Lab-5

Aim: IPv6, Address Abbreviation, Fixed Header, Extension Headers, Comparison between IPv4 and IPv6, Implementing Checksum in C for error detection.

IPv6:

IPv6 is the next planned version of the IP address system. IPv6 uses 128-bit addresses (16-bytes), which increases the number of possible addresses by an exponential amount. Because IPv6 allows for substantially more IP addresses than IPv4, the addresses themselves are more complex. They are typically written in this format:

hhhh:hhhh:hhhh:hhhh:hhhh:hhhh:hhhh:hhhh

Each "hhhh" section consists of a four-digit hexadecimal number.

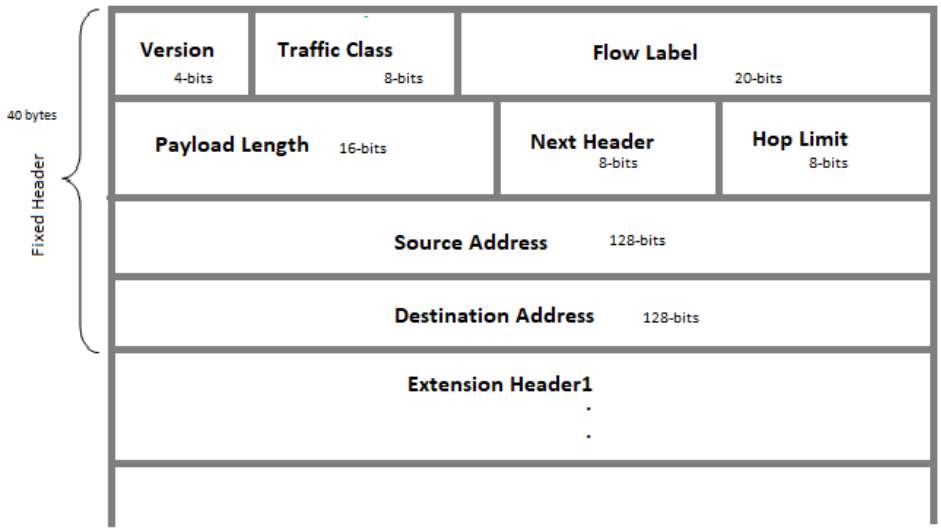
For example, 2001:0db8:0000:0200:0000:0000:0000:0007. Now this can be written in abbreviated form like 2001:db8:0:200:0:0:0:7.

Address Abbreviation:

IPv6 addresses often contain consecutive hexadecimal fields of zeros. To simplify address entry, we can use two colons (::) to represent the consecutive fields of zeros when typing the IPv6 address. For example if our IPv6 is 2001:db8:0:200:0:0:0:7, it will take the additional zeros and replace them with ":" like this: 2001:db8:0:200::7

Another example, if our IPv6 is 2001:db8:0:0:0:0:0:7, we can write like this: 2001:db8::7.

Fixed Header (40-bytes):



Version (4-bits): It specifies the Internet Protocol version. Here it contains 0110.

Traffic Class (8-bits): It is similar to Service Field in IPv4. It helps to handle the traffic on the basis of class. From 8-bits, 4-bits are fixed so in remaining 4-bits i.e. 0-15, 0-7 is for uncontrolled traffic class and 8-15 is controlled traffic class.

Flow label (20-bits): A source can use this field to label those packets for which the source requests special handling by the IPv6 routers. For example, a source can request non-default quality of service or real-time service or minimum delay, etc. For default router handling, the Flow Label field is set to 0.

Payload Length (16-bits): The Payload Length field indicates the length of the IPv6 payload. The Payload Length field includes the extension headers and the upper-layer PDU. With 16 bits, an IPv6 payload of up to 65,535 bytes can be indicated. For payload lengths greater than 65,535 bytes, the Payload Length field is set to 0 and the Jumbo Payload option is used in the Hop-by-Hop Options extension header.

Next Header (8-bits): It is similar to Protocol Field in IPv4. This field is used to indicate either the type of Extension Header, or if the Extension Header is not present then it indicates the protocols contained with Upper Layer packet such as TCP, UDP.

Hop Limit (8-bits): This field is used to stop packet to loop in the network infinitely. This is same as TTL in IPv4. The value of Hop Limit field is decremented by 1 as it passes a link (router/hop). When the field reaches 0 the packet is discarded.

Source Address (128-bits): Source Address is 128-bit IPv6 address of the original source of the packet.

Destination Address (128-bits): Destination Address field indicates the IPv6 address of the final destination.

Extension Headers:

In IPv6, the Fixed Header contains only that much information which is necessary, avoiding those information which is either not required or is rarely used. All such information is put between the Fixed Header and the Upper layer header in the form of Extension Headers. Each Extension Header is identified by a distinct value. When Extension Headers are used, IPv6 Fixed Header's Next Header field points to the first Extension Header. If there is one more Extension Header, then the first Extension Header's 'Next-Header' field points to the second one, and so on. The last Extension Header's 'Next-Header' field points to the Upper Layer Header. Thus, all the headers points to the next one in a linked list manner.

Hop-by-Hop option: It specifies delivery parameters at each hop on the path to the destination host. The Pad1 byte is the only option without a length and value. It provides 1 byte of padding. The PadN option is used when 2 or more bytes of padding are required. For 2 bytes of padding, the length of this option would be 0 and the option would consist of just the type field and the length field. For 3 bytes of padding, the length would be 1, and 1 byte of 0 would follow this length. The jumbo payload length option provides a datagram length of 32 bits and is used when the 16-bit payload length field in is inadequate.

Source Routing: The Routing header is used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet's destination. This function is very similar to IPv4's Timestamp, Strict routing, Loose Source and Record Route option.

Fragmentation: It specifies how to perform IPv6 fragmentation and reassembly services. A source node uses the fragment extension header to tell the destination node the size of the packet that was fragmented so that the destination node can reassemble the packet.

Authentication: It provides authentication, data integrity, etc.

Encrypted Security Payload: It provides data confidentiality, data authentication, etc.

Destination option: It identifies the host device, or interface on a node, to which the IPv6 packet is to be sent.

Comparison between IPv4 and IPv6 packet headers:

1. The header length field is eliminated in IPv6 because the length of the header is fixed (40-bytes) in this version.
2. The service type field is eliminated in IPv6. The priority and flow label fields together take over the function of the service type field.
3. The total length field is eliminated in IPv6 and replaced by the payload length field.
4. The identification, flag, and offset fields are eliminated from the base header in IPv6. They are included in the fragmentation extension header.
5. The TTL field of IPv4 is called hop limit in IPv6.
6. The protocol field of IPv4 is replaced by the next header field in IPv6.
7. The header checksum is eliminated because the checksum is provided by upper-layer protocols; it is therefore not needed at this level.
8. The option fields in IPv4 are implemented as extension headers in IPv6.

Comparison between IPv4 options and IPv6 extension headers:

1. The no-operation and end-of-option options in IPv4 are replaced by Pad1 and PadN options in IPv6.
2. The record route option is not implemented in IPv6 because it was not used.
3. The timestamp option is not implemented because it was not used.
4. The source route option is called the source route extension header in IPv6.
5. The fragmentation fields in the base header section of IPv4 have moved to the fragmentation extension header in IPv6.
6. The authentication extension header is new in IPv6.
7. The encrypted security payload extension header is new in IPv6.

Implementing Checksum in C for error detection (for fixed frame):

Code:

```
#include <stdio.h>

#include <stdlib.h>
```

```
#define SIZE 8


int main(void)
{
   FILE *fptr1, *fptr2;
   char c;
   char buff[SIZE];
   int i=0;
   int frame_count = 0;
   char XOR;


   printf("\nCreating Frames and Doing checksum at sender's side.....\n\n");


   if ((fptr1 = fopen("D:\\d drive\\Operating Systems\\SEM-6\\CN\\Lab\\Lab-
5\\Original.txt", "r")) == NULL)
   {
      printf("Error! Reader opening file\n");
      return (1);
   }
   if ((fptr2 = fopen("D:\\d drive\\Operating Systems\\SEM-6\\CN\\Lab\\Lab-
5\\ChecksumFile.txt", "w")) == NULL)
   {
      printf("Error! Writer/ChecksumFile opening file\n");
      return (1);
   }
   while (1)
   {
      c = fgetc(fptr1);
      buff[i]= c; // reading the file
      if(i==0)
```

```
{
    XOR = c;
}
if (c == EOF)
{
    if(i!=0)
   {
    XOR = XOR^c;
   }
    i++;
    frame_count++;

   printf("Frame %d: %.*s\n",frame_count,i,buff);
   fprintf(fptr2,"%.*s",i,buff);

   printf("CheckSum %d: %c\n\n",frame_count,XOR);
   fprintf(fptr2,"%c", XOR);

   break;
}
else
{
   //printf("%c", c);
   if(i!=0)
   {
    XOR = XOR^c;
   }
   i++;
   if (i == SIZE)
```

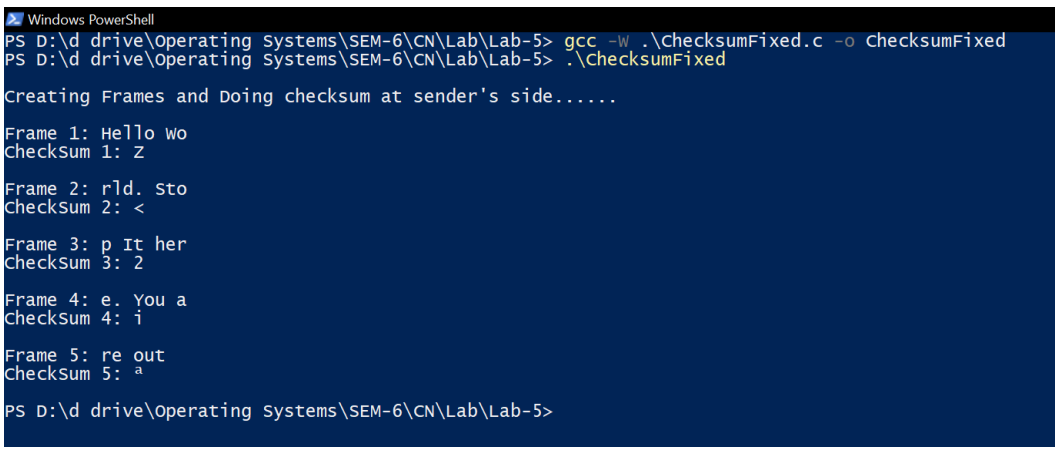
```
        {
         frame_count++;
            printf("Frame %d: %.*s\n",frame_count,SIZE,buff);
            fprintf(fptr2,"%.*s",SIZE,buff);

            printf("CheckSum %d: %c\n\n",frame_count,XOR);
            fprintf(fptr2,"%c", XOR);

            i = 0;
        }
      }
   }
   fclose(fptr1);
   fclose(fptr2);
   return (0);
}
```

Original.txt:

Output:



ChecksumFile.txt: