An Overview of DNS

ECSE 489 Telecom Network Lab

September 26, 2016

1 Introduction

The DNS protocol is defined in RFC 1035. There are plenty of websites and resources online describing its various aspects, but the RFC should be treated as authoritative. This document summarizes the main points you need to know to complete the lab exercise for ECSE 489. The document uses material directly from the RFC and is heavily inspired by a similar document prepared by Alan Mislove (Northeastern University). *Note:* Notes specific to the lab are indicated like this.

DNS uses a client-server architecture. The servers are organized into a hierarchy to implement a distributed database, mapping domain names to IP addresses and vice-versa. In other words, the records in the database help provide information about how to map a domain (like `mcgill.ca`, `www.mcgill.ca` or `mail.mcgill.ca`) to an IP address (like `132.216.177.160` or `132.216.46.252`). In RFC 1035, entries in the database are called *resource records* (RRs). Responses to queries can be cached by intermediate servers to improve system performance. A response that comes directly from the DNS server responsible for the domain name that was queried is called *authoritative*, and responses that come from other intermediate DNS servers (i.e., that have been cached) are *non-authoritative*.

2 DNS Packet Structure

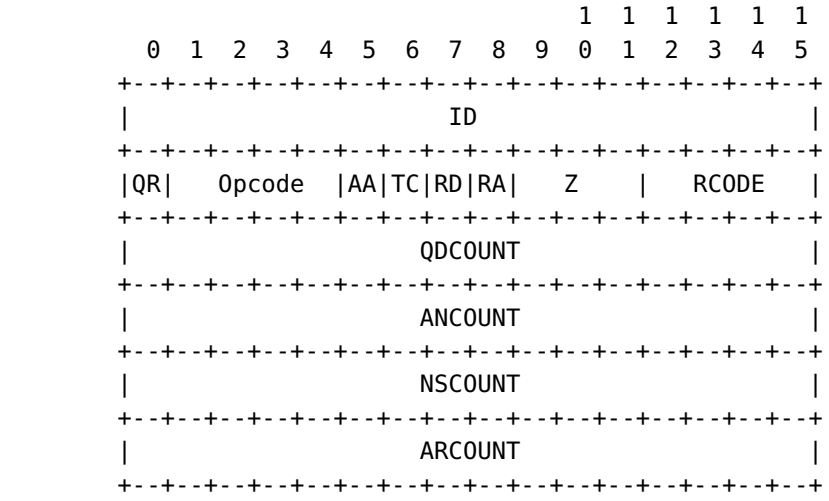
All DNS packets have following structure:

```
+---------------------+
|        Header       |
+---------------------+
|       Question      | the question for the name server
+---------------------+
|        Answer       | answers to the question
+---------------------+
|      Authority      | RRs pointing toward an authority (not used in this lab)
+---------------------+
|      Additional     | RRs holding additional information
+---------------------+
```

The header provides general information about the packet, including how many entries are in each section (more details below in Sec. 3). Following the header, the packet may contain a number of questions, answers, authority records, and additional records. *Note:* In this lab we will be ignoring records in the Authority section. Your program must accept packets that contain records in this section, but it should ignore them.

3 The DNS Packet Header

DNS follows the client-server application architecture, and it uses a state-less request-response protocol. Request packets and response packets have exactly the same header structure. The DNS header contains the following fields:



where the length and meaning of each field is described next.

`ID` is a 16-bit identifier assigned by the client issuing the query. The same ID is copied in the response packet and can be used by the client to match up responses to requests. *Note:* We recommend that your application use a new *random* 16-bit number for each request.

`QR` is a 1-bit field that specifies whether this message is a query (`0`) or a response (`1`).

`OPCODE` is a 4-bit field that specifies the kind of query in this message. *Note:* You should set this field to 0, representing a standard query.

`AA` is a bit that is only meaningful in response packets and indicates whether (`1`) or not (`0`) the name server is an authority for a domain name in the question section. *Note:* You should use this field to report whether or not the response you receive is authoritative.

`TC` indicates whether or not this message was truncated because it had a length greater than that permitted by the transmission channel.

`RD` is a bit set in the request message to indicate whether or not the client would like the name server to pursue the query recursively. *Note:* Your program should set this bit to 1 to indicate that you desire recursion.

`RA` is a bit set or cleared by the server in a response message to indicate whether or not recursive queries are supported. *Note:* Your program should print an error message if the server does not support recursive queries. If the server provided an answer then it should still be printed also.

`Z` is a 3-bit field reserved for future use. *Note:* Your program should set this to 0.

`RCODE` is a 4-bit field, only meaningful in response messages, containing the response code; it can take the following values:

- `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 requested operation for policy reasons

Note: You should set this field to 0 in your request messages. If the response message has a non-zero `RCODE` value, your program should output an appropriate error message. Make sure to handle code 3 correctly (NOTFOUND instead of ERROR).

`QDCOUNT` is an unsigned 16-bit integer specifying the number of entries in the question section. *Note:* This will always be 1 for your program.

`ANCOUNT` is an unsigned 16-bit integer specifying the number of resource records in the answer section.

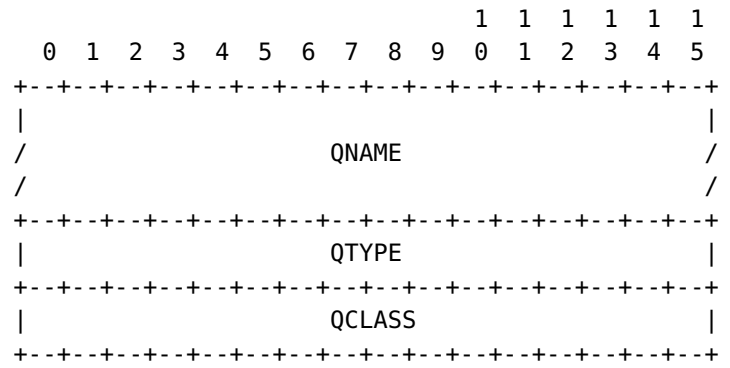
`NSCOUNT` is an unsigned 16-bit integer specifying the number of name server resource records in the Authority section. *Note:* Your program can ignore any response entries in this section.

`ARCOUNT` is an unsigned 16-bit integer specifying the number of resource records in the Additional records section.

4 DNS Questions

In this lab, your client program will always send queries for domain names and receive responses containing information about the domain (where the contents of the response depends on the type of query). A domain name is a string consisting of a sequence of *labels* separated by dots. For example, `mcgill.ca` consists of two labels and `www.mcgill.ca` has three. The standard defining domain names (RFC 1034) restricts labels to be at most 63 octets long.[1]

A DNS question has the format:



where the meaning and length of each field is as follows.

`QNAME` is a domain name represented by a sequence of labels, where each label begins with a length octet followed by that number of octets. The domain name terminates with the zero-length octet, representing the null label of the root. *Note:* See the examples below and in Sec. 7.

`QTYPE` is a 16-bit code specifying the type of query. *Note:* The three types relevant to this lab are:

`0x0001` for a type-A query (host address)

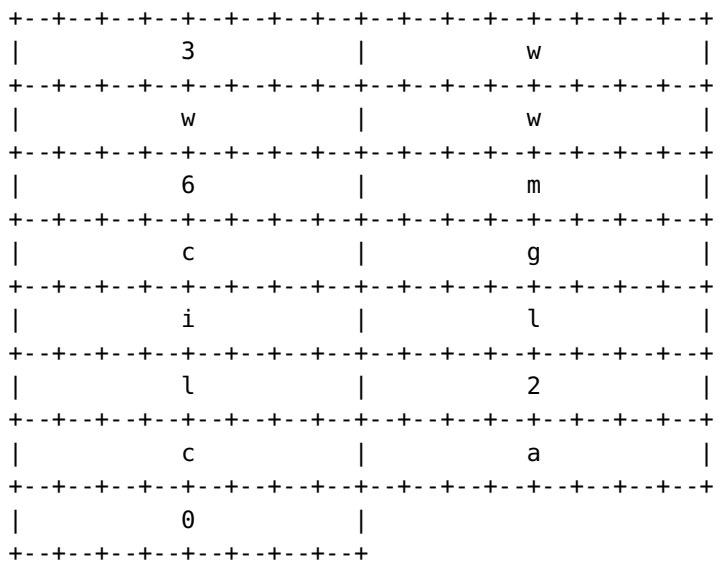
[1] An *octet* is a unit of eight bits. Historically, in the context of computer network protocols, the word *octet* is used instead of *byte* because the definition of a byte was platform-dependent. Nowadays the two words are used interchangeably.

`0x0002` for a type-NS query (name server)

`0x000f` for a type-MX query (mail server)

`QCLASS` is a 16-bit code specifying the class of the query. *Note:* You should always use `0x0001` in this field, representing an Internet address.[2]

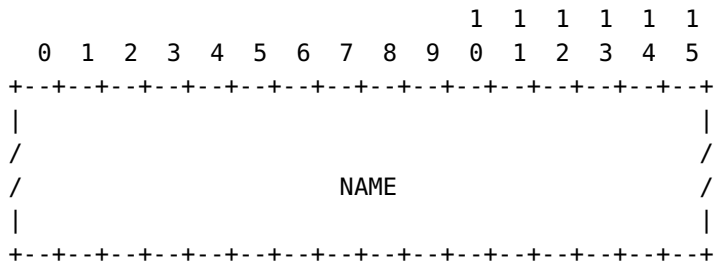
In fields such as `QNAME` within a DNS packet, domain names are represented as lists of labels. Since labels may have varying length, each label is preceded by a single byte giving the number of ASCII characters used in the label, and then each character is coded using 8-bit ASCII. To signal the end of a domain name, one last byte is written with value 0. (Think of this as a final label with length 0, signalling the end of the domain name.) For example, the domain name `www.mcgill.ca` has three labels (`www`, `mcgill`, and `ca`) with lengths 3, 6, and 2. The `QNAME` representation of this would use a total of 15 bytes and would be given by:



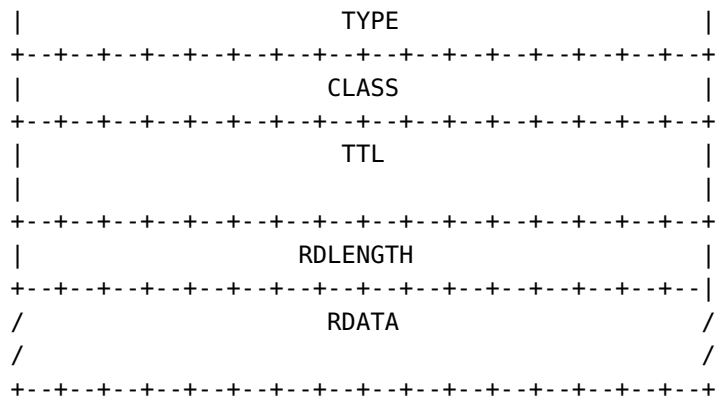
where, in the diagram above, numbers are replaced by the corresponding 8-bit unsigned integer binary representation (e.g., 3 above is `0x03`) and characters are replaced by their 8-bit ASCII "printable character" representation (e.g., 'w' is `0x77`). Of course, if a domain name contains numbers, then those numbers are represented using the ASCII representation. For example, the hexadecimal representation of a `QNAME` field for the domain name `163.com` would be `0x03 31 36 33 03 63 6f 6d`.

5 DNS Answers

Resource records in the Answer, Additional, and Authority sections have the format:



[2] At the time the DNS protocol was defined, there were other naming conventions in use (see the RFC for more). Today these are effectively obsolete, and this field will nearly always have the value `0x0001` corresponding to an `IN` or Internet address.



where the fields have lengths and meanings as follows.

`NAME` is a domain name to which this record pertains.

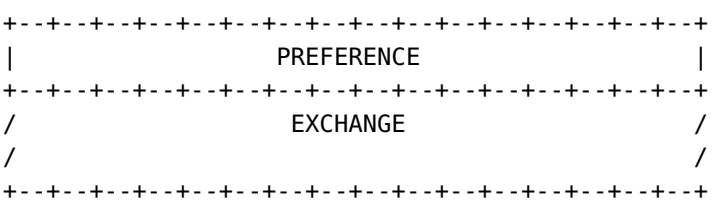
`TYPE` is a 16-bit code specifying the meaning of the data in the `RDATA` field. *Note:* Your program should be able to handle resource records with the same three values that `QTYPE` can take, and it should also handle type `0x0005` corresponding to `CNAME` records.

`CLASS` is a 16-bit code with meaning similar to that of `QCODE` in question packets. *Note:* You should expect this field to be `0x0001` and display an error if a different value is encountered.

`TTL` is an unsigned 32-bit integer that specifies the number of seconds that this record may be cached before it should be discarded (invalidated). If the response contains a resource record with `TTL` equal to zero, this should be interpreted that the response is only valid for the present query and should not be cached.

`RDLENGTH` is an unsigned 16-bit integer that specifies the length (in octets) of the `RDATA` field.

`RDATA` a variable length sequence of octets that describes the resource. The format and meaning of these octets depends on the `TYPE` of the record. If `TYPE` is `0x0001`, for an `A` (IP address) record, then `RDATA` is the IP address (four octets). If the `TYPE` is `0x0002`, for a `NS` (name server) record, then this is the name of the server specified using the same format as the `QNAME` field. If the `TYPE` is `0x005`, for `CNAME` records, then this is the name of the alias. If the type is `0x000f` for `MX` (mail server) records, then `RDATA` has the format



where

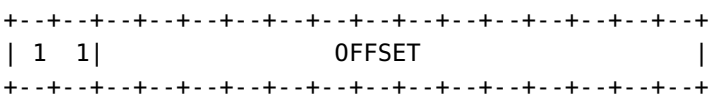
`PREFERENCE` is a 16-bit unsigned integer specifying the preference given to this resource record among others at the same owner (lower values are preferred);

`EXCHANGE` is the domain name of a mail server for this owner, using the same format as the `QNAME` field.

6 DNS Packet Compression

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

The pointer takes the form of a 16-bit sequence,



beginning with two 1s, followed by a 14-bit offset. Setting the first two bits to `1` allows pointers to be distinguished from labels since labels are restricted to be 63 octets long or less. The `OFFSET` field specifies an offset (in octets) from the start of the message to the beginning of a label (really, to the leading octet giving the length of the label); i.e., a zero offset corresponds to the first byte of the `ID` field in the header.

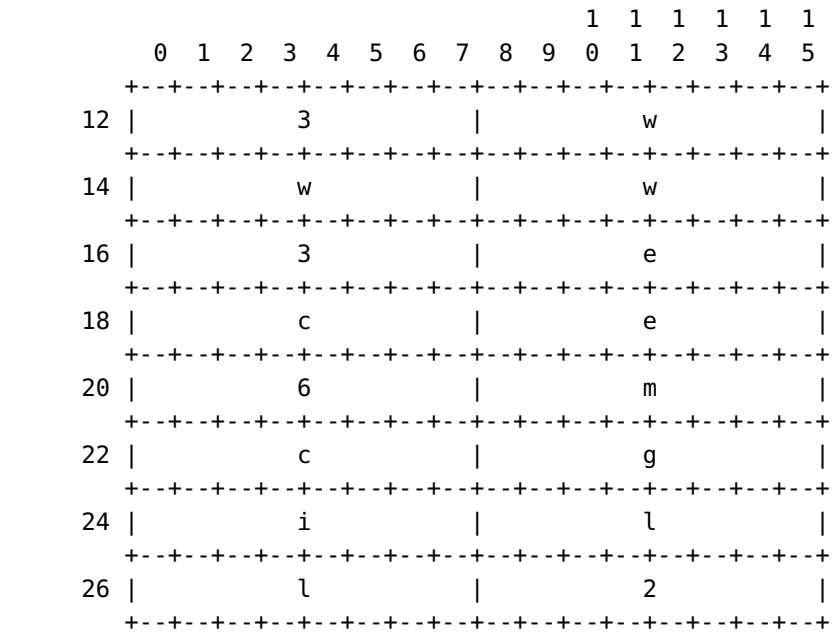
The compression scheme allows a domain name in a message to be represented as either:

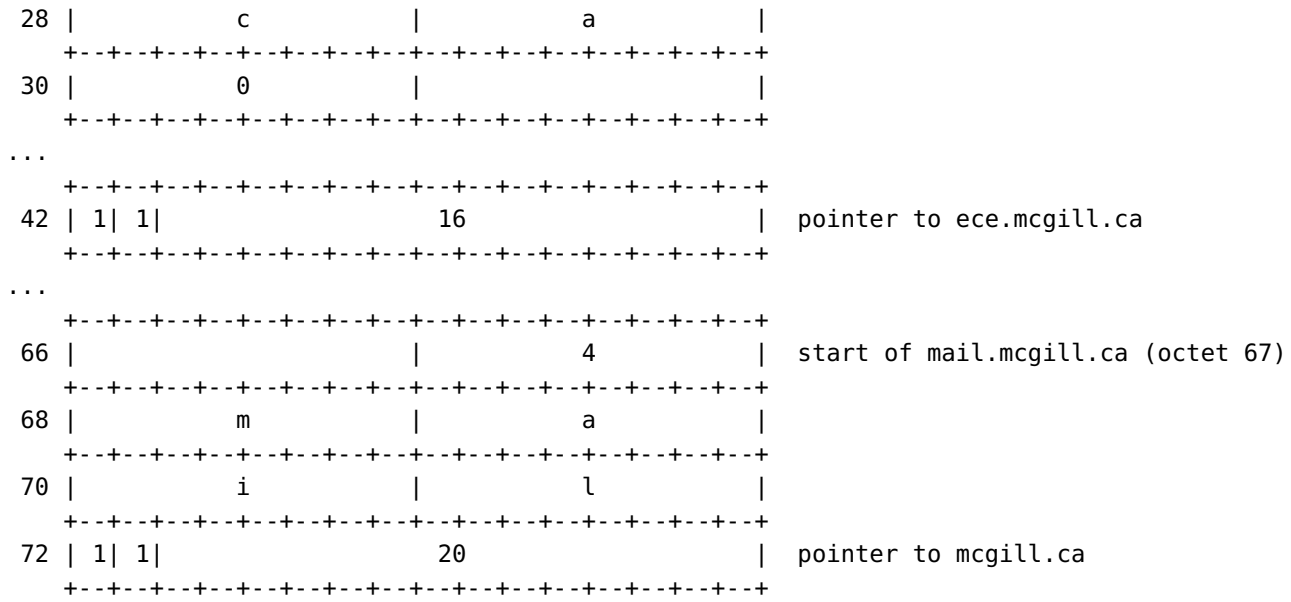
- a sequence of labels ending with a zero octet;
- a pointer;
- a sequence of labels ending with a pointer.

If a domain name is contained in a part of the message subject to a length field (e.g., the `RDATA` field of a resource record) and compression is used, then the length of the compressed name is used in the length calculation rather than the length of the uncompressed/expanded name.

Note: Your program is not required to use pointers in request messages that it generates. However, your program must be able to correctly interpret response messages that use DNS packet compression and contain pointers.

Let's illustrate packet compression using an example. Suppose that a response packet contains the domain names `www.ece.mcgill.ca`, `ece.mcgill.ca`, and `mail.mcgill.ca` (in that order). Ignoring the other fields of the message, these domain names might be represented in the following manner (where the number on the left is the hypothetical offset into the packet):





where the encoding of `www.ece.mcgill.ca` begins at offset 12 and is 19 octets long, the encoding of `ece.mcgill.ca` begins at offset 42 and is 2 octets long (just a pointer), and the encoding of `mail.mcgill.ca` begins at offset 67 and is 7 octets long.

7 Example DNS Query

Shown below is a hexdump for an A-record query for `www.mcgill.ca`.[3]

```
0000   82 7a 01 00 00 01 00 00 00 00 00 00 03 77 77 77  .z...........www
0010   06 6d 63 67 69 6c 6c 02 63 61 00 00 01 00 01     .mcgill.ca.....
```

The header to this request should be parsed as follows.

Field	Sub-field	Value	Interpretation
ID		0x827a	Response should have ID 0x827a
FLAGS		0x0100	
	QR	0	This is a query
	OPCODE	0000	A standard query
	AA	0	Not meaningful for a query
	TC	0	Not truncated
	RD	1	Recursion requested
	RA	0	Not meaningful for a query
	Z	000	Reserved, set to zero
	RCODE	0000	Not meaningful for a query
QDCOUNT		0x0001	One question
ANCOUNT		0x0000	No records in the Answer section
NSCOUNT		0x0000	No records in the Authoritative section
ARCOUNT		0x0000	No records in the Additional section

Then, the single question in this request packet should be parsed as follows.

[3] Note that non-printable ASCII characters are replaced with dots in the ASCII translation on the right.

Data	Interpretation
0x03	A label with three characters follows
0x77 77 77	The label is www
0x06	A label with six characters follows
0x6d 63 67 69 6c 6c	The label is mcgill
0x02	A label with two characters follows
0x63 61	The label is ca
0x00	A zero length label marking the end of the QNAME field
0x0001	The QTYPE of this question is A (IP address)
0x0001	The QCLASS of this question is IN (Internet address)

8 Example DNS Response

Next, shown below, is the hexdump for the response to the query above.

```
0000   82 7a 81 00 00 01 00 01 00 00 00 00 03 77 77 77  .z...........www
0010   06 6d 63 67 69 6c 6c 02 63 61 00 00 01 00 01 c0  .mcgill.ca......
0020   0c 00 01 00 01 00 00 04 13 00 04 84 d8 b1 a0     ...............
```

The header to this request should be parsed as follows.

Field	Sub-field	Value	Interpretation
ID		0x827a	This matches the query with ID 0x827a
FLAGS		0x0100	
	QR	1	This is a response
	OPCODE	0000	A standard query
	AA	0	Non-authoritative response
	TC	0	Not truncated
	RD	1	Recursion requested
	RA	0	The server can't do recursive queries
	Z	000	Reserved, ignore
	RCODE	0000	No error
QDCOUNT		0x0001	One question
ANCOUNT		0x0001	One answer
NSCOUNT		0x0000	No records in the Authoritative section
ARCOUNT		0x0000	No records in the Additional section

The next 19 bytes after the header repeat the question exactly as in the original query packet and can be parsed in the same way.

The answer record begins at the 32nd octet and should be parsed as follows.

Data	Interpretation
0xc00c	A pointer to the octet with offset 12; it expands out to www.mcgill.ca
0x0001	Type-A response (host address)
0x0001	Response class IN (Internet)
0x00000413	TTL is 413 seconds
0x0004	RDLENGTH is 4 octets (i.e., 32 bits, for an IPv4 address)
0x84d8b1a0	The IP address is 132.216.177.160