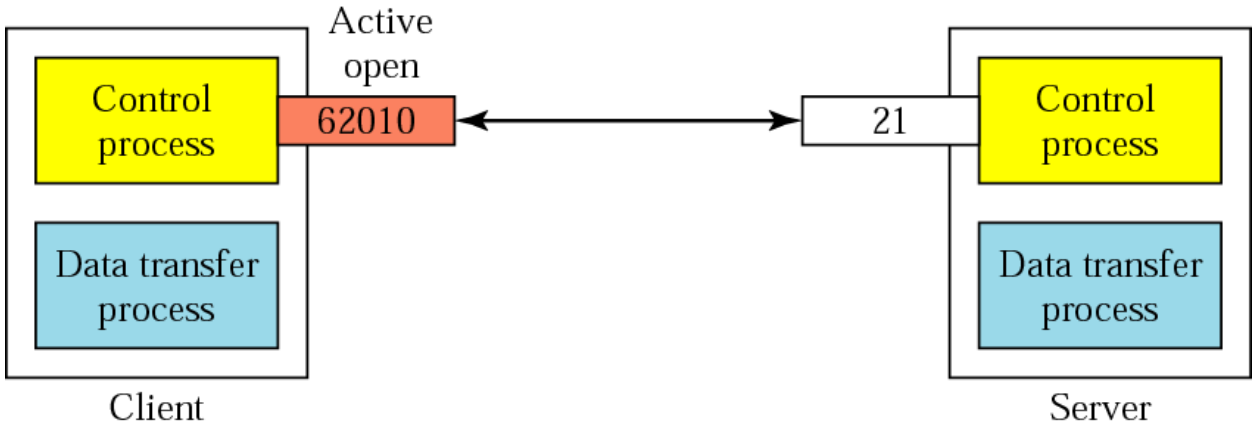
Basics of connections:

- The control connection remains connected during the entire interactive FTP session
- The data connection is opened and closed for each file transferred

Connections: The control connection



a. Passive open by server



b. Active open by client

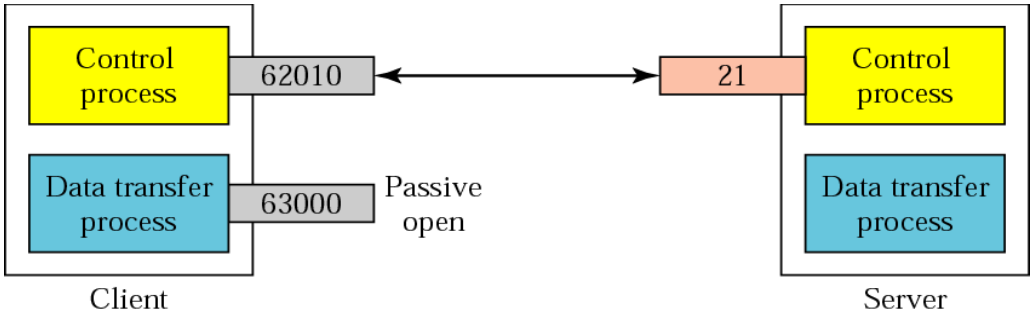
Client picks a temporary port number to open connection.

Communication over Data connection:

- File transfer occurs over the data connection under the control of the commands over the control connection.
- FTP does one of three operations:
  - A file is to be copied from the server to client.
  - A file is to be copied form the client to server
  - A list of directory/files from server to client.

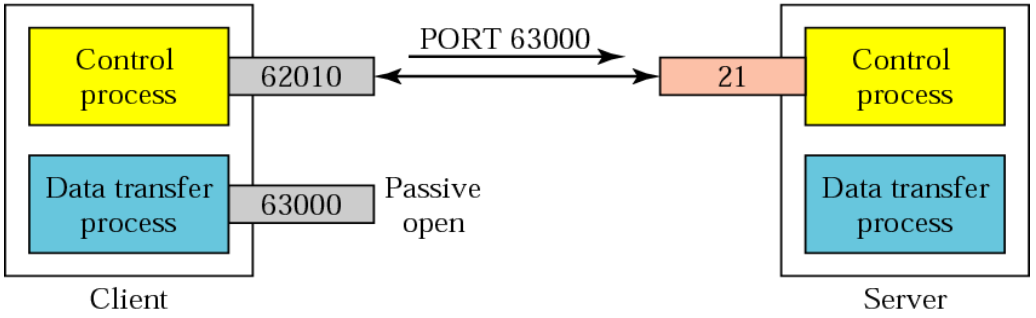
Creating the data connection

For data connection, client issues the passive open first.



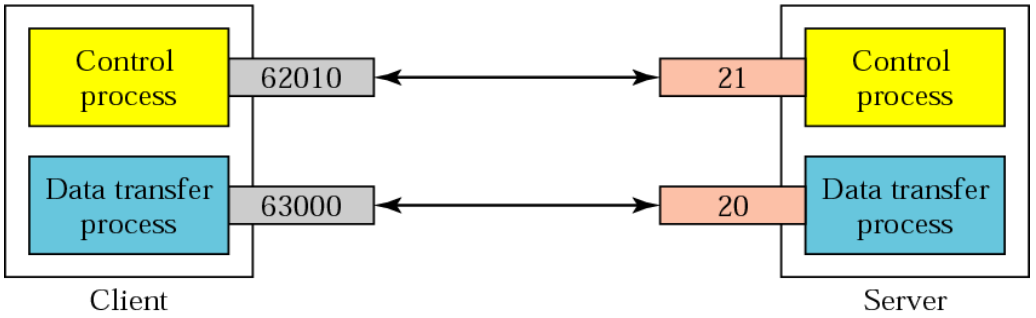
a. Passive open by client

Client sends the port number to server using PORT command.



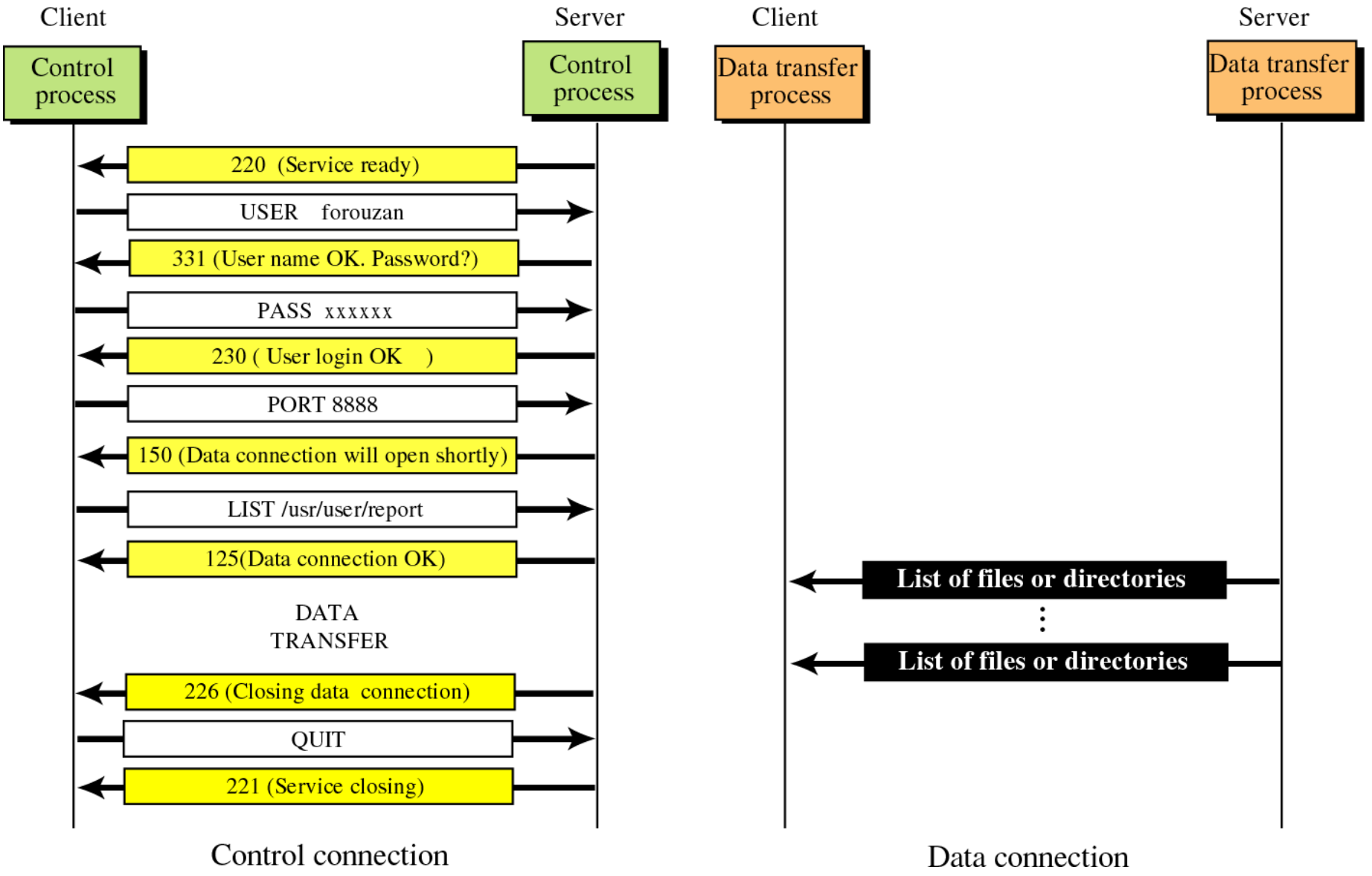
b. Sending ephemeral port number to server

Server receives the port number and issues an active open using port number 20.



c. Active open by server

Example


Client
Server
Client
Server
Control process
Control process
Data transfer process
Data transfer process
220 (Service ready)
USER forouzan
331 (User name OK. Password?)
PASS xxxxxx
230 (User login OK)
PORT 8888
150 (Data connection will open shortly)
LIST /usr/user/report
125(Data connection OK)
DATA TRANSFER
List of files or directories
List of files or directories
226 (Closing data connection)
QUIT
221 (Service closing)
Control connection
Data connection

Example 1

Figure shows an example of using FTP for retrieving a list of items in a directory.

1. After the control connection to port 21 is created, the FTP server sends the 220 (service ready) response on the control connection.
2. The client sends the USER command.
3. The server responds with 331 (user name is OK, password is required).
4. The client sends the PASS command.
5. The server responds with 230 (user login is OK)

6. The client issues a passive open on an ephemeral port for the data connection and sends the PORT command (over the control connection) to give this port number to the server.
7. The server does not open the connection at this time, but it prepares itself for issuing an active open on the data connection between port 20 (server side) and the ephemeral port received from the client. It sends response 150 (data connection will open shortly).
8. The client sends the LIST message.
9. Now the server responds with 125 and opens the data connection.

10. The server then sends the list of the files or directories (as a file) on the data connection. When the whole list (file) is sent, the server responds with 226 (closing data connection) over the control connection.
11. The client now has two choices. It can use the QUIT command to request the closing of the control connection or it can send another command to start another activity (and eventually open another data connection). In our example, the client sends a QUIT command.
12. After receiving the QUIT command, the server responds with 221 (service closing) and then closes the control connection.

Example 2

- The following shows an actual FTP session that parallels Example 1.
- The colored lines show the responses from the server control connection;
- The black lines show the commands sent by the client.
- The lines in white with black background shows data transfer.

```
$ ftp voyager.deanza.fhda.edu  Connected to voyager.deanza.fhda.edu.
220 (vsFTPd 1.2.1)
530 Please login with USER and PASS.
Name (voyager.deanza.fhda.edu:forouzan): forouzan
331 Please specify the password.
```

Password:
230 Login successful. Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls reports
227 Entering Passive Mode (153,18,17,11,238,169)
150 Here comes the directory listing.

drwxr-xr-x 2 3027 411 4096 Sep 24 2002 business
drwxr-xr-x 2 3027 411 4096 Sep 24 2002 personal
drwxr-xr-x 2 3027 411 4096 Sep 24 2002 school

226 Directory send OK. ftp> quit
221 Goodbye.

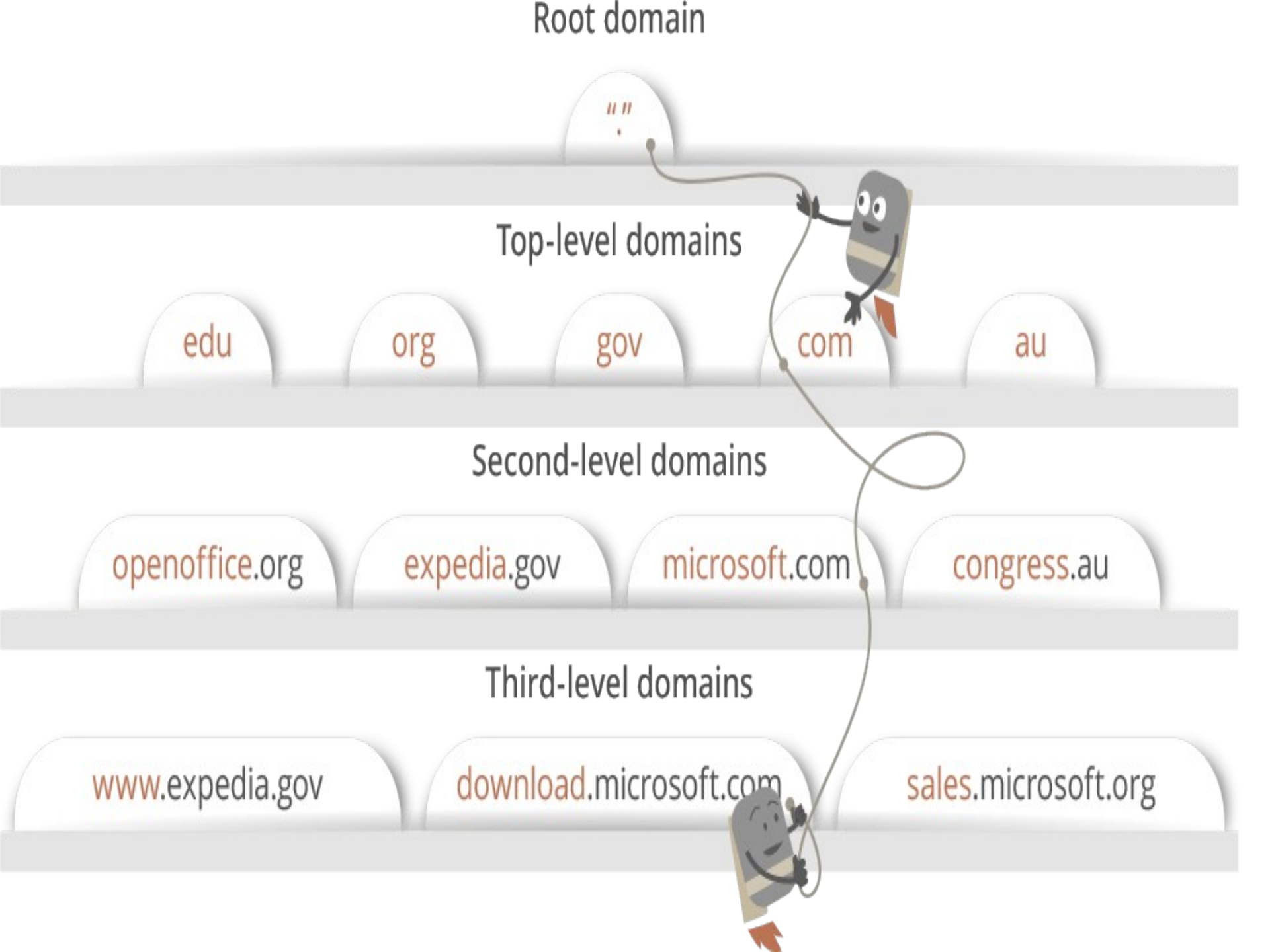
DNS - Domain Name System

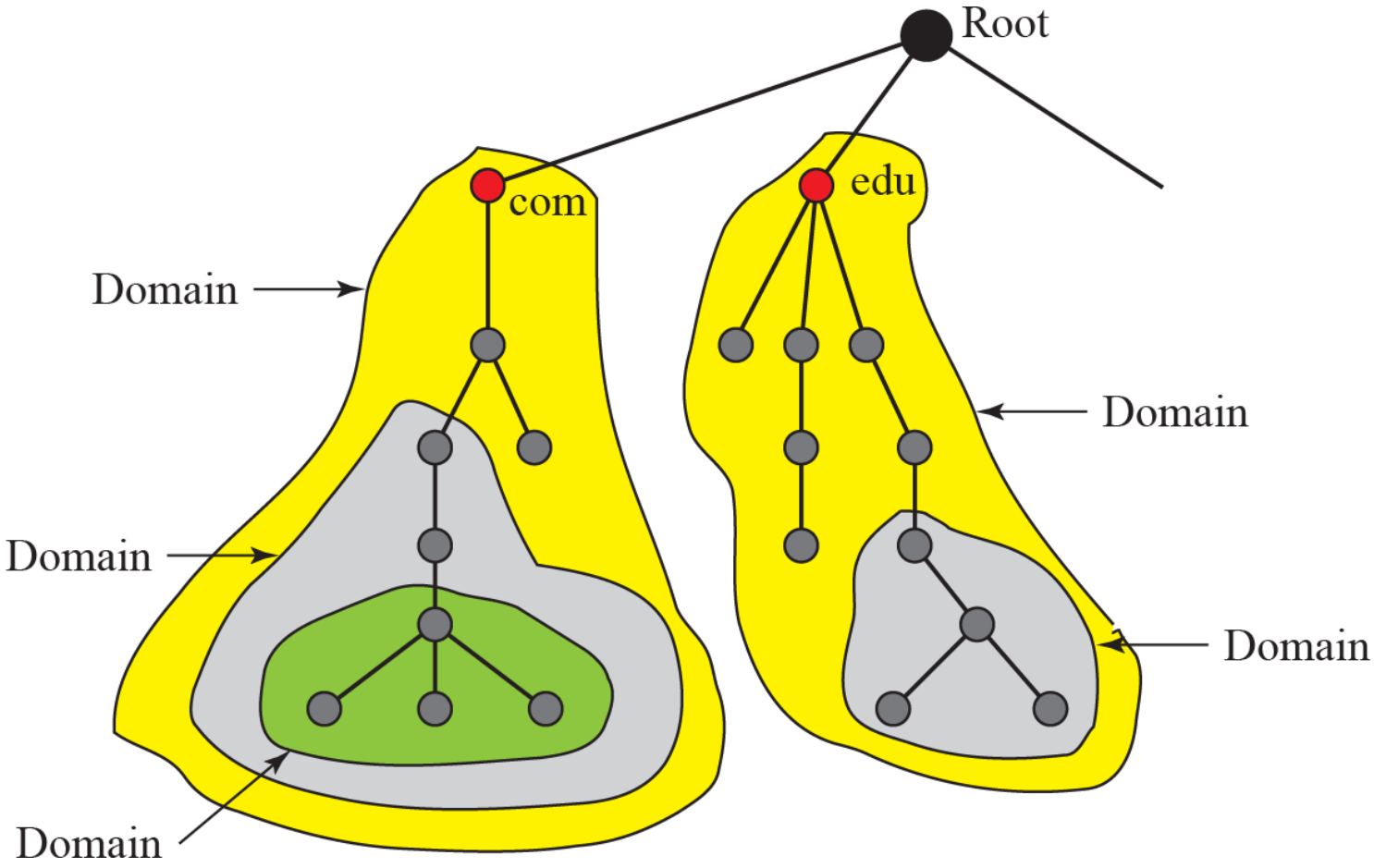
Need for DNS

- To identify an entity, TCP/IP protocols use the IP address.
- 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.

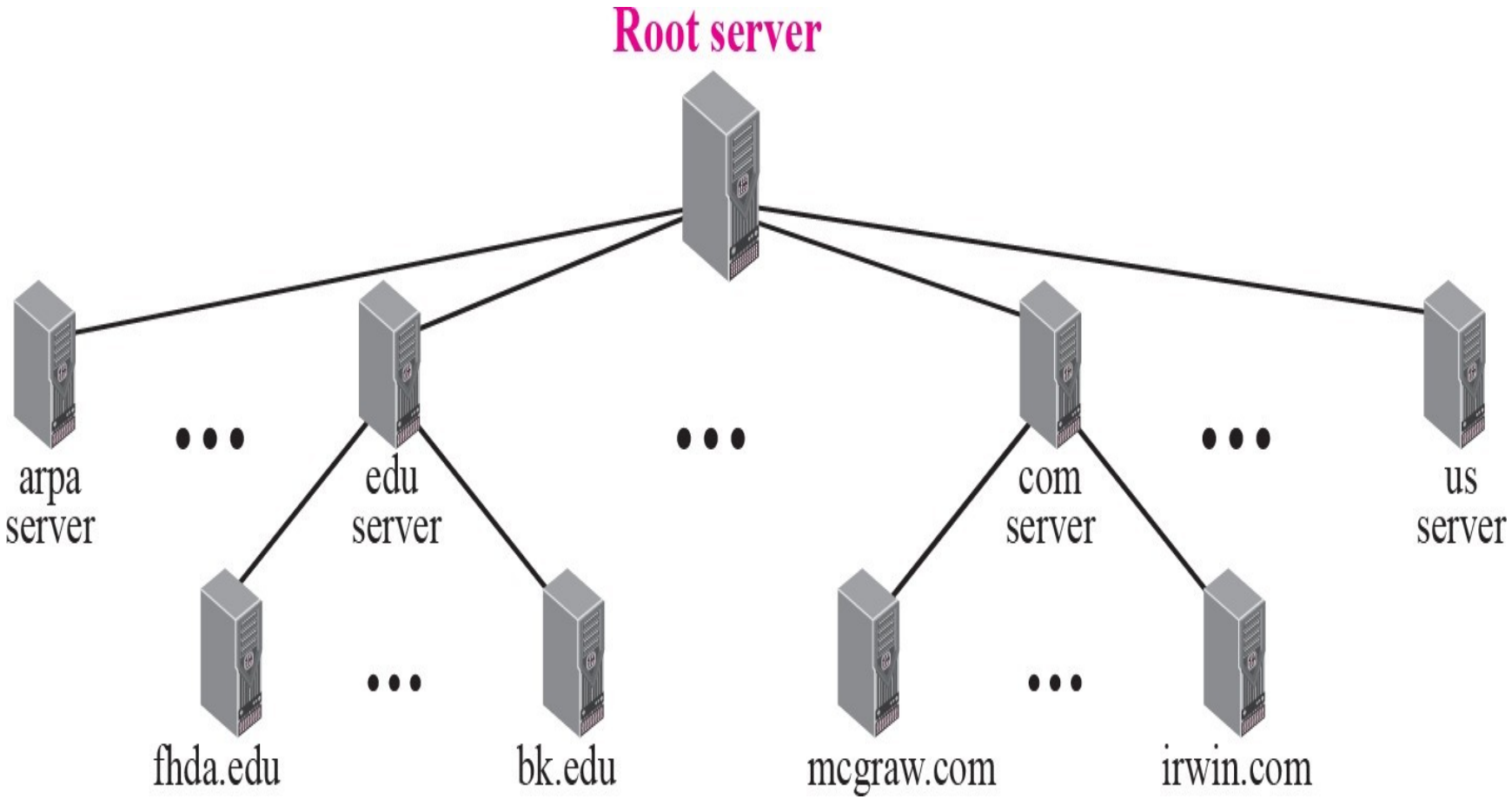
Hierarchical domain name space

- In hierarchical domain name space each name is made of several parts.
- First part can define the **nature** of the organization.
- Second part candefine the **name** of the organization.
- Third can define the **department** in the organization.
- So on.

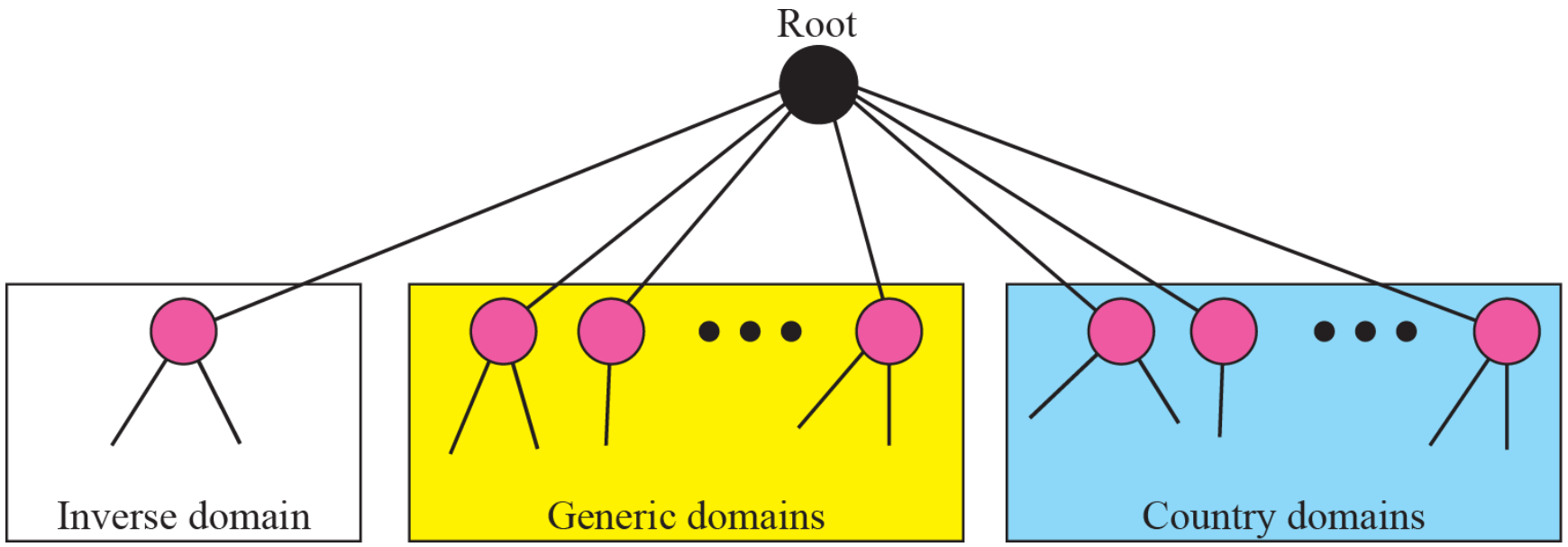

Root domain
""
Top-level domains
edu
org
gov
com
au
Second-level domains
openoffice.org
expedia.gov
microsoft.com
congress.au
Third-level domains
www.expedia.gov
download.microsoft.com
sales.microsoft.org


Root
com
edu
Domain
Domain
Domain
Domain
Domain

Distributed Name servers:



DNS in Internet



In internet the domain name space is divided in three different sections.

Generic domains:

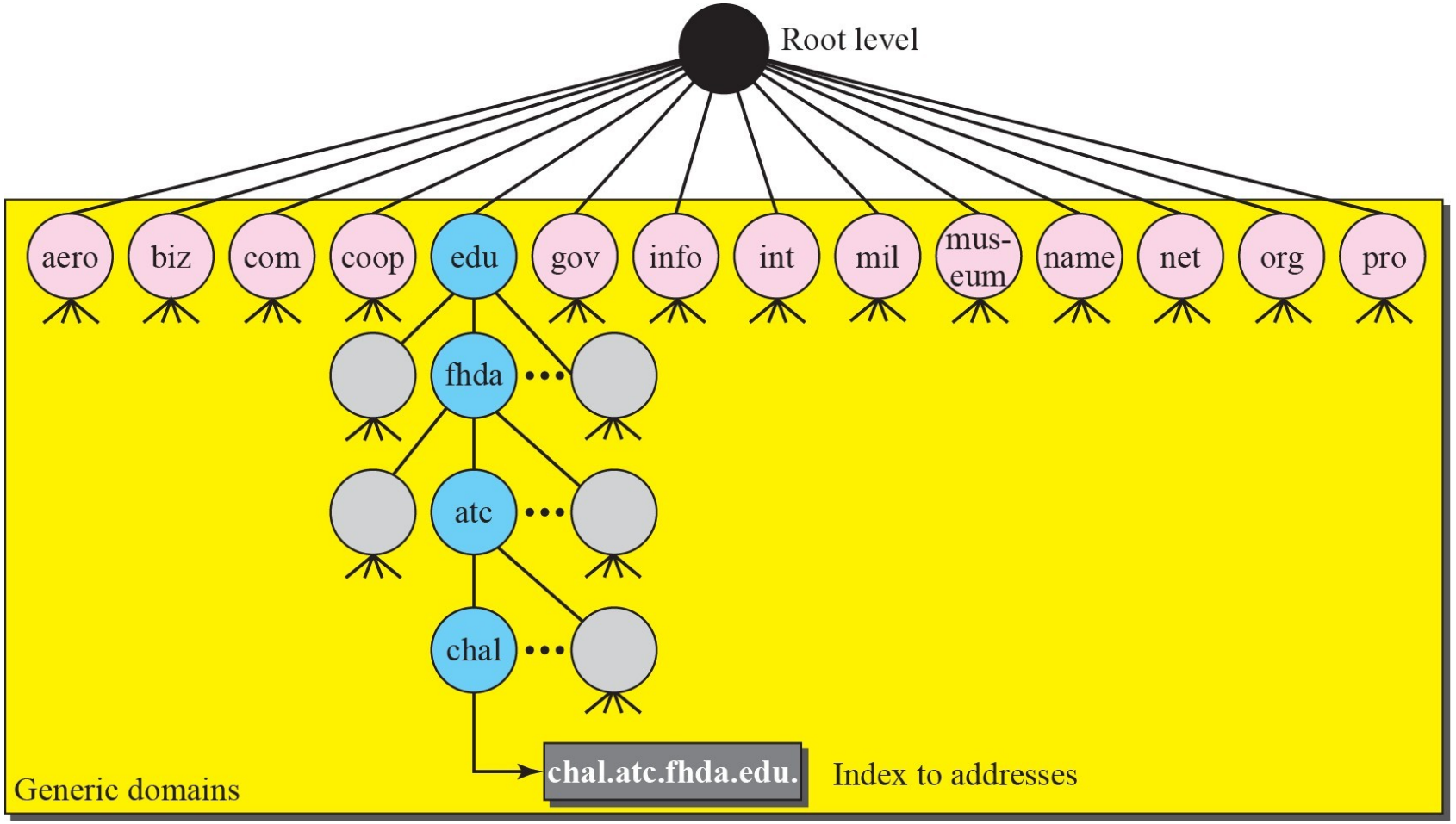
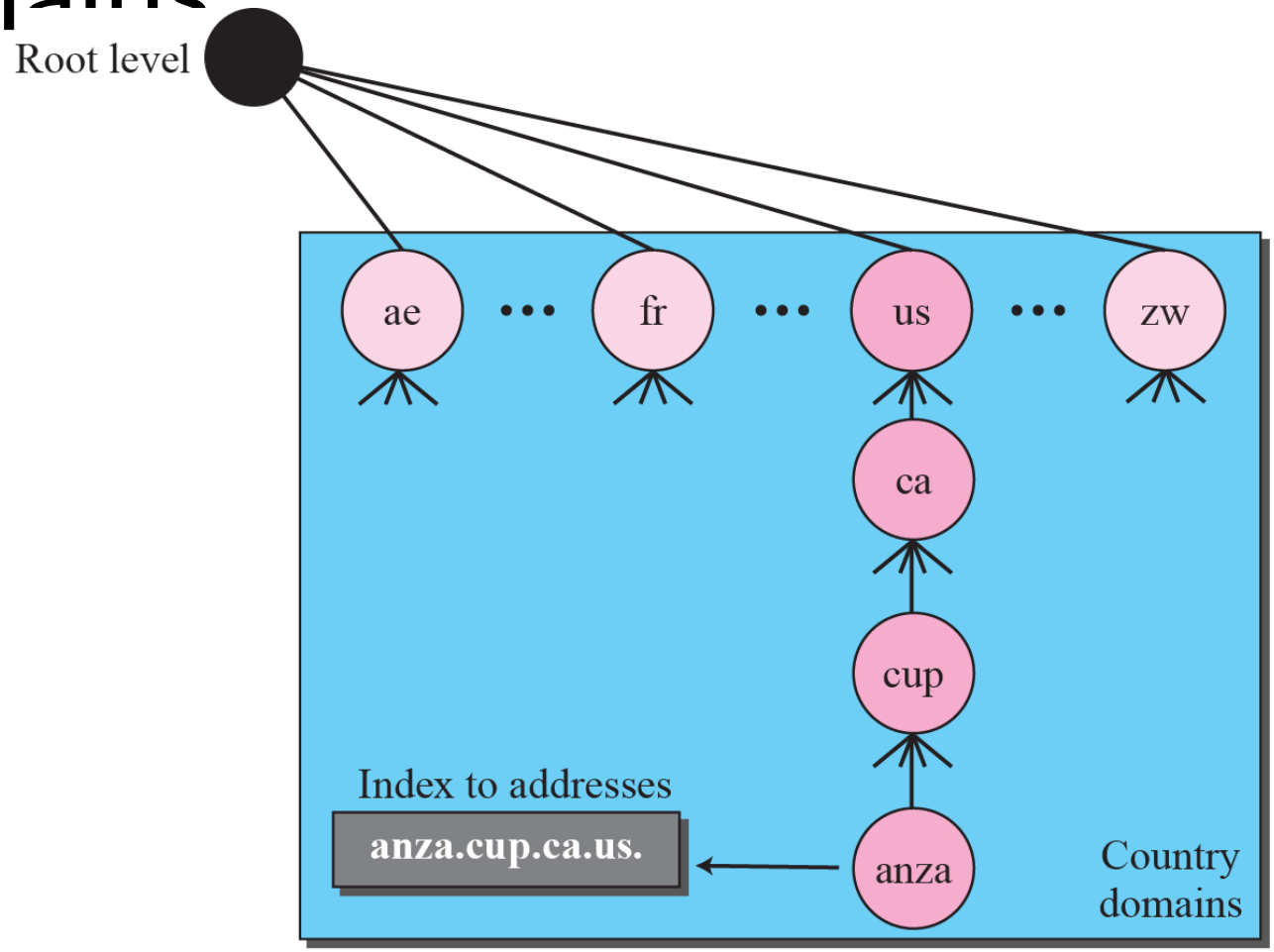


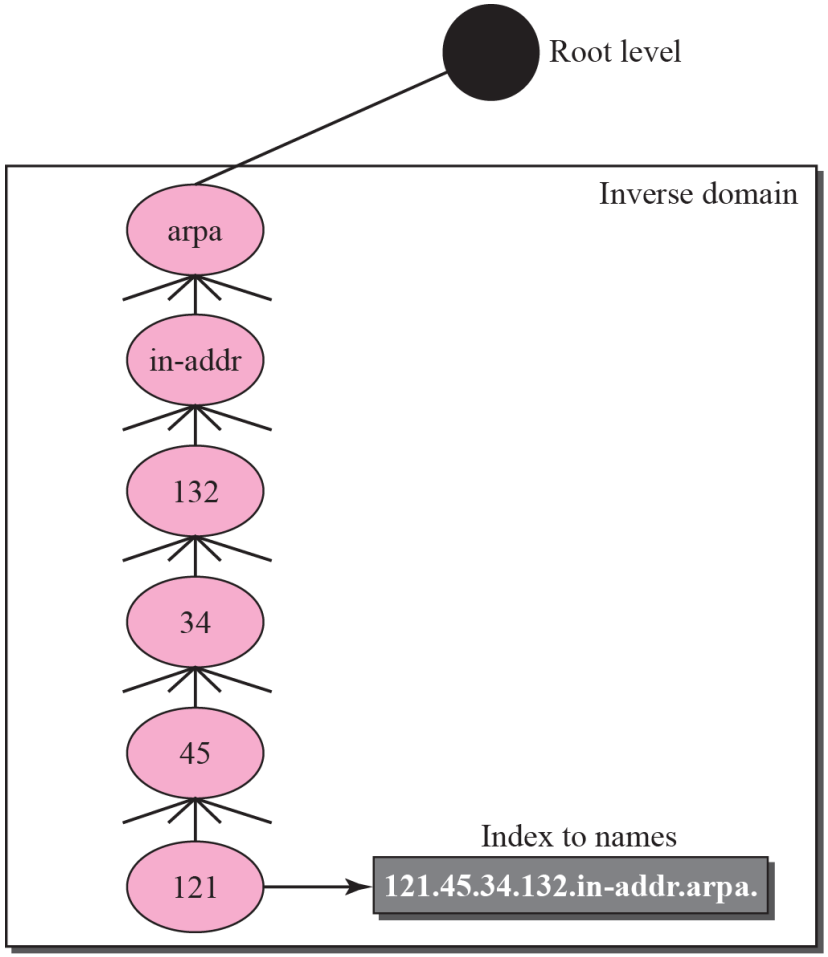
Table 19.1 *Generic domain labels*

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 non-profit organizations
name	Personal names (individuals)
net	Network support centers
org	Nonprofit organizations
pro	Professional individual organizations

Country domains



Inverse domain



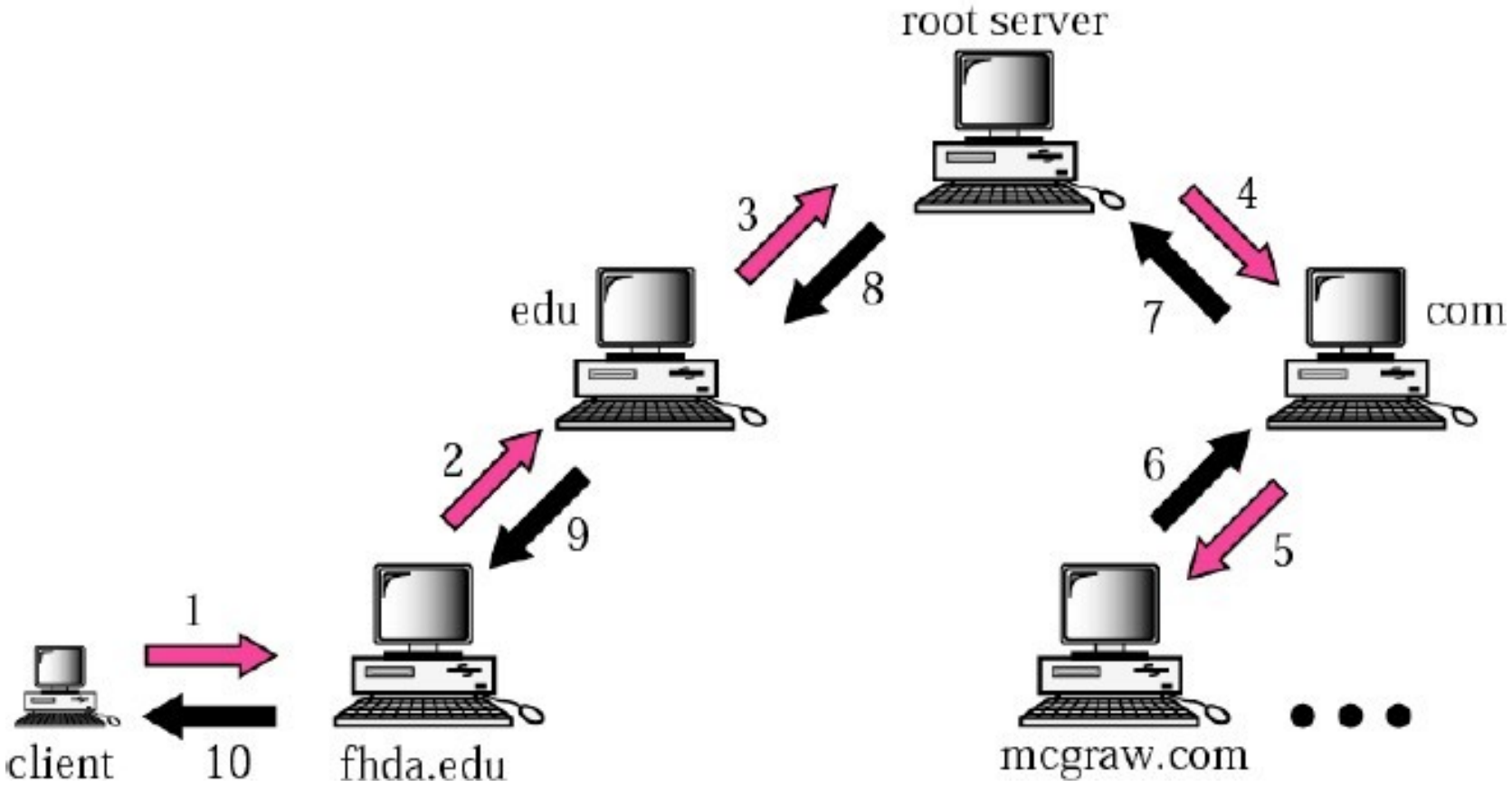
Inverse domain:

- It is used to map an address to a name.
- This type of query is known as inverse or PTR(pointer) query.
- Inverse domain is added in domain name space with..
  - First level node → arpa(historical reasons)
  - Second level node → in-addr(inverse address)
  - Netid
  - Subnet id
  - Host id
- Reading convention is followed here also (bottom to top in name space).

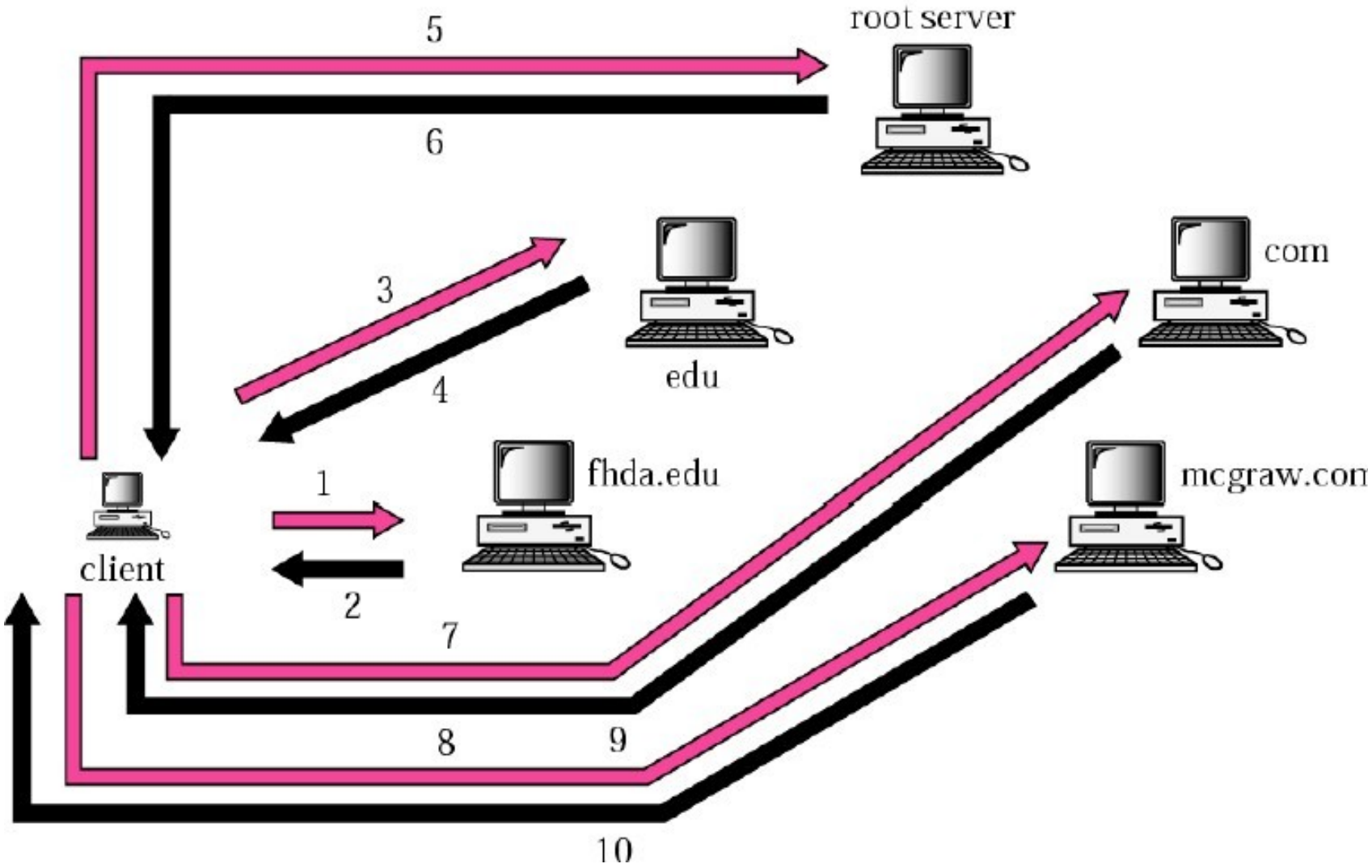
RESOLUTION

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

Recursive resolution



Iterative resolution



Caching:

- Each time a server receives a query for a name that is not in its domain, it needs to search in its database for server IP address.
- Reduction in search time will increase the efficiency.
- DNS handles this with caching

Caching

- When a server asks a mapping from another server and receives the response, it receives the response, it stores the information in its cache memory before sending it to its client.

- If some another client asks for the same mapping, it can check the cache memory and can solve the problem.

- However to inform the client that the response is coming from the cache memory and not from the authoritative source, the server marks the response unauthoritative.

Problems associated with caching

- If server caches the mapping for the long time, it can send the outdated mapping to the client.
- To counter this, two techniques are used.

Solution to the problems:

- First is authoritative always adds TTL- time to live. It defines the time in seconds that receiving server can cache the information.
- After that mapping is invalid, and query must be sent to authoritative server.
- Second, each server keep a TTL counter for each mapping it caches.
- The cache memory must be searched periodically, those mappings with an expired TTL must be purged.

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
- If the size of the response message is more than 512 bytes, a TCP connection is used.