The DNS protocol is well-documented online, however, we describe the salient pieces here for clarity. Note, however, that the official reference for the DNS protocol are the requests for comment (RFCs) that cover DNS (namely, RFC 1035). The RFC itself should be considered authoritative, most of the primer below is borrowed from the RFC itself. Note have been added in *italics* concerning many of the parts of the protocol that we will disregard in this project.

DNS is a hierarchical client-server protocol. Each domain (e.g., `neu.edu`, `microsoft.com`, etc) is served by one or more DNS servers, meaning requests for subdomains (e.g., `www.neu.edu`, and `research.microsoft.com`) are sent to these servers. Replies can also be cached by intermediate servers in order to improve performance; replies that come directly from the responsible DNS server are termed *authoratative* while replies that come from other DNS servers are *non-authoritative*.

1 DNS Packet Structure

All DNS packets have a structure that is

```
+---------------------+
|        Header       |
+---------------------+
|       Question      | the question for the name server
+---------------------+
|        Answer       | Answers to the question
+---------------------+
|      Authority      | Not used in this project
+---------------------+
|      Additional     | Not used in this project
+---------------------+
```

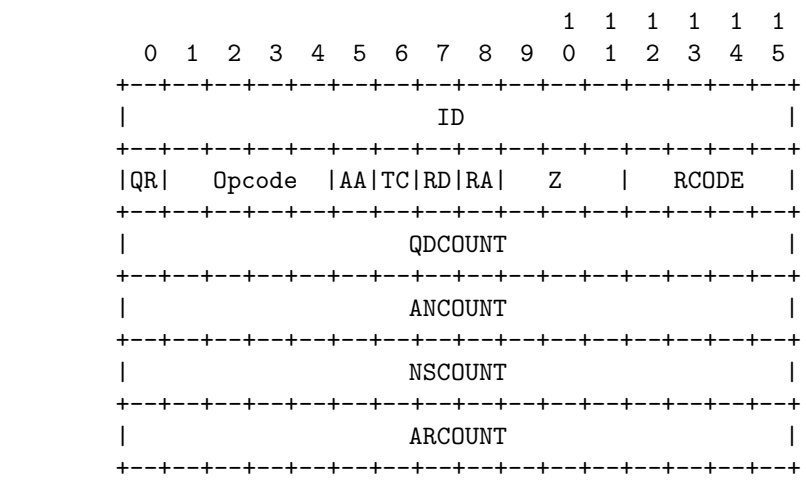
The header describes the type of packet and which fields are contained in the packet. Following the header are a number of questions, answers, authority records, and additional records. *For this project, we will be ignoring the authority and additional fields - your client program must accept packets with such fields, but must ignore them.*

Note that a response for a single question may contain multiple answers, such as if an address has multiple IP addresses, or if an address has a `CNAME` and an `A` record. Your client must process the entire answer section and report on each one of these records.

The format of each of these sections is described is the sections that follow.

2 DNS Headers

DNS packets have a *header* that is shown below. Note that requests and replies follow the same header format.



Where each of these fields is as described below:

`ID` A 16 bit identifier assigned by the program that generates any kind of query. This identifier is copied the corresponding reply and can be used by the requester to match up replies to outstanding queries. *You should select a new, random 16 bit number for each request.*

`QR` A one bit field that specifies whether this message is a query (0), or a response (1). *Obviously, you should use 0 for your requests, and expect to see a 1 in the response you receive.*

`OPCODE` A four bit field that specifies kind of query in this message. *You should use 0, representing a standard query.*

`AA` Authoritative Answer - this bit is only meaningful in responses, and specifies that the responding name server is an authority for the domain name in question section. *You should use this bit to report whether or not the response you receive is authoritative.*

`TC` TrunCation - specifies that this message was truncated. *For this project, you must exit and return an error if you receive a response that is truncated.*

`RD` Recursion Desired - this bit directs the name server to pursue the query recursively. *You should use 1, representing that you desire recursion.*

`RA` Recursion Available - this be is set or cleared in a response, and denotes whether recursive query support is available in the name server. Recursive query support is optional. *You must exit and return an error if you receive a response that indicates the server does not support recursion.*

`Z` Reserved for future use. *You must set this field to 0.*

`RCODE` Response code - this 4 bit field is set as part of responses. The values have the following interpretation:

`0` No error condition

`1` Format error - The name server was unable to interpret the query.

2 Server failure - The name server was unable to process this query due to a problem with the name server.

3 Name Error - Meaningful only for responses from an authoritative name server, this code signifies that the domain name referenced in the query does not exist.

4 Not Implemented - The name server does not support the requested kind of query.

5 Refused - The name server refuses to perform the specified operation for policy reasons.

You should set this field to 0, and should assert an error if you receive a response indicating an error condition. You should treat 3 differently, as this represents the case where a requested name doesn't exist.

`QDCOUNT` an unsigned 16 bit integer specifying the number of entries in the question section. *You should set this field to 1, indicating you have one question.*

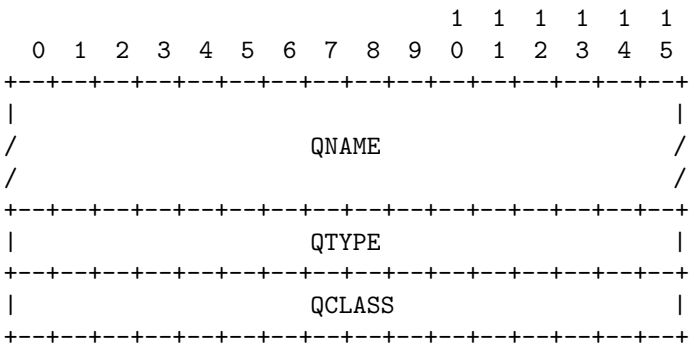
`ANCOUNT` an unsigned 16 bit integer specifying the number of resource records in the answer section. *You should set this field to 0, indicating you are not providing any answers.*

`NSCOUNT` an unsigned 16 bit integer specifying the number of name server resource records in the authority records section. *You should set this field to 0, and should ignore any response entries in this section.*

`ARCOUNT` an unsigned 16 bit integer specifying the number of resource records in the additional records section. *You should set this field to 0, and should ignore any response entries in this section.*

3 DNS Questions

A DNS question has the format



Where each of these fields is as described below:

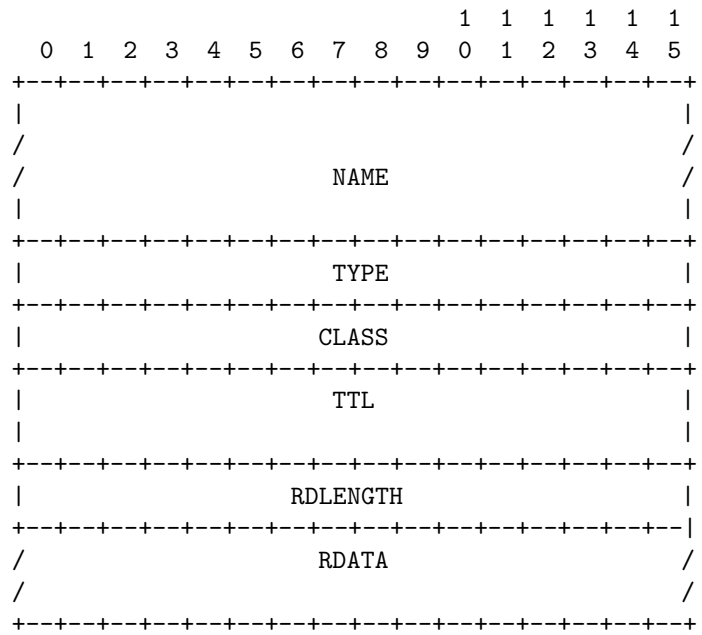
`QNAME` A domain name represented as a sequence of labels, where each label consists of a length octet followed by that number of octets. The domain name terminates with the zero length octet for the null label of the root. See the DNS Example query below.

`QTYPE` A two octet code which specifies the type of the query. *You should use* `0x0001` *for this project, representing* `A` *records (host addresses). If you are completing the graduate version of this project, you will also need to use* `0x000f` *for mail server (MX) records and* `0x0002` *for name servers (NS) records.*

`QCLASS` A two octet code that specifies the class of the query. *You should always use* `0x0001` *for this project, representing Internet addresses.*

4 DNS Answers

A DNS answer has the format



Where each of these fields is as described below:

`NAME` The domain name that was queried, in the same format as the `QNAME` in the questions.

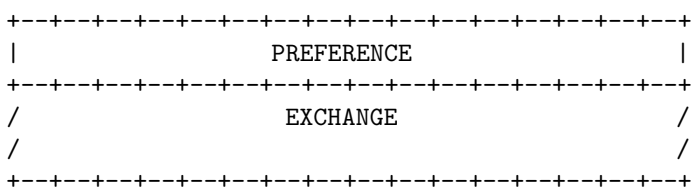
`TYPE` Two octets containing one of th type codes. This field specifies the meaning of the data in the `RDATA` field. *You should be prepared to interpret type* `0x0001` *(*`A` *record) and type* `0x0005` *(*`CNAME`*). If you are completing the graduate version of this project, you should also be prepared to accept type* `0x0002` *(name servers) and* `0x000f` *(mail servers).*

`CLASS` Two octets which specify the class of the data in the RDATA field. *You should expect* `0x0001` *for this project, representing Internet addresses.*

`TTL` The number of seconds the results can be cached.

`RDLENGTH` The length of the RDATA field.

`RDATA` The data of the response. The format is dependent on the `TYPE` field: if the `TYPE` is `0x0001` for `A` records, then this is the IP address (4 octets). If the type is `0x0005` for `CNAME`s, then this is the name of the alias. If the type is `0x0002` for name servers, then this is the name of the server. Finally if the type is `0x000f` for mail servers, the format is

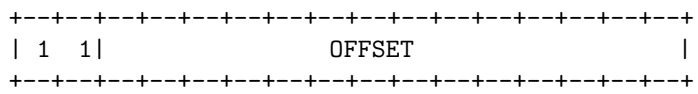


where `PREFERENCE` is a 16 bit integer which specifies the preference of this mail server, and `EXCHANGE` is a domain name stored in the same format as `QNAME`s. The latter two types are only relevant for the graduate version of this project.

5 DNS Packet Compression

In order to reduce the size of messages, the domain system utilizes a compression scheme which eliminates the repetition of domain names in the NAME, QNAME, and RDATA fields. In this scheme, an entire domain name or a list of labels at the end of a domain name is replaced with a pointer to a prior occurrence of the same name.

The pointer takes the form of a two octet sequence:



The first two bits are ones. This allows a pointer to be distinguished from a label, since the label must begin with two zero bits because labels are restricted to 63 octets or less. The `OFFSET` field specifies an offset from the start of the message (i.e., the first octet of the `ID` field in the domain header). A zero offset specifies the first byte of the `ID` field, etc.

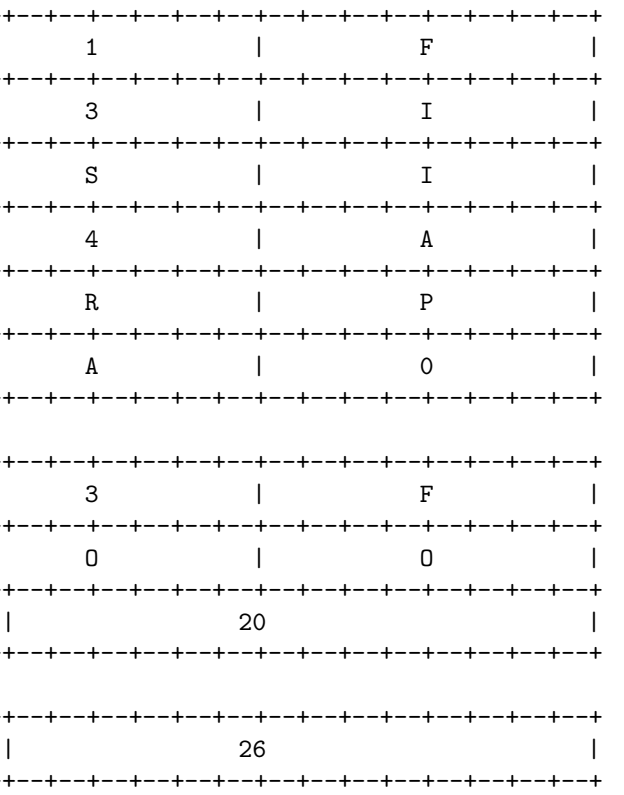
The compression scheme allows a domain name in a message to be represented as either:

- a sequence of labels ending in a zero octet
- a pointer
- a sequence of labels ending with a pointer

If a domain name is contained in a part of the message subject to a length field (such as the RDATA section of an RR), and compression is used, the length of the compressed name is used in the length calculation, rather than the length of the expanded name.

You are free to avoid using pointers in messages they generate, however, you are required to understand arriving messages that contain pointers.

For example, a datagram might need to use the domain names `F.ISI.ARPA`, `FOO.F.ISI.ARPA`, and `ARPA`. Ignoring the other fields of the message, these domain names might be represented as:



The domain name for `F.ISI.ARPA` is shown at offset 20. The domain name `FOO.F.ISI.ARPA` is shown at offset 40; this definition uses a pointer to concatenate a label for `FOO` to the previously defined `F.ISI.ARPA`. The domain name `ARPA` is defined at offset 64 using a pointer to the `ARPA` component of the name `F.ISI.ARPA` at 20; note that this pointer relies on `ARPA` being the last label in the string at 20.

6 Example DNS query

Shown below is the hexdump (gathered via `tcpdump` and `xxd`) for an A-record query for `www.northeastern.edu`.

```
0000000: db42 0100 0001 0000 0000 0000 0377 7777  .B...........www
0000010: 0c6e 6f72 7468 6561 7374 6572 6e03 6564  .northeastern.ed
0000020: 7500 0001 0001                           u.....
```

The header to this request should be read as:

Field	Sub-field	Value	Intrepretation
ID		0xdb42	Response should have ID 0xdb42
Flags		0x0100	
	QR	0	It's a query
	OPCODE	0	Standard query
	TC	0	Not truncated
	RD	1	Recursion requested
	RA	0	Not meaningful for query
	Z	0	Reserved
	RCODE	0	Not meaningful for query
QDCOUNT		0x0001	One question follows
ANCOUNT		0x0000	No answers follow
NSCOUNT		0x0000	No records follow
ARCOUNT		0x0000	No additional records follow

Finally, the single question that is included in this request can be parsed as

Data	Intrepretation
0x03	String of length 3 follows
0x777777	String is www
0x0c	String of length 12 follows
0x6e6f72746865617374726e	String is northeastern
0x03	String of length 3 follows
0x656475	String is edu
0x00	End of this name
0x0001	Query is a Type A query (host address)
0x0001	Query is class IN (Internet address)

7 Example DNS response

Shown below is the hexdump (gathered via tcpdump and xxd) for the query above.

```
0000000: db42 8180 0001 0001 0000 0000 0377 7777  .B...........www
0000010: 0c6e 6f72 7468 6561 7374 6572 6e03 6564  .northeastern.ed
0000020: 7500 0001 0001 c00c 0001 0001 0000 0258  u..............X
0000030: 0004 9b21 1144                           ...!.D
```

The header to this request should be read as:

Field	Sub-field	Value	Intrepretation
ID		0xdb42	Response to query ID 0xdb42
Flags		0x8180	
	QR	1	It's a response
	OPCODE	0	Standard query
	TC	0	Not truncated
	RD	1	Recursion requested
	RA	1	Server can do recursion
	Z	0	Reserved
	RCODE	0	Not meaningful for query
QDCOUNT		0x0001	One question follows
ANCOUNT		0x0001	One answer follows
NSCOUNT		0x0000	No records follow
ARCOUNT		0x0000	No additional records follow

The question text is interpreted exactly as in the request. Finally, the answer that is included in this request can be parsed as

Data	Intrepretation
0xc	Name is a pointer
0x00c	Pointer is to the name at offset 0x00c (0x03777777...)
0x0001	Answer is a Type A query (host address)
0x0001	Answer is class IN (Internet address)
0x00000258	Response is valid for 0x258 (600) seconds
0x0004	Address is 4 bytes long
0x9b211144	IP address is 0x9b211144 or 155.33.17.68