1 UDP packet structure

The UDP packet starts with an 8 octet (byte) header. (see (Postel, 1980, RFC 768))

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Source port	Destination port
Length	Checksum
Some data	

Length This field specifies the length in bytes of the UDP header and UDP data. The minimum length is 8 bytes, the length of the header.

2 DNS Data

The UDP data field contains the DNS messages, requests, and replies.
The structure is given in (Mockapetris, 1987, RFC 1035 section 4)

2.1 Format

All communications inside of the domain protocol are carried in a single format called a message. The top level format of message is divided into 5 sections:

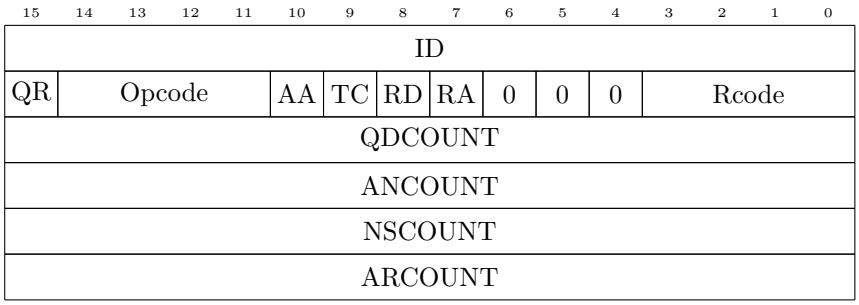
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Header	
Question	} the question for the name server
Answer	} RRs answering the question
Authority	} RRs pointing toward an authority
Additional	} RRs holding additional information

The header section is always present. The header includes fields that specify which of the remaining sections are present, and also specify whether the message is a query or a response, a standard query or some other opcode, etc.

2.1.1 Header section format

The header fields and flags are described in (Mockapetris, 1987, RFC 1035 4.1.1)

QR A 1-bit flag that indicates if the message is a query (0) or a response (1)



QDCOUNT an unsigned 16 bit integer specifying the number of entries in the question section.

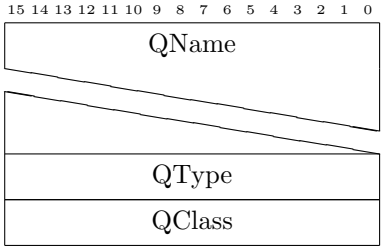
ANCOUNT an unsigned 16 bit integer specifying the number of resource records in the answer section.

NSCOUNT an unsigned 16 bit integer specifying the number of name server resource records in the authority records section.

ARCOUNT an unsigned 16 bit integer specifying the number of resource records in the additional records section.

2.1.2 The Question section format

The question section contains QDCOUNT entries, usually 1



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. Note that this field may be an odd number of octets; no padding is used.

QType a two octet code which specifies the type of the query. The values for this field include all codes valid for a TYPE field, together with some more general codes which can match more than one type of RR.

QClass a two octet code that specifies the class of the query. For example, the QClass field is IN for the Internet.

The name is formatted as a sequence of labels `unn.ac.uk` is split into 4 labels; `unn`, `ac`, `uk`, and the zero length null label for the root. It occupies 11 octets;

15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
3	‘u’
‘n’	‘n’
2	‘a’
‘c’	2
‘u’	‘k’
0	

The following `Qtype` label follows *immediately* and so is not aligned to 16-bit boundaries.

3 C programming interface

The libpcap library calls a user defined function for decoding packet captures

```
void packetProcessor( u_char *userData,
                      const struct pcap_pkthdr* pkthdr, const u_char* packet)
{
}
```

The UDP header is found by

```
  udpHeader = (struct udphdr*)(packet + sizeof(struct ether_header) + sizeof(struct ip));
```

Now `udpHeader` is now a pointer to the `udphdr` structure. This structure is defined in `udp.h`

```
struct udphdr {
 uint16_t uh_sport;
 uint16_t uh_dport;
 uint16_t uh_ulen;
 uint16_t uh_sum;
};
```

The address of the UDP payload can be found using

```
u_char *udpPayload = (u_char *)(packet
                                + sizeof(struct ether_header)
                                + sizeof(struct ip)
                                + sizeof(struct udphdr)
                               );
```

We can't just add to the pointer to the header (`udpHeader + sizeof(struct udphdr)`) because `const struct udphdr* udpHeader;` is a pointer to a structure, and so adding to it increments by (`sizeof(struct udphdr)`) bytes.

3.1 DNS data

Assume that `dnsbase` has been set to the address of the UDP data found from `udpPayload` above.

Header The DNS header is a fixed size of 32 octets. It can be given by

```
struct dnshdr {
  uint16_t id;
  uint16_t flags;
  uint16_t QDcount;
  uint16_t ANcount;
  uint16_t NScount;
  uint16_t ARcount;
};
```

We can set up a structure to use the data by:

```
struct dnshdr *DNShdr = (struct dnshdr *)dnsbase;
unsigned int querycount = DNShdr->QDcount;
```

Query — Name The base address of the query is 32 octets on from the base address.

```
uint8_t *query = dnsbase +32;
```

We are working with pointer to 8-bit datatypes, so pointer addition works the right way for us.

The base of the name is the same address as the pointer to the query.

> The name ends with a zero-valued byte. It could be used as a string. Except that it has embedded lengths in it, which are not printable ascii values.

The best solution is to use a loop, and string copy operations.
In the code following remember that the idiom

```
len = *query++;
```

means, assign the value pointed to by `query` to the variable `len`, and then increment `query` by 1.

The label lengths are 8-bit values, with the top 2 bits clear,

```
/* 6 bits for length,
   max label is 63 characters
   plus \0 for end of C string
*/
char fqdn[256] ;
unit8_t len;
char label[64];
fqdn[0] = '\0'; /* ensure starting with empty string */
for( len = *query++ ; len>= ; len=*query++ ) {
    strncpy( label, query, len );
    label[len] = '\0'; /* puts zero byte at end */
    query += len; /* move pointer to end of string
                     location of next length
                  */
        /* Do something with the label... */
        strcat(fqdn, label);
        strcat(fqdn, ".");
}
```

An alternative is to overwrite the length values with the ascii for '.' to create a domain name.

```
/* 6 bits for length,
   max label is 63 characters
   plus \0 for end of C string
*/
char *label = (char *)query;
unit8_t len;
char fqdn[64];
for( len = *query ; len>= ; len=*query ) {
        *query = '.';
        query += len+1; /* query at next len value */
}
strcpy(fqdn, label);
```

This ends up starting the name with a `.` which may be the wrong thing.

References

P. Mockapetris. *RFC 1035 Domain Names - Implementation and Specification*. Internet Engineering Task Force, November 1987. URL `http://tools.ietf.org/html/rfc1035`.

J. Postel. User Datagram Protocol. RFC 768 (Standard), August 1980. URL `http://www.ietf.org/rfc/rfc768.txt`.