Date

Assignment - V

Aim :- Write a program using UDP sockets to enable file transfer (script, text, Audio & video one each file) between two machines Demonstrate the packets captured traces using Wireshark packet Analyzer tool for peer to peer mode.

Requirements :-
Fedora OS, Wireshark packet analyzer tool.

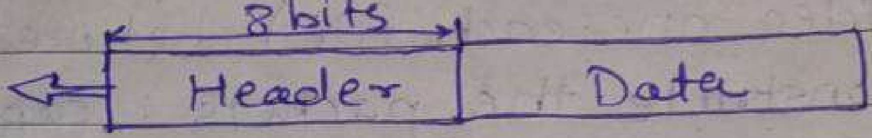
Theory :-
The user datagram protocol (UDP) is called connectionless, unreliable transport protocol. It does not add anything to services of IP except to provide process-to-process communication instead of host-to-host communication.

UDP Features :-

① Suitable for processess which require simple request response communication with little concern of error & flow of control.

② Suitable for process with internal & flow control.

③ Suitable for multitasking
④ Used for management processes such as SNMP.
⑤ Used for some root updating pro-tocol such as routing information protocol.



Source port no. 16 bits	Destinat^n Port 16 bits
Total length 16 bits	checksum 16 bits

1. User Datagram format

UDP packets, called user datagram have a fixed size header of 8 bytes figure 1 shows the format of a user datagram.

The field as follows :-

① Source port Number :-
This port number used by the process running on the source host. It is 16 bits long, which mea that the port number can range from 0 to 65535. If the sourc

host is the client, the port number, in most cases, is an ephermeral port number requested by the process & chosen by the UDP software running on the source host.

② Destination port number :-
This is the port numbers used by the process running on the destination host. It is ~~to~~ also 16 bits long.

③ Length : It is 16 bit field.

④ Checksum :- This field is used to detect errors over the entire user datagram.

<u>Socket programming</u> :-

① public DatagramSocket (int port) throws SocketException
// constructs a datagram socket & binds it to the specified port on the local host machine.

② DatagramSocket (int port, InetAddress)
// create a datagram socket, bound to the specified local address

(3) public final class DatagramPacket
// This class represents a datagram packet. Datagram packets are used to implement a connectionless packet delievery service. Each message is routed from one machine to another based solely on information contained within that packet

(4) DatagramPacket (byte [] buf, int length, InetAddress address, int port)
// constructs a datagram packet for sending packets of length to the specified port number on specified host.

(5) bind (SocketAddress addr)
// Binds this DatagramSocket to a specific address & port.

(6) void close()
// closes this datagram socket

(7) void connect (InetAddress addr, int port)
// connects the socket to a remote address for this socket.

(8) void connect (SocketAddress addr)
// connects this socket to remote socket address (IP addr & port no).

(9) void disconnect ()
// disconnet the socket.

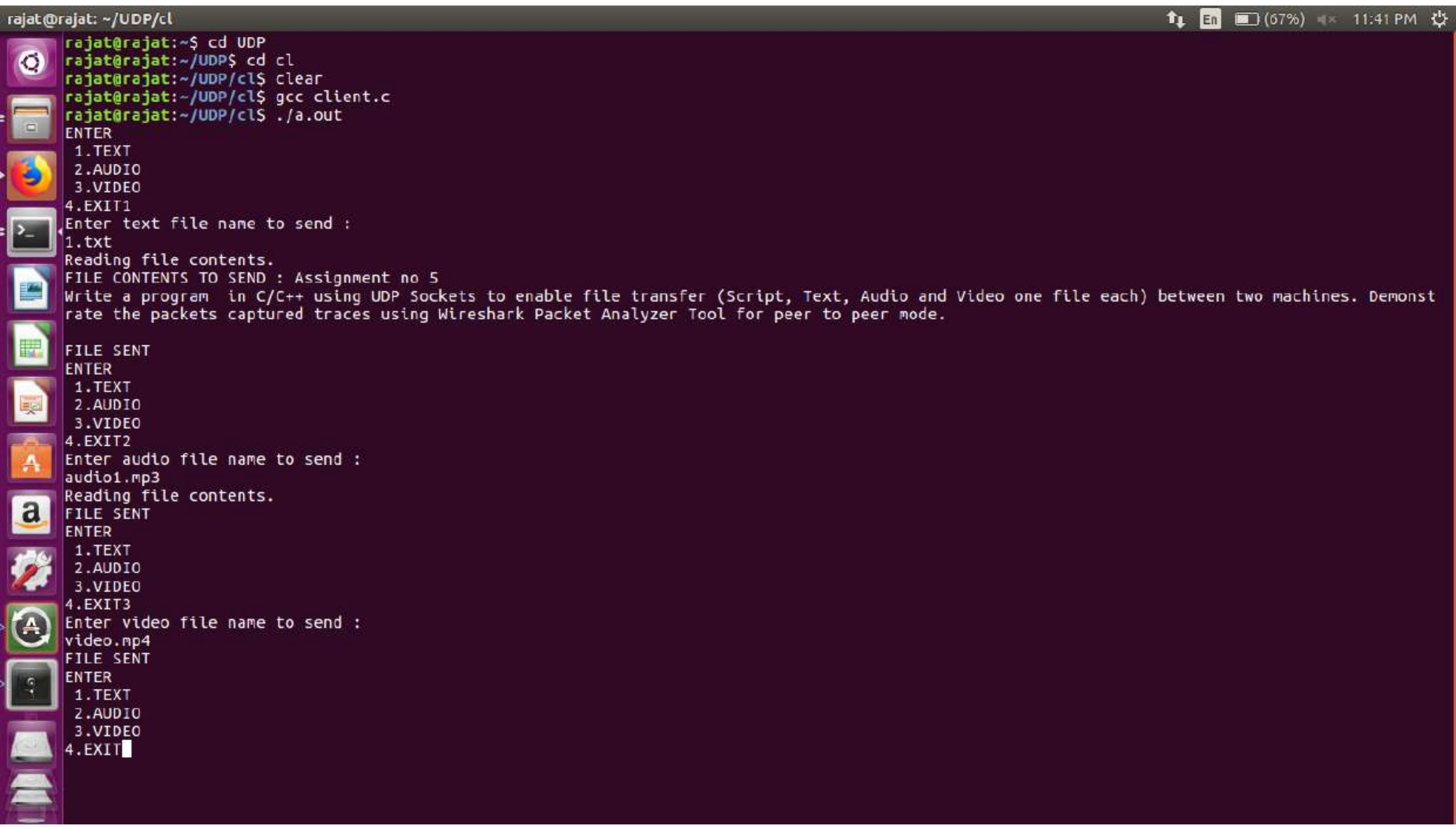
(10) Commonly used methods of Inet address class

Method	Description
String getHostName()	It returns host name of IP addr
String getHostAddr()	It returns address (IP) in string format.
public static InetAddress getByname(String host) throws UnknownHostException.	It returns Instance of InetAddress containing local host IP & Name

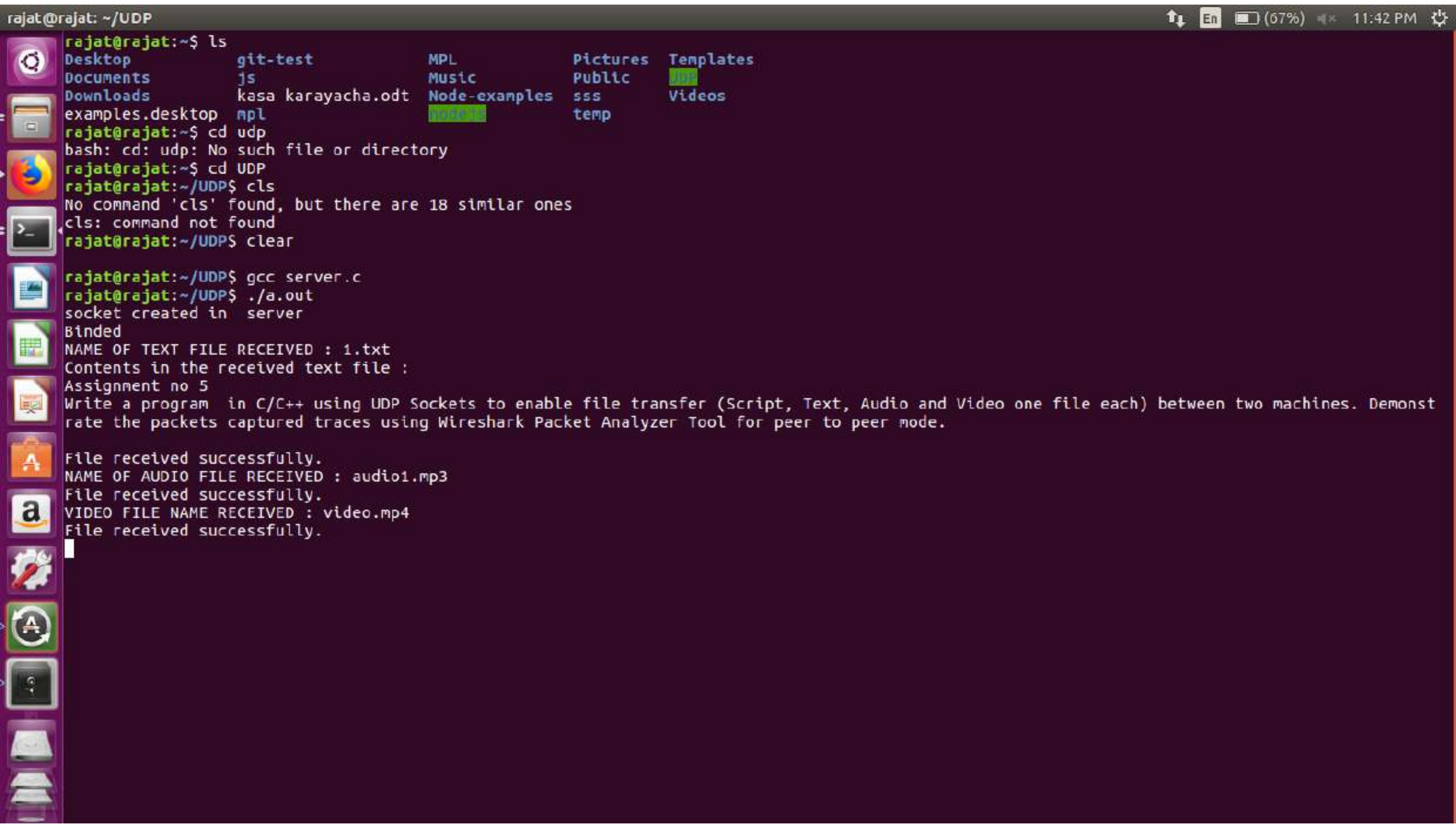
<u>Conclusion</u> :-
We successfully implement the UDP sockets to enable file ~~transfer~~ beth two machines

```
//Server

#include<sys/socket.h>
#include<arpa/inet.h>
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<string.h>
#include<stdlib.h>
#define maxlen 70000
#define mlen 100000
int main()
{
  char fileName[100];
  char filebuffer[2000],caufile[maxlen];
  char *vfilep;
  int aufile[700000],vfile[mlen];
  int sd,connfd,len;

for(int i=0;i<=100;i++){
 fileName[i]='\0';
}
  struct sockaddr_in servaddr,cliaddr;

  sd = socket(AF_INET, SOCK_DGRAM, 0);

  if(sd==-1)
    {
      printf(" socket not created in server\n");
      exit(0);
    }
  else
    {
      printf("socket created in  server\n");
    }

  bzero(&servaddr, sizeof(servaddr));

  servaddr.sin_family = AF_INET;
  servaddr.sin_addr.s_addr = INADDR_ANY;
  servaddr.sin_port = htons(8000);
  memset(&(servaddr.sin_zero),'\0',8);
  if ( bind(sd, (struct sockaddr *)&servaddr, sizeof(servaddr)) != 0 )
    printf("Not binded\n");
  else
    printf("Binded\n");

  len=sizeof(cliaddr);

  int choice =1;
  while(1)
  {
   char num;
  recvfrom(sd,&num,sizeof(num),0,(struct sockaddr *)&cliaddr, &len);


choice = num;


switch(choice)
{
 case 1:
  recvfrom(sd,fileName,1024,0,(struct sockaddr *)&cliaddr, &len);
    printf("NAME OF TEXT FILE RECEIVED : %s\n",fileName);
  FILE *fp;
     printf("Contents in the received text file : \n");
     recvfrom(sd,filebuffer,1024,0,(struct sockaddr *)&cliaddr, &len);
     printf("%s\n",filebuffer);
    int fsize=strlen(filebuffer);
  fp=fopen(fileName,"w");
    if(fp)
    {
    fwrite(filebuffer, fsize, 1, fp);
     printf("File received successfully.\n");
    }
    else
    {
     printf("Cannot create to output file.\n");
    }
    memset(fileName, '\0', sizeof(fileName));
    fclose(fp);
    break;
   case 2:
    recvfrom(sd,fileName,1024,0,(struct sockaddr *)&cliaddr, &len);
     printf("NAME OF AUDIO FILE RECEIVED : %s\n",fileName);
     FILE *afp;
     int numbytes;
       afp=fopen(fileName,"w");
       size_t  afsize;
       afsize=recvfrom(sd,aufile,700000,0,(struct sockaddr *)&cliaddr, &len);
       if(afp)
       {
        fwrite(aufile, afsize, 1, afp);
        printf("File received successfully.\n");
       }
       else
       {
        printf("Cannot open output file.\n");
       }
       memset(fileName, '\0', sizeof(fileName));
       fclose(afp);
       break;

      case 3:
       recvfrom(sd,fileName,1024,0,(struct sockaddr *)&cliaddr, &len);
         printf("VIDEO FILE NAME RECEIVED : %s\n",fileName);
         FILE *vfp;
         vfp=fopen(fileName,"w");
         size_t  vfsize;
         vfsize=recvfrom(sd,vfile,100000,0,(struct sockaddr *)&cliaddr, &len);

         if(vfp)
         {
             fwrite(vfile, vfsize, 1, vfp);
             printf("File received successfully.\n");
            }
            else
            {
             printf("Cannot open output file.\n");
            }
       fclose(vfp);
       break;

      case 4:
    close(sd);
    break;

}
 }
 return(0);
}



//Client

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

int main() {
int fd;
char fileName[2000],afileName[2000],vfileName[2000],file_buffer[2000],c,caufile[70000],aufile[7000000],vfile[1
000000];
struct sockaddr_in  servaddr;

// Creating socket file descriptor
if ( (fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
 perror("socket creation failed");
 exit(EXIT_FAILURE);
}

memset(&servaddr, 0, sizeof(servaddr));

bzero(&servaddr,sizeof(servaddr));
// Filling server information
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(8000);
servaddr.sin_addr.s_addr = INADDR_ANY;
// servaddr.sin_addr.s_addr=inet_addr("10.10.10.73");
int choice = 1;

while(choice!=4)
{
 printf("ENTER \n 1.TEXT \n 2.AUDIO \n 3.VIDEO\n4.EXIT");
 scanf("%d",&choice);

 char num=choice;

 sendto(fd, &num, sizeof(num), 0,(struct sockaddr *)&servaddr, sizeof(struct sockaddr));

 switch(choice)
 {
  case 1:
   printf("Enter text file name to send : \n");
       scanf("%s",fileName);
       sendto(fd, fileName, strlen(fileName), 0,(struct sockaddr *)&servaddr, sizeof(struct sockaddr));

       FILE *fp;
        fp=fopen(fileName,"r");

        if(fp)
        {
         printf("Reading file contents.\n");
         fseek(fp,0,SEEK_END);
           size_t file_size=ftell(fp);
           fseek(fp,0,SEEK_SET);
          if(fread(file_buffer,file_size,1,fp)<=0)
            {
              printf("Unable to copy file into buffer or empty file.\n");
              exit(1);
            }
          }
          else
          {
        printf("Cannot open file.\n");
         exit(0);
        }
       printf("FILE CONTENTS TO SEND : %s\n",file_buffer);
       if(sendto(fd, file_buffer,strlen(file_buffer), 0,(struct sockaddr *)&servaddr, sizeof(struct sockaddr))<0)
       {
         printf("FILE WAS NOT SENT\n");
       }
       else
       {
         printf("FILE SENT\n");
       }
       fclose(fp);
       break;
      case 2:
       printf("Enter audio file name to send : \n");
       scanf("%s",afileName);
       sendto(fd, afileName, strlen(afileName), 0,(struct sockaddr *)&servaddr, sizeof(struct sockaddr));
   FILE *afp;
   afp=fopen(afileName,"r");
   fseek(afp,0,SEEK_END);
   size_t  afsize=ftell(afp);
   fseek(afp,0,SEEK_SET);

   if(afp)
   {
    printf("Reading file contents.\n");
     if(fread(aufile,afsize,1,afp)<=0)
            {
              printf("Unable to copy file into buffer or empty file.\n");
              exit(1);
            }
   }
   else
   {
    printf("Could not read audio file.\n");
    exit(0);
   }

   if(sendto(fd, aufile, afsize, 0,(struct sockaddr *)&servaddr, sizeof(struct sockaddr))<0)
   {
     printf("FILE WAS NOT SENT\n");
       }
       else
       {
         printf("FILE SENT\n");
       }
   fclose(afp);
   break;

  case 3:
   printf("Enter video file name to send : \n");
       scanf("%s",vfileName);
   sendto(fd, vfileName, strlen(vfileName), 0,(struct sockaddr *)&servaddr, sizeof(struct sockaddr));
   FILE *vfp;
   vfp=fopen(vfileName,"r");

   fseek(vfp, 0, SEEK_END);
   size_t vfsize = ftell(vfp);
   fseek(vfp, 0, SEEK_SET);

   if(vfp)
   {
    if(fread(vfile, 1, vfsize, vfp)<=0)
    {
     printf("No contents or error reading file \n");
    }
```

```
   }
   else
   {
    printf("Could not read audio file.\n");
    exit(0);
   }
   if(sendto(fd, vfile, vfsize, 0,(struct sockaddr *)&servaddr, sizeof(struct sockaddr))<0)
   {
   printf("FILE WAS NOT SENT\n");
   }
   else
   {
     printf("FILE SENT\n");
   }
   fclose(vfp);
   break;

  case 4:
   close(fd);
   break;



  }


 }


}
```



```
rajat@rajat:~$ cd UDP
rajat@rajat:~/UDP$ cd cl
rajat@rajat:~/UDP/cl$ clear
rajat@rajat:~/UDP/cl$ gcc client.c
rajat@rajat:~/UDP/cl$ ./a.out
ENTER
 1.TEXT
 2.AUDIO
 3.VIDEO
4.EXIT1
Enter text file name to send :
1.txt
Reading file contents.
FILE CONTENTS TO SEND : Assignment no 5
Write a program  in C/C++ using UDP Sockets to enable file transfer (Script, Text, Audio and Video one file each) between two machines. Demonst
rate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode.

FILE SENT
ENTER
 1.TEXT
 2.AUDIO
 3.VIDEO
4.EXIT2
Enter audio file name to send :
audio1.mp3
Reading file contents.
FILE SENT
ENTER
 1.TEXT
 2.AUDIO
 3.VIDEO
4.EXIT3
Enter video file name to send :
video.mp4
FILE SENT
ENTER
 1.TEXT
 2.AUDIO
 3.VIDEO
4.EXIT
```



```
rajat@rajat:~$ ls
Desktop          git-test            MPL            Pictures  Templates
Documents        js                  Music          Public    UDP
Downloads        kasa karayacha.odt  Node-examples  sss       Videos
examples.desktop mpl                 nodeJS         temp
rajat@rajat:~$ cd udp
bash: cd: udp: No such file or directory
rajat@rajat:~$ cd UDP
rajat@rajat:~/UDP$ cls
No command 'cls' found, but there are 18 similar ones
cls: command not found
rajat@rajat:~/UDP$ clear

rajat@rajat:~/UDP$ gcc server.c
rajat@rajat:~/UDP$ ./a.out
socket created in  server
Binded
NAME OF TEXT FILE RECEIVED : 1.txt
Contents in the received text file :
Assignment no 5
Write a program  in C/C++ using UDP Sockets to enable file transfer (Script, Text, Audio and Video one file each) between two machines. Demonst
rate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode.

File received successfully.
NAME OF AUDIO FILE RECEIVED : audio1.mp3
File received successfully.
VIDEO FILE NAME RECEIVED : video.mp4
File received successfully.
```