# Host Configuration : DHCP

* Introduction :-

→ Most computers today need two other

piece of information : the address of a default router to be able to communicate with other networks and the address of a name server to be use names instead of addresses as well will see in the next chapter, In other words, four peice of information are normally needed.

(1) The IP address of the computer.
(2) The Subnet mask of the computer.
(3) The IP address of a router.
(4) The IP address of a name server.

* Previous Protocol :-

• RARP :-

→ RARP can provide only the IP address of the computer, but a computer today needs all four pieces of information mentioned above.

- **BOOTP :-**

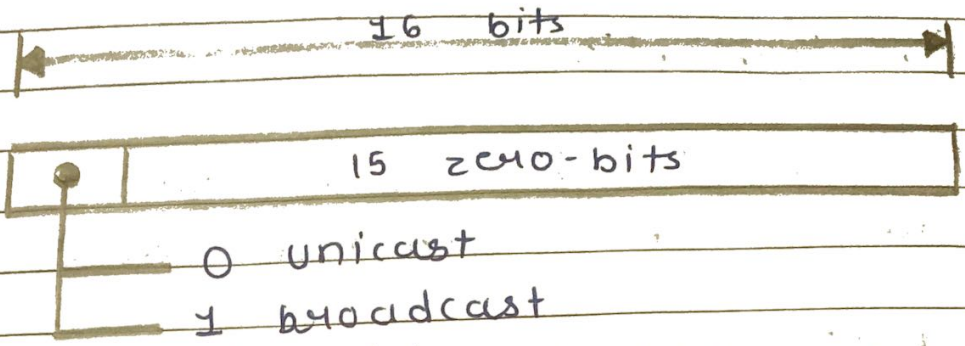
→ BOOTP, however is a static configuration.

→ The client can then use a TFTP message which is encapsulated in a UDP user datagram, to obtain the rest of the needed information.

* Packet format :

| Operation code | Hardware type | Hardware length | Hop count |
|---|---|---|---|
| Transaction ID | | | |
| Number of seconds | Flags | | |
| Client IP address | | | |
| Your IP address | | | |
| Server IP address | | | |
| Gateway IP address | | | |
| Client hardware address (16 bytes) | | | |
| Server name (64 bytes) | | | |
| Boot file name (128 bytes) | | | |
| Options (variable length) | | | |

- **Operation code:-** This 8-bit Field defines the type of DHCP packet: request (1) or reply (2).
- **Hardware type:-** This is an 5-bit field defining the type of physical network.
- **Hardware length:-** This is an 8-bit field defining the length of the physical address in bytes.
- **Hop count:-** This is an 8 bit Field defining the maximum number of hop.
- **Transaction ID:-** This is a 4-byte field carrying an integer.
- **Number of second:-** This is a 16-bit field that indicates the number.
- **Flag:-** This is a 16-bit field.




- **Client IP address :-** This is a 4-byte field that contains the client IP address.

- **Your IP address :-** This is a 4-byte field that contains the client IP address.

- **Server IP address :-** This is a 4-byte field containing the server IP address.

- **Server name :-** This is a 64-byte field that is optionally field by the server in a reply packet.

- **options :-** This is a 64-byte field with a dual purpose.

* Configuration :-

→ The DHCP has been advised to provide static and dynamic address allocation.

* Static Address Allocation :-

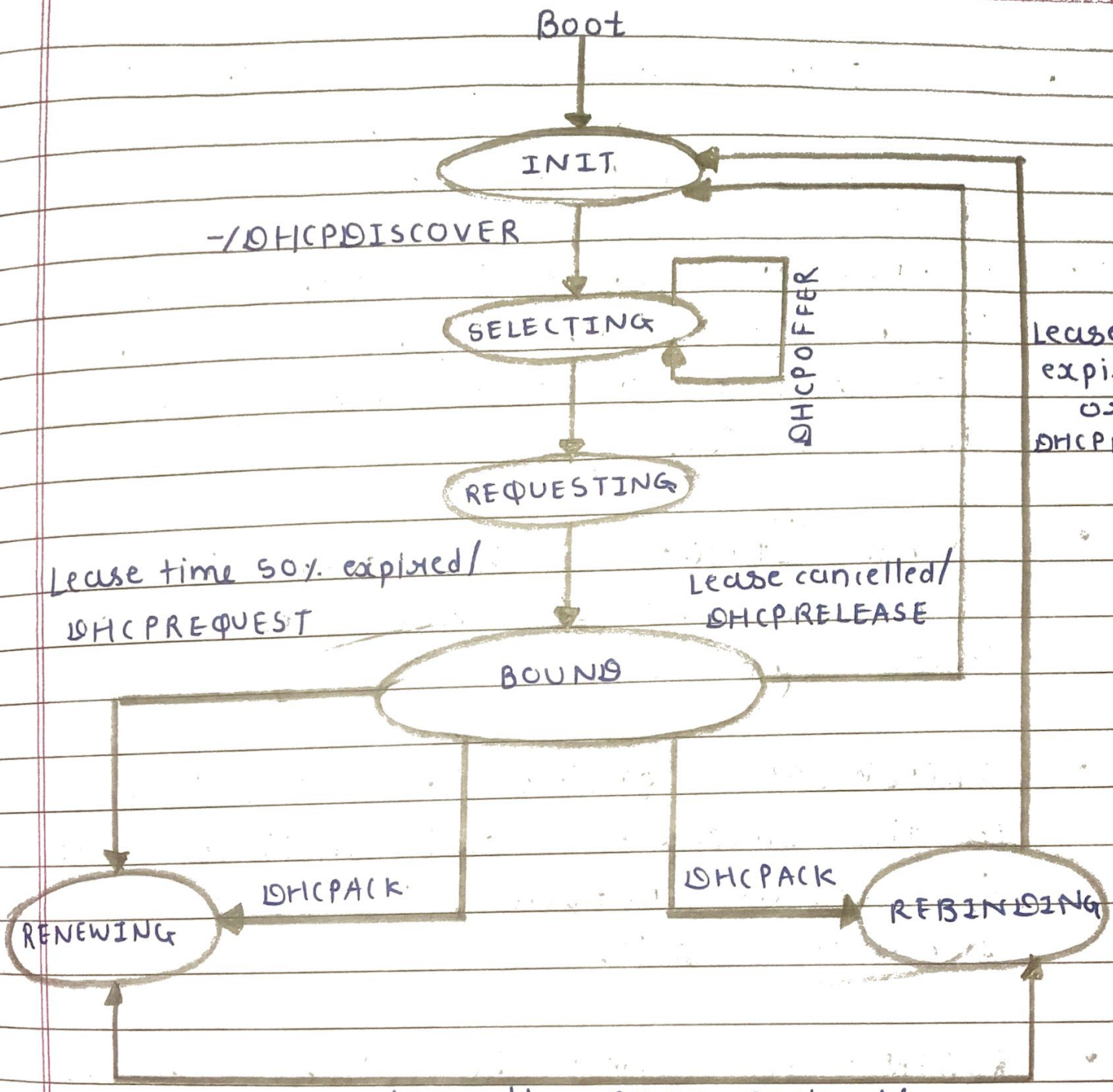
→ In this capacity, a DHCP server has a database that statically binds physical addresses to IP addresses.

* Dynamic Address Allocation :-

→ The dynamic aspect of DHCP is needed when a host moves from network to network or is connected and disconnected from a network.

→ The addresses assigned from the pool are temporary addresses.

* Transaction States :-




- **INIT State:-** when the DHCP client first states, it is in the INIT state.

- **SELECTING State:-** After sending the DHCPDISCOVER message, the client goes to the selecting state.

- **REQUESTING State:-** The client remains in the requesting state until it receives a DHCPACK message from the server.

- **Bound State:-** In this state, the client can use the IP addresses until the lease expires.

- **RENEWING State:-** The client remains in the renewing state until one of two event happens, the client goes to rebinding state.

- **REBINDING State:-** The client remains in the rebinding state until one of three events happens, it goes to the bound state and resets the timer.

## * Name Space :-

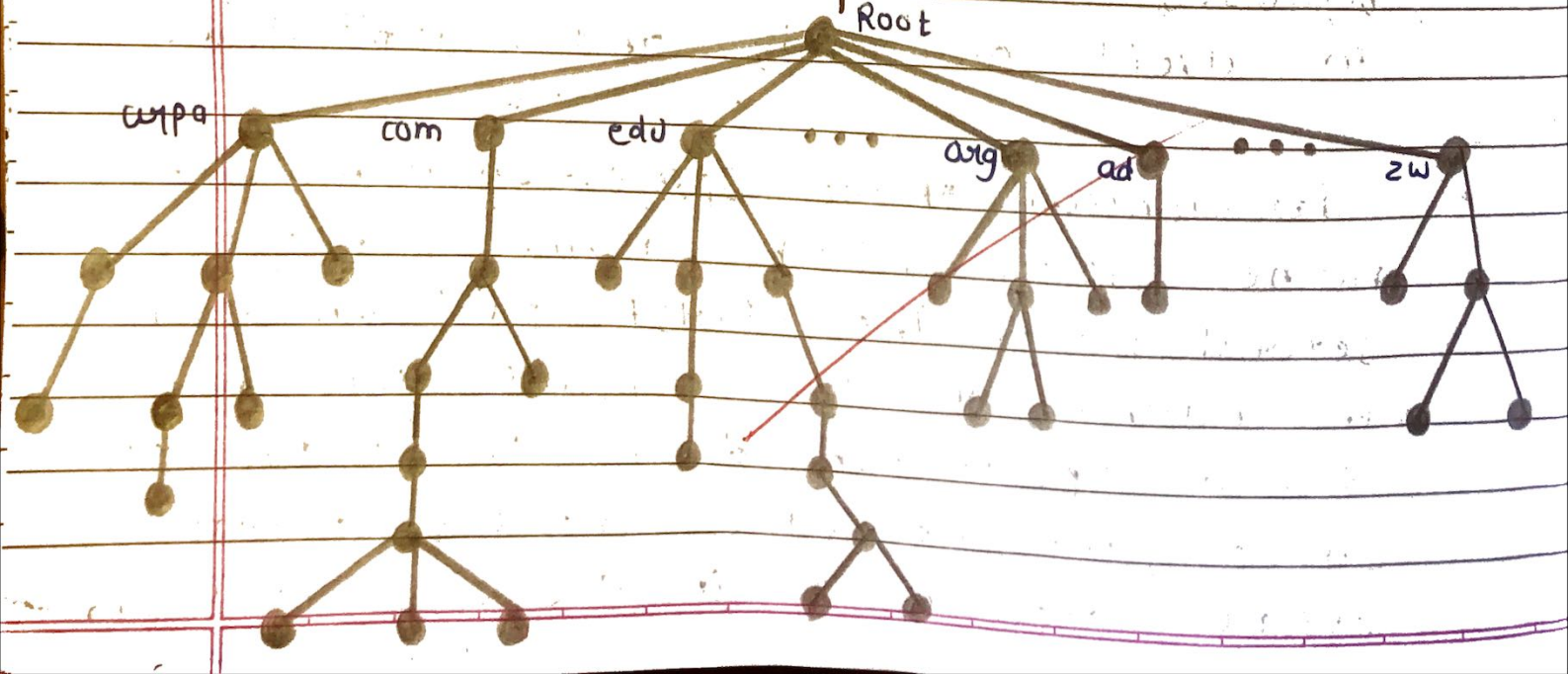
→ A name space that maps each address to a unique name can be organized in two ways : Flat or hierarchical.

- 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 names may or may not have a common section : if they do, it has no meaning. The main disadvantage of a flat 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 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.

→ For example, assume two colleges and a company call on of their computers challenger. The first college is given a name by the central authority such as fhda.edu, the second college is given a name berkeley.edu, and the company is given the name smart.com. When each of these organizations adds the name challenger to the name they have already been given, the end result is three distinguishable names: challenger.fhda.edu, challenger.berkeley.edu, and challenger.smart.com.

* Domain Name space :-



* Label:

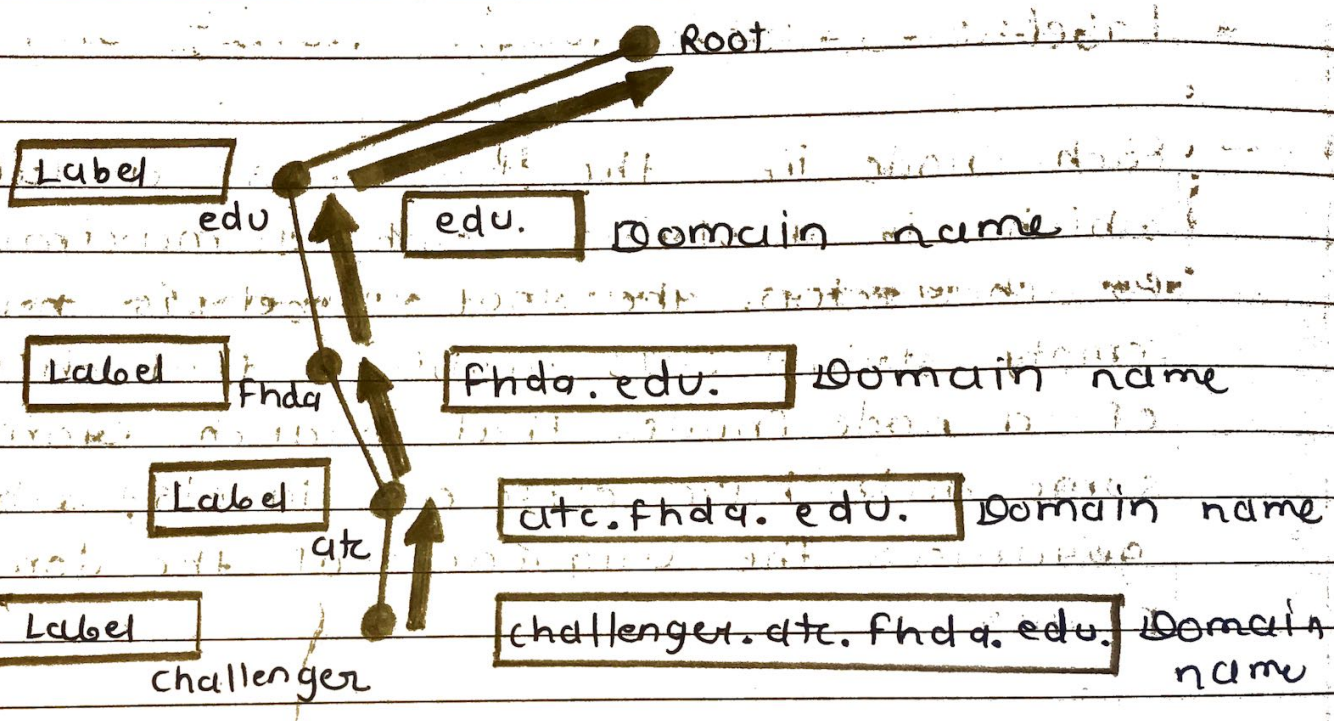
→ Each node in the tree has a label, which is a string with a maximum of 63 characters. The root label is null string (empty string). DNS requires that children of a node (nodes that branch from the same node) have different labels, which guarantees the uniqueness of the domain names.

* domain name:-

→ Each node in the tree has a domain name. A full domain name is a sequence of labels seperated by dots(.). The domain names are always ~~red~~ read from the node up to the root. The last label is the label of the root. 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.

Page
Date

## Domain names and labels




→ Fully Qualified Domain name (FQDN)
If a Label is terminated by a null string, it is called a fully qualified domain name (FQDN).

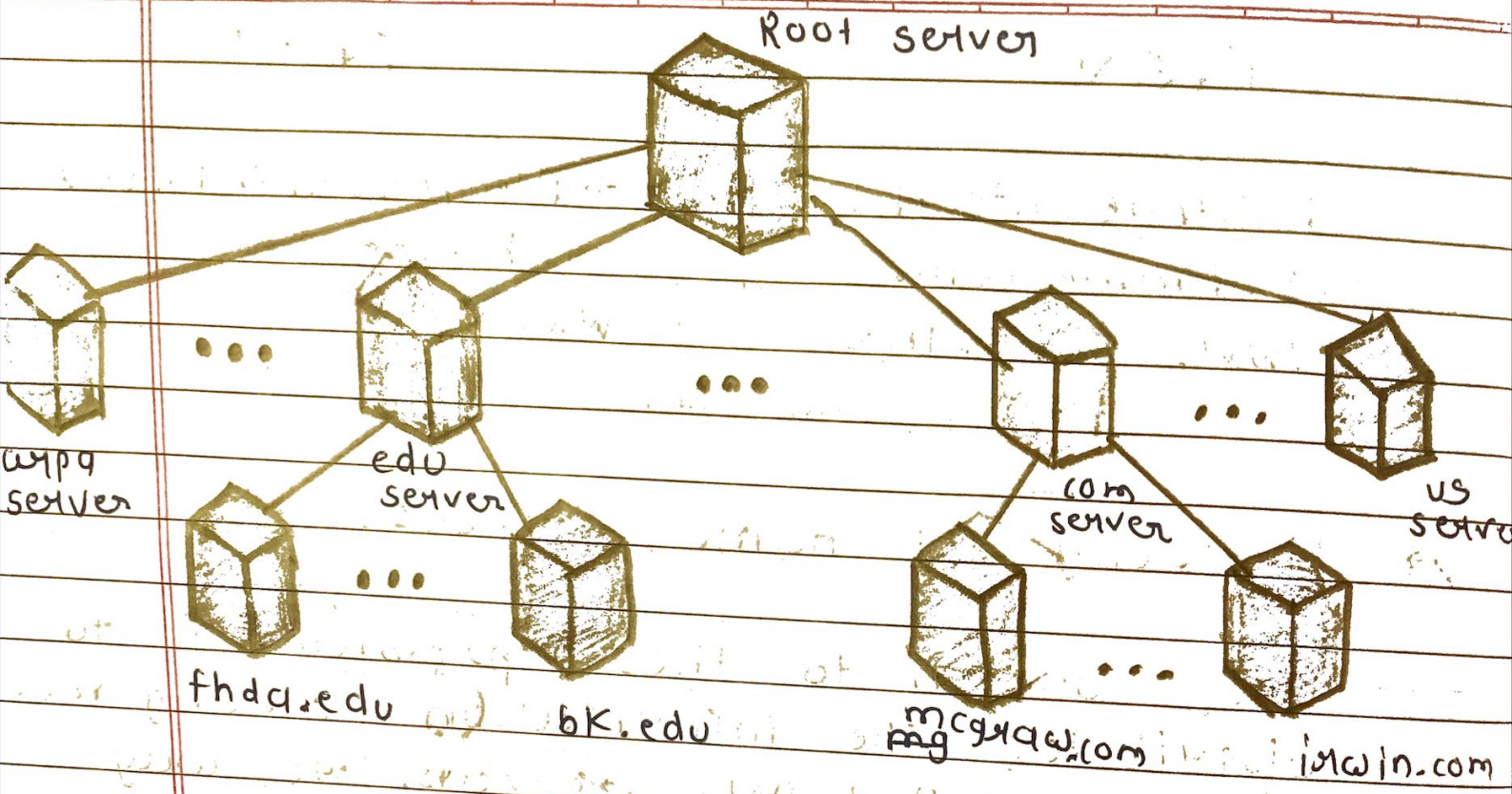
→ Partially Qualified Domain name (PQDN)
If a label is not terminated by a null string, it is called a partially qualified domain name (PQDN).

| FQDN | PQDN |
| --- | --- |
| challenger.atc.fhda.edu. | challenger.atc.fhda.edu |
| cs.hmme.com. | cs.hmme |
| www.funny.int. | www |

* Hierachy 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 as there are first-level nodes. Because a domain created this way could be ~~large~~ very large, DNS allows domains to be divided further into smaller domains. Each server can be responsible for either a large or small domain. In other words, we have a hierachy of servers in the same way that we have a hierachy of names




* **zone :-**

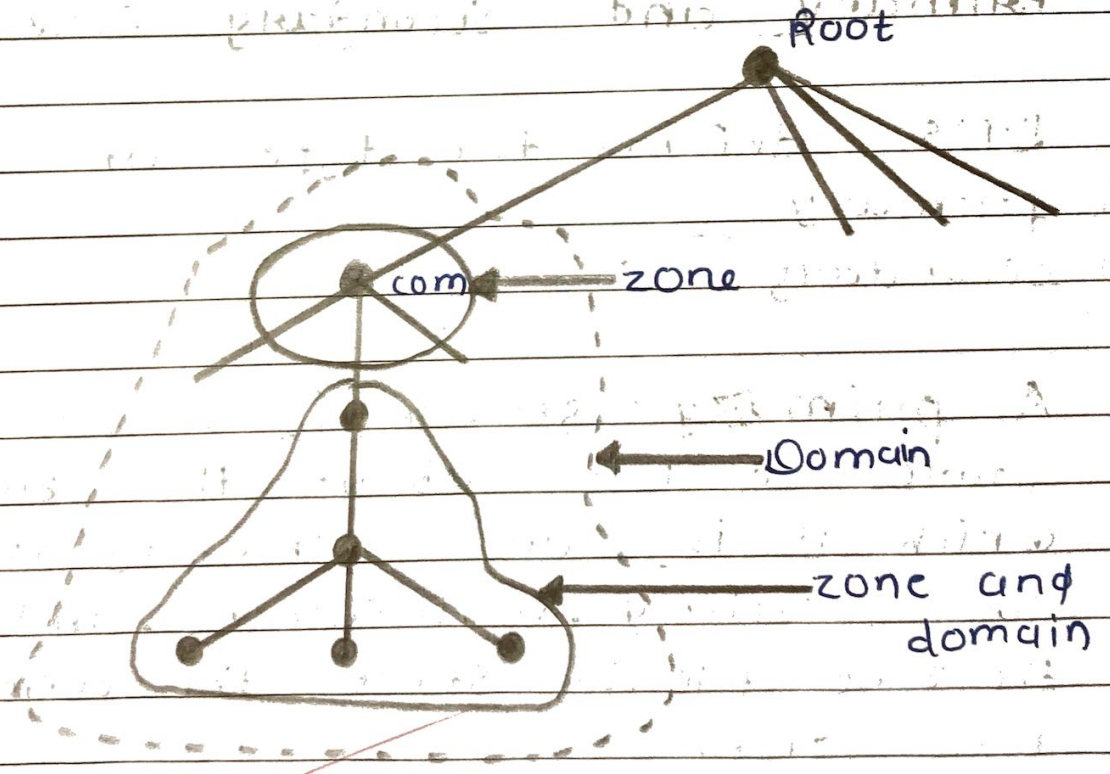
→ Since the complete domain name hierarchy cannot be stored on a single server, it is divided among many servers.

→ 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 entiree tree. If a server accepts responsibility for a domain and

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.

zones and domain




* Root server :- A root 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 root servers are distributed all around the world.

* **Primary and Secondary Servers:**

→ DNS defines two types of servers:
(1) primary
(2) 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.

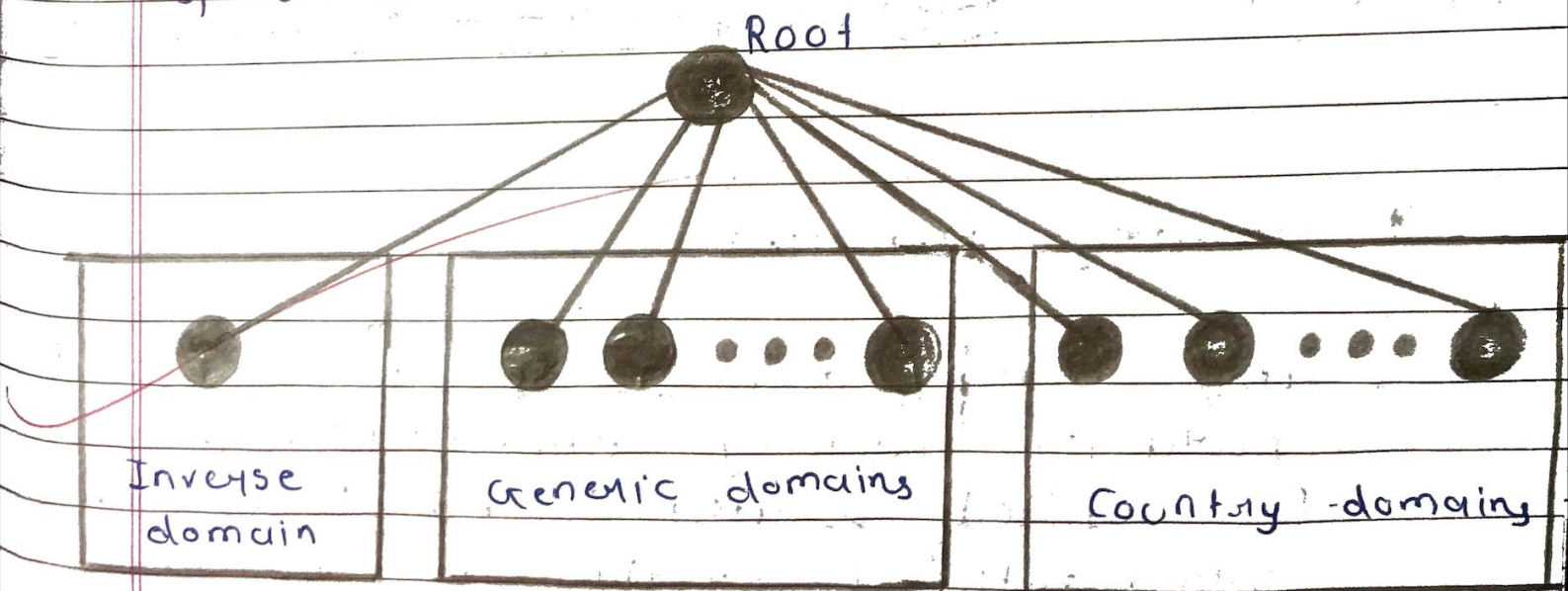
→ A secondary server is a server that transfers the complete information about a zone from another server

(primary or secondary) and stores the file on its local disk. The secondary server neither creates nor updates the zone files.

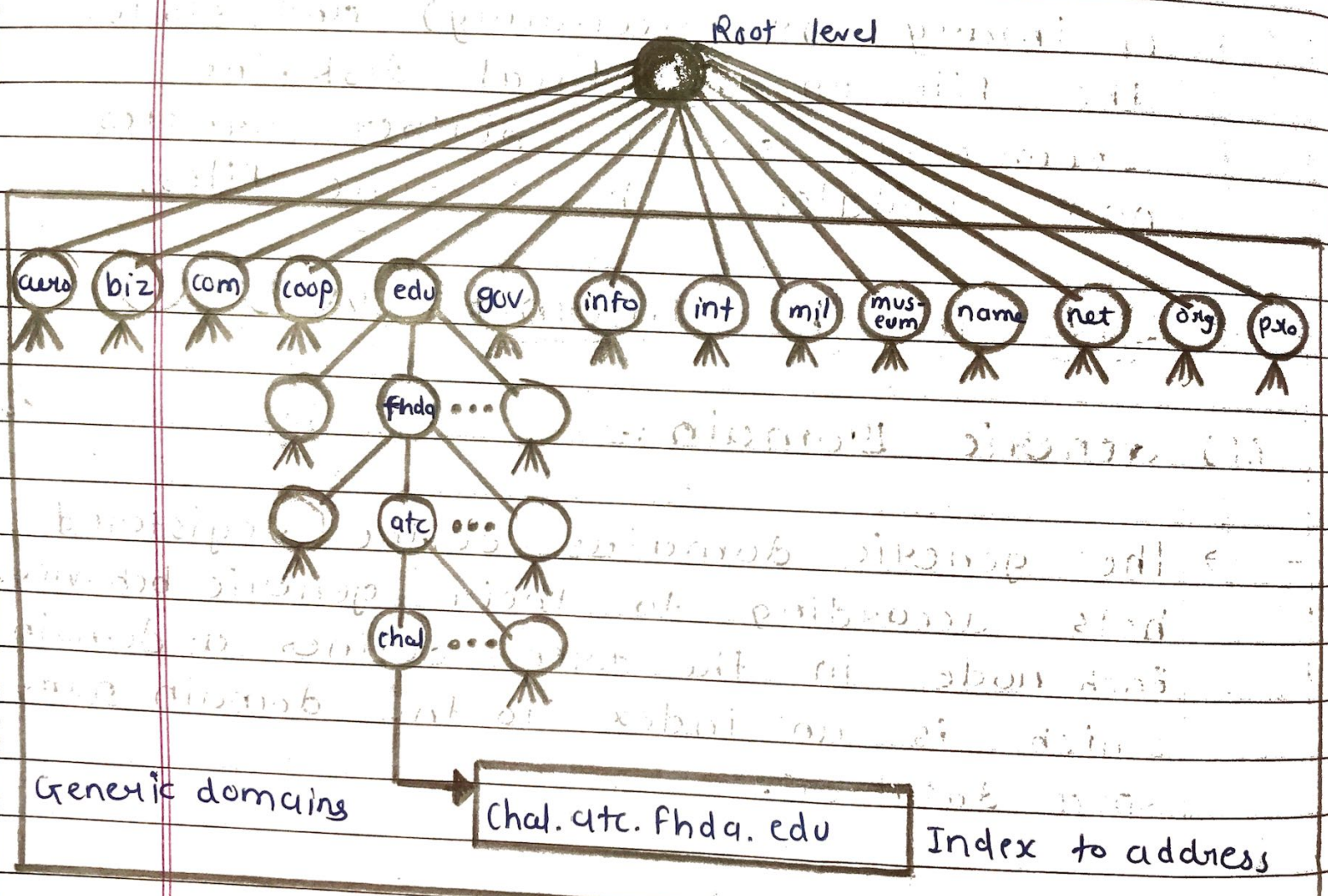
## * DNS in the Internet :

### (1) Generic Domain :-

→ The generic domains define registered hots according to their generic behaviour. Each node in the tree defines a domain, which is an index to the domain name space database.



Generic domains




* Country Domains

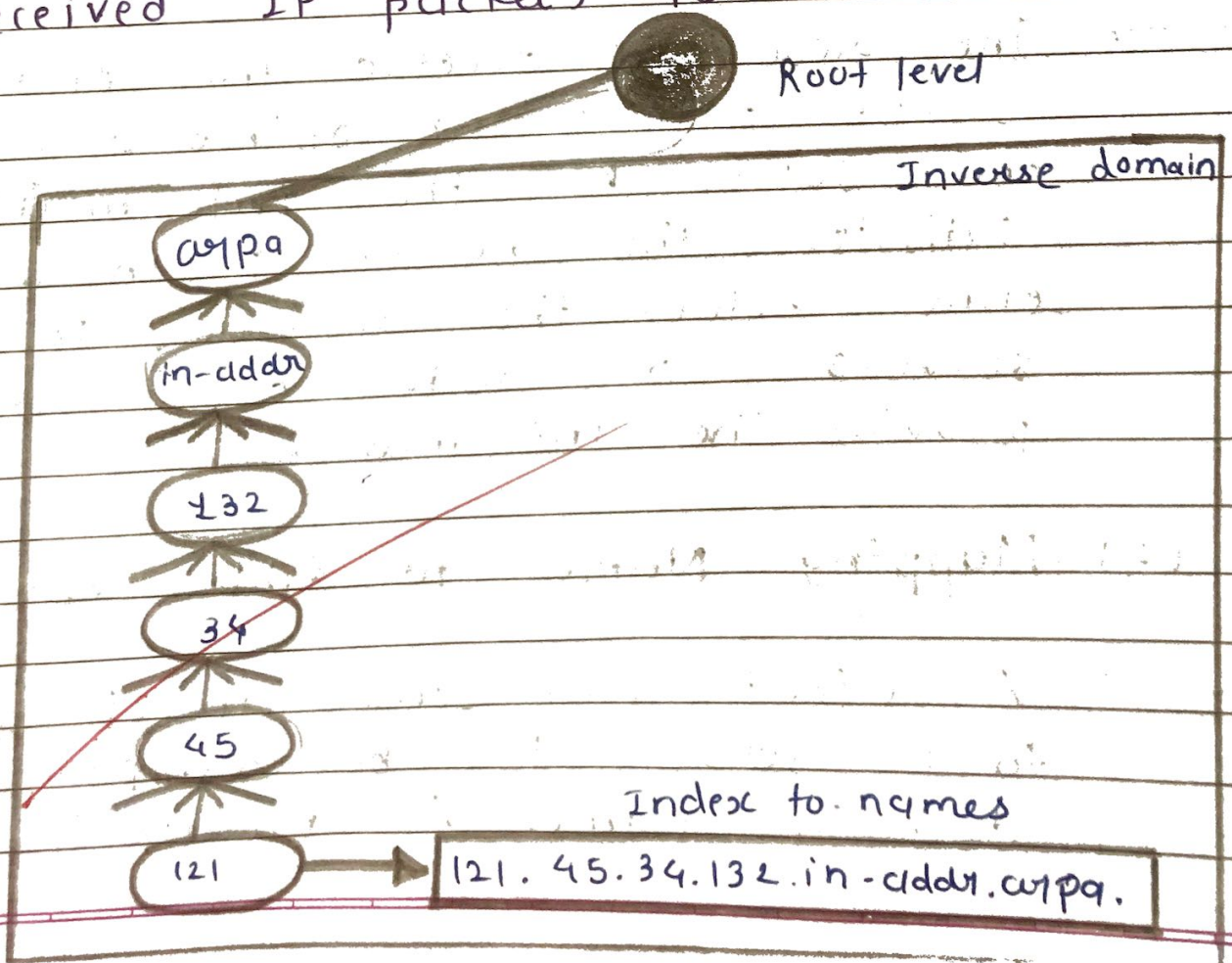
→ The country domain selection uses two-character country abbreviations (e.g., us for United States) second labels can be organizational, or they can be more specificational designations. The United states,

Page
Date

for example, Uses state abbreviations as a subdivision of US (e.g., CA.US.).

## * 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 authorised clients, only the IP address of the client (extracted from the received IP packet) is listed.



* Resolution :-

(1) 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 closet 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.

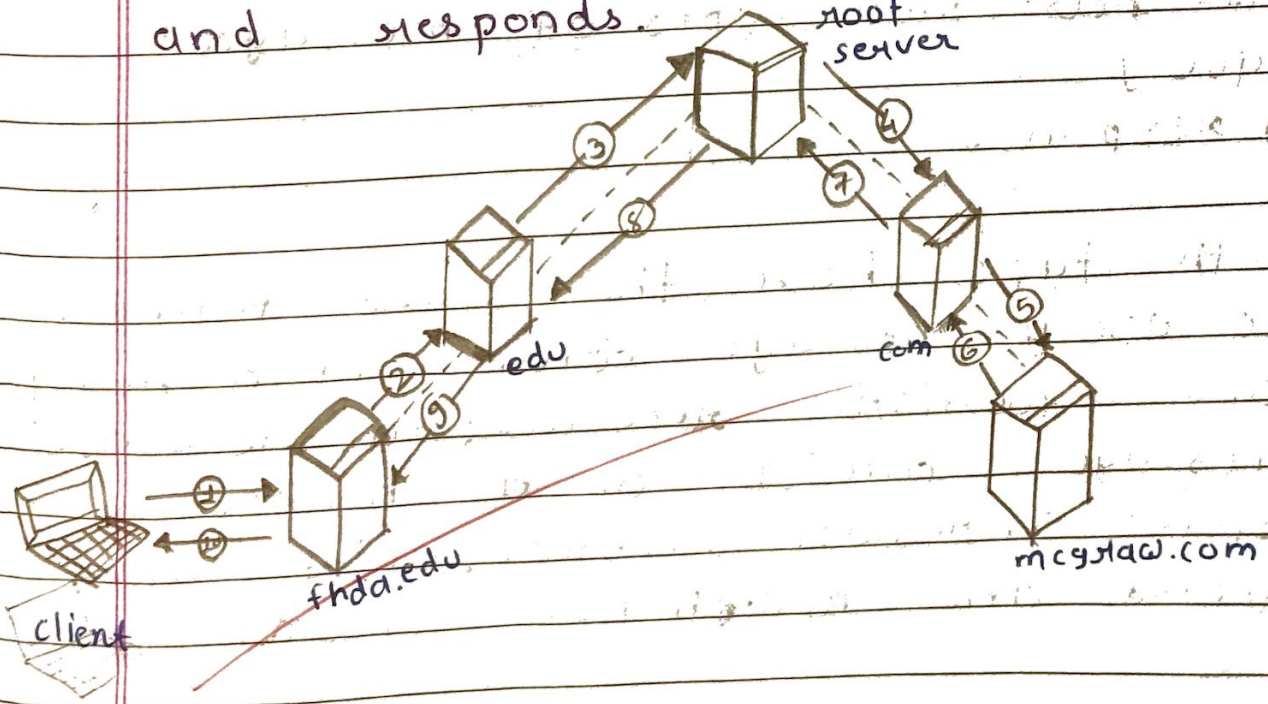
(2) Mapping Names to Addresses :-

→ 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.

(3) 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 domain name, it checks its database and responds.



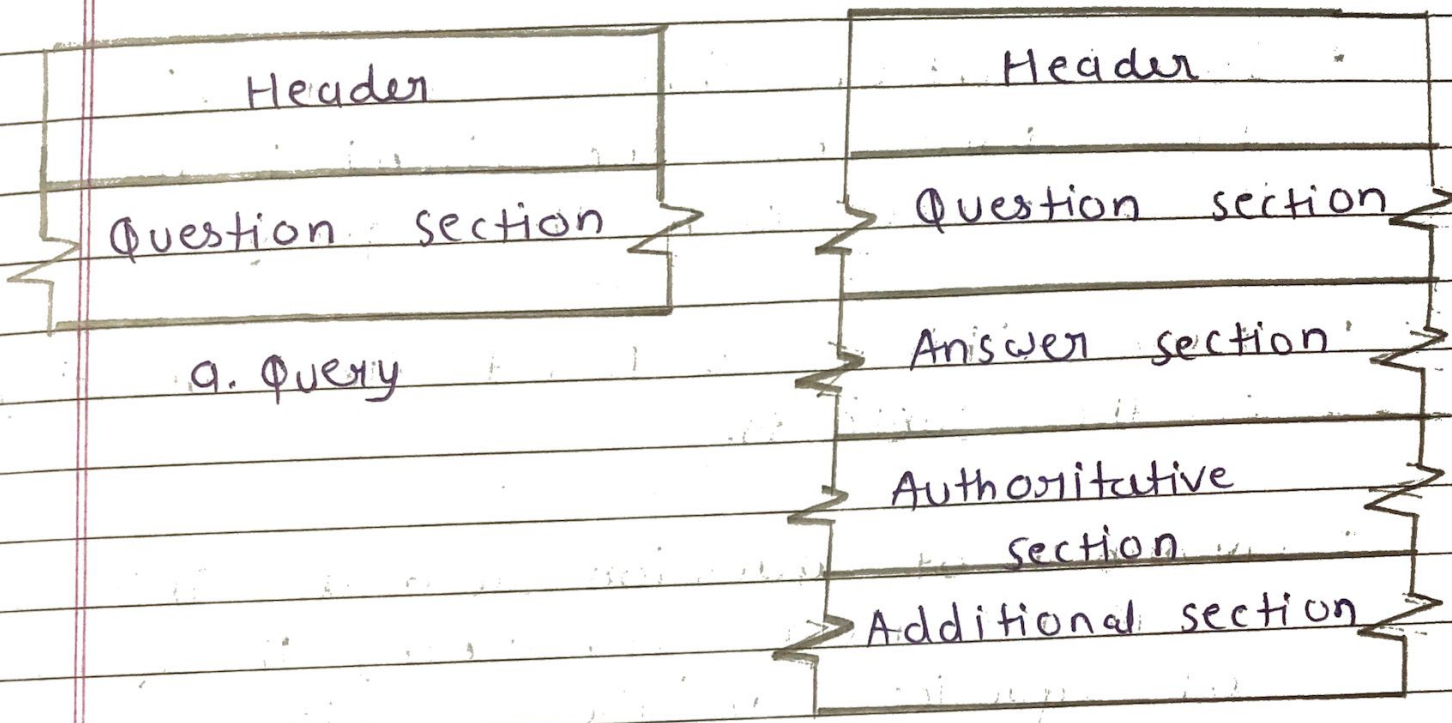
(4) Iterative Resolution:

→ 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 the IP address of the server that it thinks can resolve the query.

* DNS Messages :-

→ DNS has two types of messages:
(1) query
(2) response

→ Both types have the same format. The query message consits of a header and question records; the response message consists of a header, question records, answer records, authoritative records, and additional records.




* **Header :-**

→ Both query and response messages have the same header format with some fields set to zero for the query messages.

| 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 |

- **Identification:-** This is a 16-bit field used by the client to match the response with the query.

- **Flags:-** This is a 16-bit field consisting of the subfields.

- **Number of question records:** This is a 16-bit field containing the number of queries in the question section of the messages.

- **Number of answer records:** This is a 16-bit field containing the number of answer records in the answer section of the response message. Its value is zero in the query message.

- **Number of authoritative records:** This is a 16-bit field containing the number of authoritative records in the authoritative section of a response message. Its value is zero in query message.

- Number of additional records: This is a 16-bit field containing the number of 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.

* Answer Section: This is a section consisting of one or more resource records. It is present only on response message. This section includes the answer from the server to the Client (resolver).

* Authoritative Section: This is a section consisting of one or more resource records. It is present only on response messages. This section gives information (domain name) about one or more authoritative server for the query.

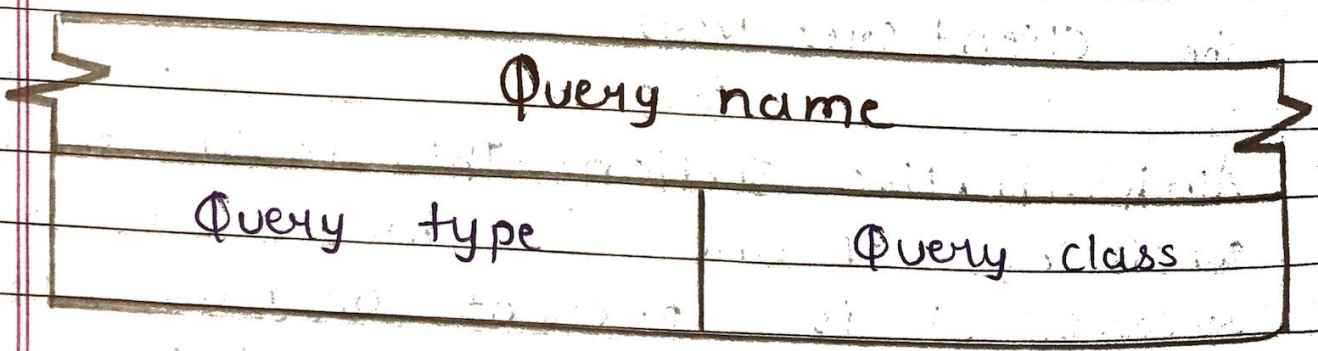
* Additional Information Section:

→ This is a section consisting of one or more resource records. It is present only on response messages. This section provides additional information that may help the resolver.

* Types of Records:-

(1)* Question Record

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




- Query name: This is a variable-length field containing a domain name. The count field refers to the number of

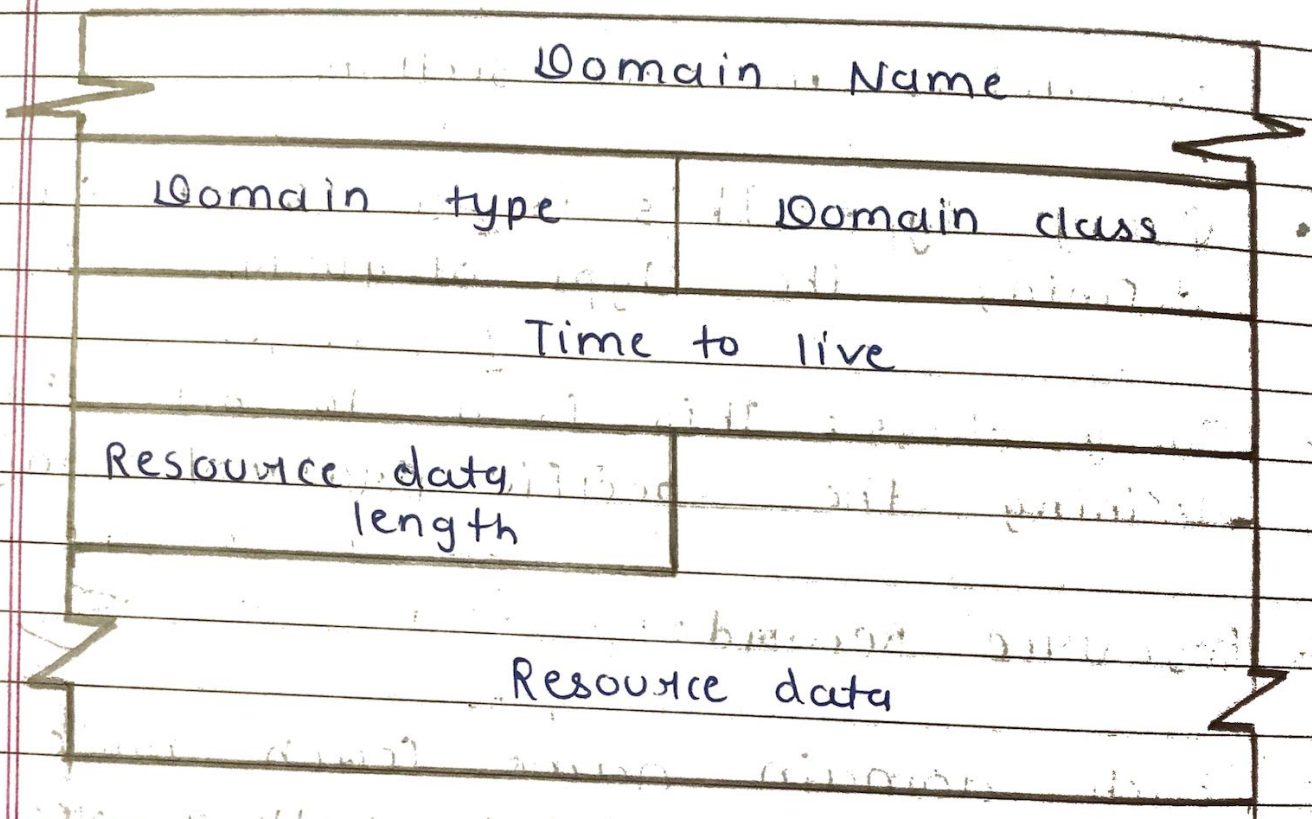
characters in each section.

- Query type: This is a 16-bit field defining the type of query.
- Query class: This is a 16-bit field defining the specific protocol using DNS.

(2)* Resource Record:

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

→




- **Domain name**: This is a variable-length field containing the domain name.

- **Domain type**: This field is the same as the query type field in the question record except the last two types are not allowed.

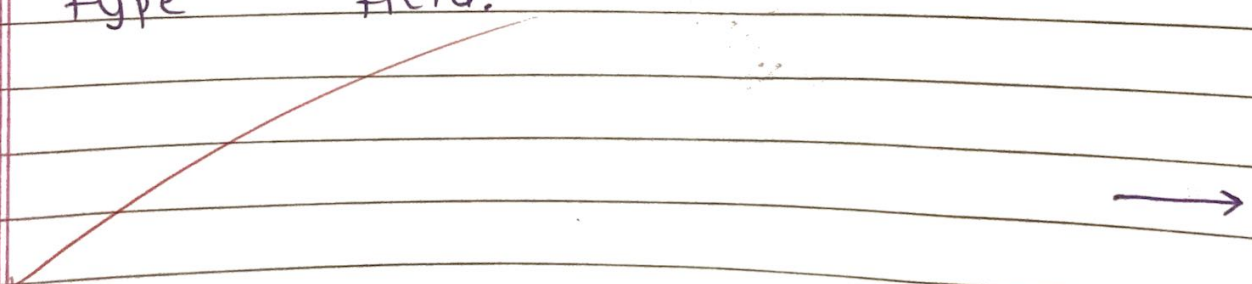
- **Domain class**: This field is the same as the query class field in the question record.

- **Time-to-live :-** This is a 32-bit field that defines the number of seconds the answer is valid.

- **Resource data length :** This is a 16-bit field defining the length of ~~the~~ the resource data.

- **Resource data :** This is a variable-length field containing the answer to the query (in the answer section) or domain name of the authoritative server (in the authoritative section) or additional information (in the additional information section).

- The format and contents of this field depend on the value of the type field.



* Compression :-

→ DNS requires that a domain name be replaced by an offset pointer if it is repeated.

→ For example, in a resource record the domain name is usually a repetition of the domain name in the question record. For effeciency, DNS defines a 2-byte offset pointer that points to a previous occurrence of the domain or part of it.