CENG393 Computer Networks
Labwork 4

1 IPv4 Packet Structure

Every IPv4 packet has a header that follows a standart definition and has the following fields:

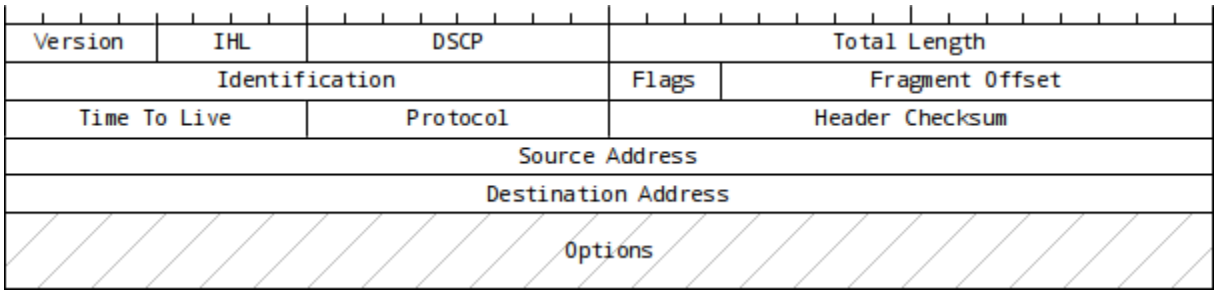

Figure 1: IPv4 Packet Header

- **Version:** Version of protocol. For IPv4, its value is 4.
- **IHL:** Header length. The minimum value is 5 (32-bit words).
- **DSCP:** Differentiated Services.
- **Total Length:** Length of the whole packet.
- **Identification:** Numerical identification of a packet.
- **Flags:** Don't Fragment (DF) and More Fragments (MF).
- **Fragment Offset:** Offset of a fragment relative to the beginning of the complete packet.
- **Time To Live:** Specifies how many routers this packet can traverse through until it is expired.
- **Protocol:** Used to define the upper layer protocol used in the data.
- **Header Checksum:** Error checking field for the header.
- **Source Address:** IP address of source.
- **Destination Address:** IP address of destination.
- **Options:** Used to modify routing behaviors, experimental features, etc.

2 IPv4 Header Checksum

The checksum field in the header is used for verifying if the header has been transmitted without problems or not. It is calculated by the source device before transmission and it is used for verifying by destination device.

2.1 Calculation

In order to calculate the checksum, all information except the checksum must already be available in the header. For example lets assume the following sequence is a 20 bytes long IPv4 header:

```
45 00 00 30 0f 41 40 00 80 06 00 00 91 fe a0 ed 41 d0 e4 df
```

1. Calculate sum of every 16-bit words in header.

   ```
   = 4500 + 0030 + 0f41 + 4000 + 8006 + 0000 + 91fe + a0ed + 41d0 + e4df
   = 00036e11
   ```

2. If the result is longer than 16 bits, divide it into 16-bit words and add them together. Repeat this step as many times as necessary.

   ```
   = 0003 + 6e11
   = 6e14
   ```

3. Calculate 1's complement.

   ```
   = ~6e14
   = 91eb
   ```

After this calculation, the result is written in its place in the header:

```
45 00 00 30 0f 41 40 00 80 06 91 eb 91 fe a0 ed 41 d0 e4 df
```

2.2 Verification

When the next device in the network acquires this packet, it first checks the header if it has been received correctly or not. In order to do so, this device follows the same calculation algorithm, but this time the data used by the device already contains the checksum:

1. Calculate sum of every 16-bit words in header.

   ```
   = 4500 + 0030 + 0f41 + 4000 + 8006 + 91eb + 91fe + a0ed + 41d0 + e4df
   = 0003fffc
   ```

2. If the result is longer than 16 bits, divide it into 16-bit words and add them together. Repeat this step as many times as necessary.

   ```
   = 0003 + fffc
   = ffff
   ```

3. Calculate 1's complement.

   ```
   = ~ffff
   = 0000
   ```

If the result is zero, that means the data in the header can be assumed as correctly received.

3 Address Resolution Protocol Packet Structure

Address Resolution Protocol (ARP) is a protocol used for obtaining MAC address of a network device with a given IP address. It's packet structure is shown below in Figure 2:

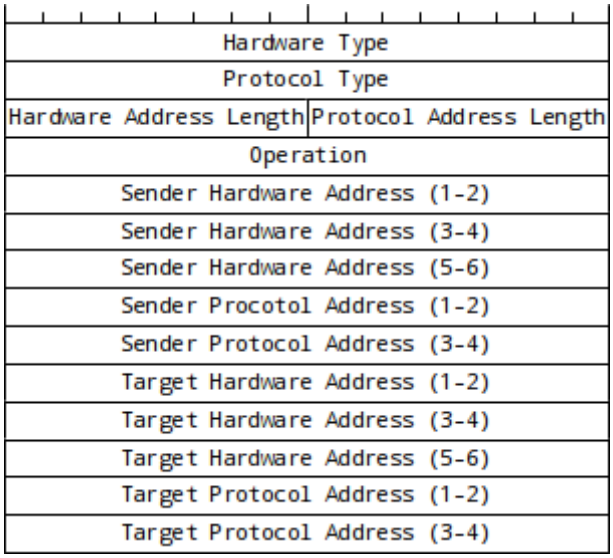

Figure 2: ARP Packet

- **Hardware Type:** Ethernet: 1
- **Protocol Type:** IPv4: 0x0800
- **Hardware Address Length:** MAC: 6
- **Protocol Address Length:** IPv4: 4
- **Operation:** 1: Request, 2: Reply
- **Sender Hardware Address:** Source MAC address
- **Sender Protocol Address:** Source IPv4 address
- **Target Hardware Address:** Destination MAC address
- **Target Protocol Address:** Destination IPv4 address

4 Exercises

1. Download the following file: `http://ceng393.cankaya.edu.tr/uploads/files/file/http.txt`. Load this file in Wireshark and study the sample IPv4 packets.
2. In Linux command line interface, **ping** command can be used to send ICMP ECHO_REQUEST messages between hosts and **traceroute** command can be used to print the route to a specified destination. Choose any host you want and use ping and traceroute commands to observe the output. Then using **man** documentation, find out which options are used for changing TTL value and use ping and traceroute commands again with lower TTL values. What difference do you see now?

3. Complete the following C program to calculate checksum for the header given in the array:

```
#include <stdio.h>
int main() {
  int i, byte_grp, header[20] = { 0x45, 0x00, 0x00, 0x30, 0x0f, 0x41, 0x40,
      0x00, 0x80, 0x06, 0x00, 0x00, 0x91, 0xfe, 0xa0, 0xed, 0x41, 0xd0,
      0xe4, 0xdf };

  for (i = 0; i < 10; i++) {
    byte_grp = (header[2 * i] << 8) + header[2 * i + 1];
    printf("%04x ", byte_grp);
  }
  return 0;
}
```

   In order to do bitwise calculations, you may use the operators demonstrated in Appendix A below.

4. Download the following file: `http://ceng393.cankaya.edu.tr/uploads/files/file/arpsamples.txt`. Load this file in Wireshark and study the sample ARP packets.

5. In Linux command line interface, **arp** command can be used to view the computer's ARP table. This table is a cached list of previously established IPv4 connections. Ping several devices in your own network and observe the changes in this table.

6. Using **man** documentation, find out which options are used for manually adding and deleting entries in the ARP table. Using these options, remove a previously available entry in the ARP table, and then manually add it back with incorrect MAC address. Ping the same computer now and observe what happens.

Appendices

A Boolean Operators in C

```
&  : Bitwise AND                |  : Bitwise OR
^  : Bitwise XOR                ~  : Bitwise NOT
>> : Bitwise Shift Right        << : Bitwise Shift Left

Example:
int a  = 0x6D, b = 60; // both are 32-bit variables

a & b  = 0x2C (44)               a | b = 0x7D (125)
a ^ b  = 0x51 (81)                  ~a = 0xFFFFFF92
a << 1 = 0xDA (218)              b >> 2 = 0x0F (15)
```