Lab 8 Domain Name System (DNS)

1. Introduction of DNS

The Domain Name System (**DNS**) is a system that maps domain names to addresses or addresses to domain names. By using DNS, computer users are able to directly communicate with domain names. DNS is designed as an online and distributed database system as well as a client/server application, and thus the distributed structure makes it strongly fault-tolerant.

2. Domain Name Space

Domain name space is designed hierarchically. 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 the following figure).

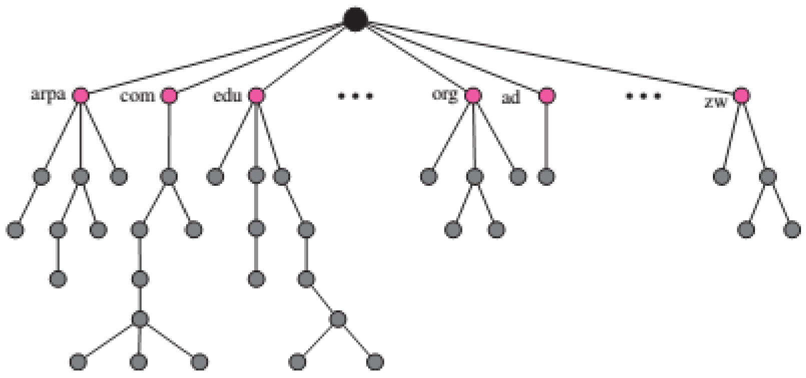

Figure 11-1 Domain Name Space

2.1. 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.

2.2. 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 11-2 shows some domain names.

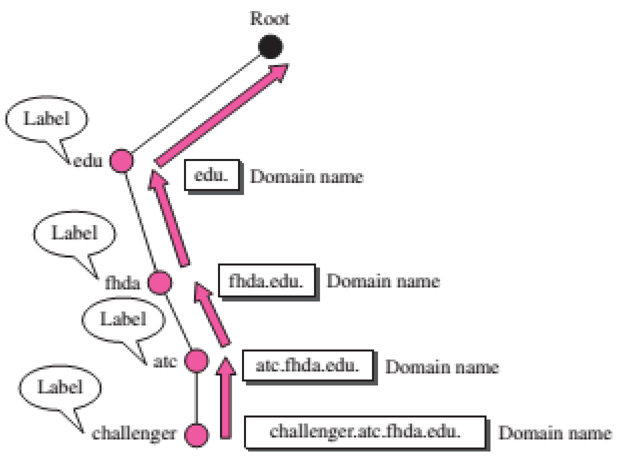

Figure 11-2 Domain names and labels

2.2.1. 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.atc.fhda.edu* is the FQDN of a computer named *challenger*.

2.2.2. 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 *atc.fhda.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.fhda.edu* before passing the address to the DNS server.

2.2.3. 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 11-3 shows some domains. Note that a domain may itself be divided into domains (or subdomains as they are sometimes called).

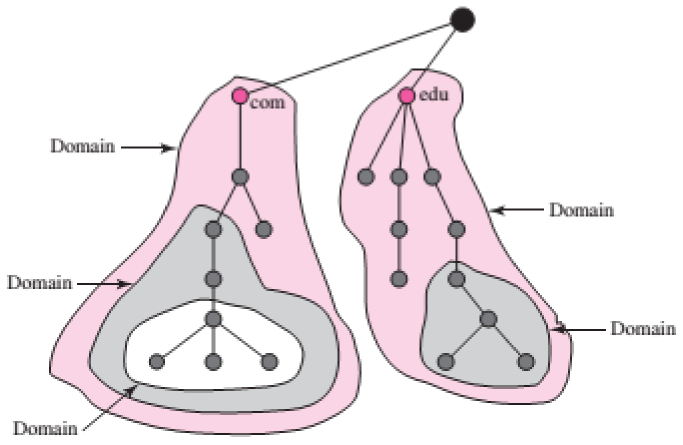

Figure 11-3 Domain

3. Distribution of Name Space

In the Internet, the domain name space (tree) is divided into three different sections: generic domains, country domains, and the inverse domain (see Figure 11-4).

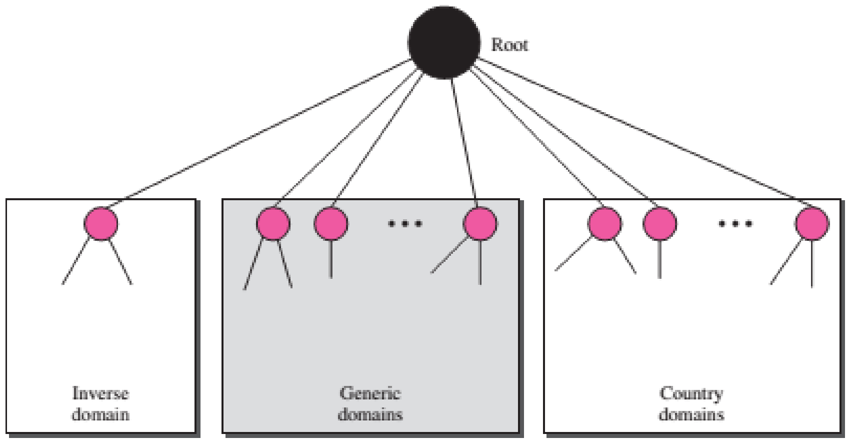

Figure 11-4 DNS used in the Internet

3.1. 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 11-5).

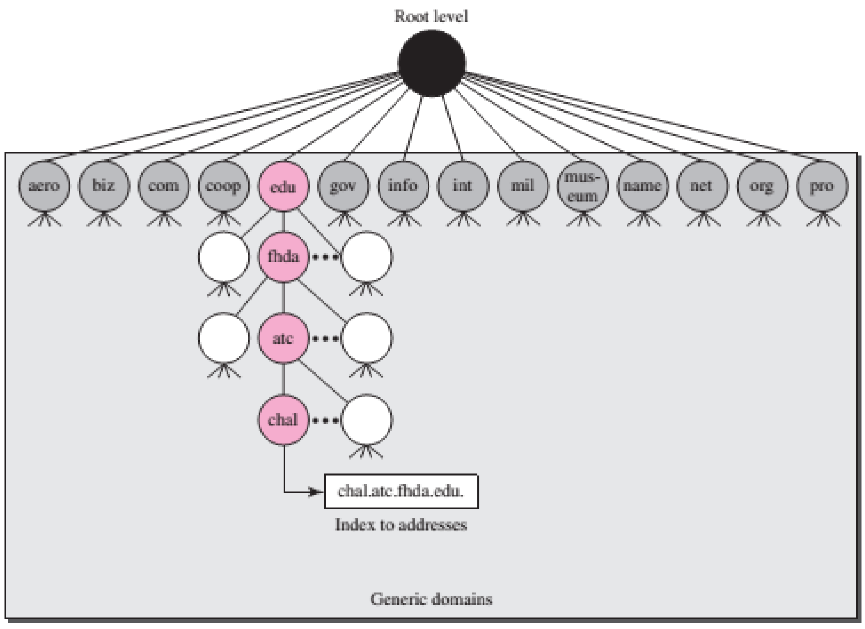

Figure 11-5 Generic domains

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 11-1.

Table 11-1 Generic domain labels

Label	Description	Label	Description
aero	Airlines and aerospace companies	int	International organizations
biz	Businesses or firms (similar to “com”)	mil	Military groups

com	Commercial organizations	museum	Museums and other nonprofit organizations
coop	Cooperative business organizations	name	Personal names (individuals)
edu	Educational institutions	net	Network support centers
gov	Government institutions	org	Nonprofit organizations
info	Information service providers	pro	Professional individual organizations

3.2. Country Domains

The **country domain**s section uses two-character country abbreviations (e.g., cn for China). Second labels can be organizational, or they can be more specific, national designations.

3.3. 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.

Inverse domain are also hierarchical. This means the net-id part of the address should be at a higher level than the subnet-id part, and the subnet-id part higher than the host-id 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 IP address such as **132.34.45.121** (a class B address with net-id 132.34) is read as **121.45.34.132.in-addr.arpa**. See Figure 11-6 for an illustration of the inverse domain configuration.

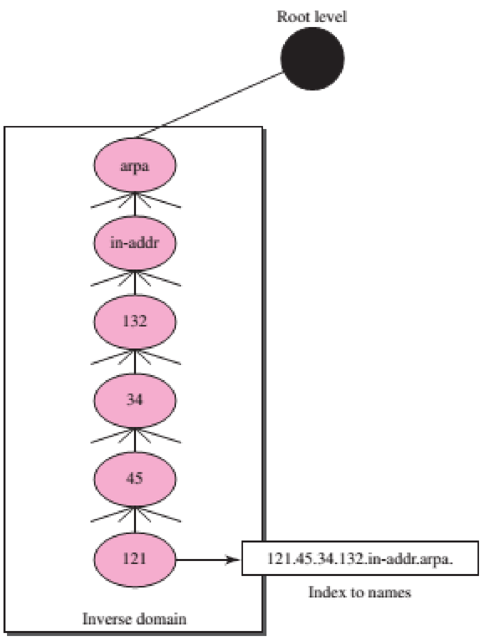

Figure 11-6 Inverse domain

4. Format of DNS message

DNS message consists of 12-byte header and 4 variable fields:

Identification (16-bit)	Flags (16-bit)
Number of question records (16-bit)	Number of answer records (16-bit)
Number of authoritative records (16-bit)	Number of additional records (16-bit)
Question section	
Answer section (available number)	
Authoritative section (available number)	
Additional section (available number)	

Figure 11-7 DNS message format

1) **Identification**: This 2-byte subfield is used by the client to match the response with the query.
2) **Flags**: This 2-byte subfield is a collection of subfields that define the type of the message which is shown in Figure 11-8.:

QR	Opcode	AA	TC	RD	RA	Reserved	AD	CD	rcode
1-bit	4-bit	1-bit	1-bit	1-bit	1-bit	1-bit	1-bit	1-bit	4-bit

Figure 11-8 DNS Flags fields

The description of each field is shown as follows:

- **QR** (Query/Response): 0 for the questioner (query) and 1 in the response (answer).
- **Opcode**: 0 for standard *query*, 1 for *inverse query* and 2 for DNS *status request.*
- **AA** (Authoritative Answer): When it is set to be 1, it indicates that the name server is an authoritative server; this field is only valid in responses.
- **TC** (TrunCation): Specifies that this message was truncated due to length greater than that permitted on the transmission channel. For example, a response message contains a lot of name servers, the size may exceed the permitted MTU. This message will be fragmented and TC will be set by 1.
- **RD** (Recursion Desired): This bit is be set in a query if the destination name server does not contain the requested information, the client request recursive query.
- **RA** (Recursion Available): This bit is valid in a response (answer) and denotes whether recursive query support is available (1) or not (0) in the name server.
- **AD** (Authenticated data): This bit indicates that all data has been identified by the server.
- **CD** (Checking Disabled): This bit indicates that the requester allows accepting unidentified data.
- **Rcode**: This 4-bit field indicates whether error occurs in the DNS response.

3) **Number of question records**: This 2-byte subfield contains the number of queries in the question section of the message.
4) **Number of answer records**: This 2-byte subfield contains the number of answer records in

the answer section of the response message.

5) **Number of authoritative records**: This 2-byte subfield contains the number of authoritative records in the authoritative section of a response message.
6) **Number of additional records**: This 2-byte subfield contains the number additional records in the additional section of a response message.
7) **Question Section**: This is a section consisting of one or more question records. It is present in both query and response messages. The format of each question is shown as follows:

Query name	
Query type	Query class

Figure 11-9 DNS question section

- **Query name**: The domain name being queried.
- **Query type**: The resource records being requested.
- **Query class**: The resource record(s) class being requested e.g. internet, chaos etc.

8) **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). The format of resource record is shown as follows:

Domain name	
Type	Class
Time to live	
Resource data length	Resource data

Figure 11-10 DNS answer section

- **Domain name**: This field recorded as a variable-length field following the same formatting as the Query Name field.
- **Type**: The resource record type value, corresponding to Query Type in the query section.
- **Class**: The resource record class code, corresponding to Query Class in the query section.
- **Time to Live**: The TTL expressed in seconds as a 32-bit unsigned field.
- **Resource Data Length**: 2-byte field indicating the length of the resource data.
- **Resource Data**: Variable-length data corresponding to the resource record type.

9) **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 servers for the query.

Domain name	
Type	Class
Time to live	
Resource data length	Resource data

Figure 11-11 DNS authoritative section

- **Domain name**: This field recorded as a variable-length field following the same formatting as the Query Name field.

- **Type**: The resource record type value, corresponding to Query Type in the query section.
- **Class**: The resource record class code, corresponding to Query Class in the query section.
- **Time to Live**: The TTL expressed in seconds as a 32-bit unsigned field.
- **Resource Data Length**: 2-byte field indicating the length of the resource data.
- **Resource Data**: Variable-length data corresponding to the resource record type.

10) **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. 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.

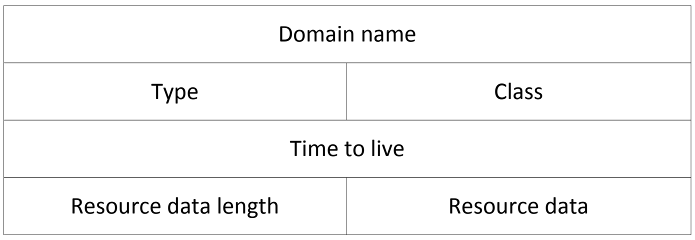

Figure 11-12 DNS additional information section

- **Domain name**: This field recorded as a variable-length field following the same formatting as the Query Name field.
- **Type**: The resource record type value, corresponding to Query Type in the query section.
- **Class**: The resource record class code, corresponding to Query Class in the query section.
- **Time to Live**: The TTL expressed in seconds as a 32-bit unsigned field.
- **Resource Data Length**: 2-byte field indicating the length of the resource data.
- **Resource Data**: Variable-length data corresponding to the resource record type.

5. Forward Lookup versus Reverse Lookup

5.1. 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 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.

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.

5.2. Forward Lookup: 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. 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. The procedure is the same.

5.3. Reverse Lookup: 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 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.

6. Recursive Resolution versus Iterative Resolution

6.1. 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 11-13.

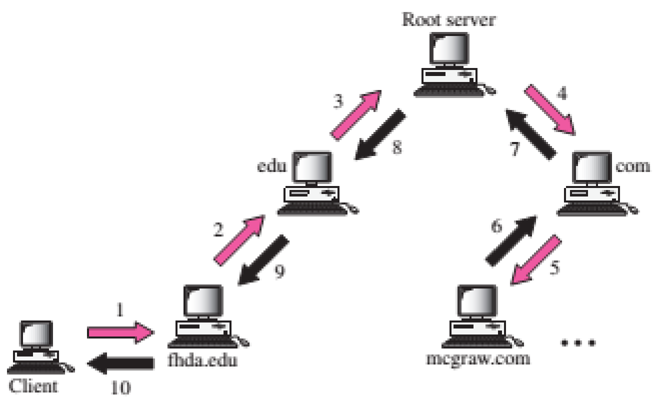

Figure 11-13 Recursive Resolution

6.2. 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 11-14 the client queries four servers before it gets an answer from the mcgraw.com server.

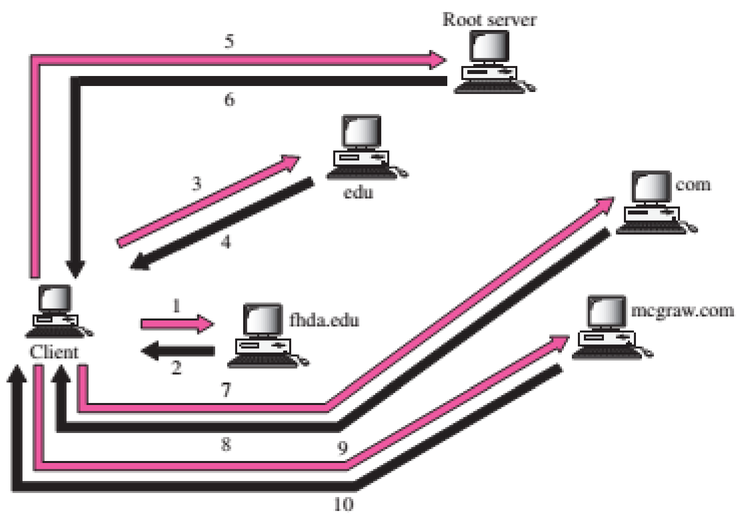

Figure 11-14 Iterative Resolution

7. 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 inform the client that the response is coming from the cache memory rather than 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 would become 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. Mappings with an expired TTL must be purged.

8. Compression

A single DNS message may contain many domain names. Consider that when a name server sends a response containing multiple domain names, these names are usually in the same zone, or are related to the zone. Most of these names will have common elements to their names.

Consider a mail example of a client asking for an MX record for “xyzindustries.com”. The response to this client will contain, among other information, two records:

MX Record: An MX record that has “xyzindustries.com” as the Name of the record, and “mail.xyzindustries.com” in the RData field.

A Record: Assuming the name server knows the IP address of “mail.xyzindustries.com”, the Additional section will contain an A record that has “mail.xyzindustries.com” as the Name and its IP address in the RData field.

This is an example of name duplication. Considering other types of records in DNS messages, some string patterns may be repeated many times. This would be wasteful, since a large portion of these names is common.

To remove duplications, a special technique called message compression is used. Instead of a DNS name encoded as above using the combination of labels and label-lengths, a two-byte subfield is used to represent a pointer to another location in the message where the name can be found. The first two bits of this subfield are set to one (the value "11" in binary), and the remaining 14 bits contain an offset that species where the name can be found in the message, counting the first byte of the message (the first byte of the ID field) as 0.

Considering the above example. Suppose that in the DNS message, the RData field of the MX record, containing "mail.xyzindustries.com", begins at byte 47. Thus, in the second instance, where "mail.xyzindustries.com" shows up in the Name field of the A record, we can put two "1" bits, followed by the number 47 encoded in binary. So this would be the 16-bit binary pattern "11000000 00101111", or two numeric byte values "192" and "47". This second instance now takes 2 bytes instead of duplicating the 24 bytes needed for the first instance of the name.

How does a device differentiate a pointer from a "real" name? This is the reason that "11" is used at the start of the field. Doing this guarantees that the first byte of the pointer will always have a value of 192 or larger. Since labels are restricted to a length of 63 or less, when the host reads the first byte of a name, if it sees a value of 63 or less in a byte, it knows this is a "real" name; a value of 192 or more means it is a pointer.

9. 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.