Assignment - VI

Title :-

Write a program using TCP socket for wired network for following -

a. Say Hello to each other
b. File Transfer.
c. Calculator (mithmatic)
d. Calculator (Trigonometric).

Requirements :-

Fedora OS 64 bit, Eclipse

Theory :-

a. Say Hello to Each other

1. TCP socket programming for wired network :

The two key classes from java .net package used in creation of server & client programs are:
Σ server socket & Σ Socket. A server program creates a specific type of socket that is used to listen for client request (server socket), In case of

connection request, the program creates a new socket through which it will exchange data with the client using input & output streams. The socket abstraction is very similar to the file concept: developers have to open a socket, perform I/O & close it.

A simple server program in Java

The steps for creating a simple server program are:

① Open the server socket:
Server_socket server = new Server_socket (PORT)
② Wait for the client request
Server client = server.accept();
③ Create I/O streams for communicating to the client DataInputStream is = new DataInputstream (client.getInputStream());
④ perform communication with client recieve from client
⑤ close socket : client.close();

A simple client program in Java

The steps for creating a simple client program are:

① Create a socket object :-
Socket client = new socket (server, port_id)
② Create I/O streams for communicating with the servers, is- new DataInput-stream (client.getInputStream)
③ Perform I/O / communication with server
Receive data from the server :
String line = is.readline();
send data to server :
OS.writeBytes ("Hello");
④ Close the socket when done client.close

2. Running Socket Programs

Compile both server & client programs & then deploy server program code on a machine which is going to act as a server & client program which is going to act as a client.
If required, both client & server programs can run on the same machine To illustrate execution of server & client programs let us assume that a machine called mundroo.cse.unimelb.edu.au on which one want to run a server program as indicated below.
[raj@mundroo] java simpleserver

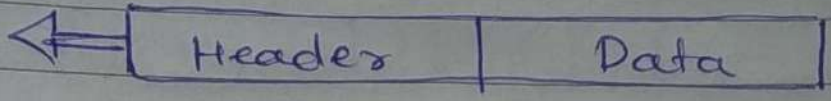
The client program can run on any computer in the network (LAN or WAN or internet) as long as there is no firewall bet" then that blocks communication. let us way we want to run our client.

[raj@gribus] java SimpleClient.

The client program is just establishing a connection with the server & then waits for a message

b. File transfer :

A TCP client, initiates the communication with the server which is waiting for the connection TCP is connection oriented & UDP is connection less, which means that UDP sockets do not need to be connected before being used.



Source Port Address 16 bits								Destination port address
Sequence number 32 bits								
Acknowledgement number 32 bits								
HLEN 4 bits	Reserved 6 bits	URG	ACK	PSH	RST	SYN	FIN	Window Size 16 bits
checksum 16 bits								Urgent pointer 16 bits
Options & Padding								

TCP Header.

Conclusion :-

Hence we studied & implemented program to demonstrate TCP socket programming.

```
//cserver.cpp

#include <iostream>
#include <sys/socket.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>

using namespace std;

#define PORT 8052
void die(char *error)
{
 perror(error);
 exit(1);
}
int main() {

 sockaddr_in server_addr,client_addr;

 int sock=socket(AF_INET,SOCK_STREAM,0);
 if(sock<0)
  die("SOCKET CREATE ERROR");
 else
  cout<<"Socket Created.";
 bzero((char *)&server_addr,sizeof(server_addr));
 server_addr.sin_family=AF_INET;
 server_addr.sin_addr.s_addr=INADDR_ANY;
 server_addr.sin_port=htons(PORT);

 if(bind(sock,(struct sockaddr*)&server_addr,sizeof(server_addr))==-1)
  die("ERROR IN BINDING");

 if(listen(sock,10)<0)
 {
  die("ERROR WHILE LISTENING");
 }
 socklen_t socklen=sizeof(client_addr);

 int newSocket=accept(sock,(struct sockaddr*)&client_addr,&socklen);

 char buffer[256];

 while(1)
 { cout << "Awaiting client response..." << endl;
  bzero(buffer,256);
  recv(newSocket,buffer,255,0);
  cout<<"Client: "<<buffer<<endl;
  cout<<">";
  //cin.ignore();
  string data;
  getline(cin,data);
  bzero(buffer,256);
```

```
 strcpy(buffer,data.c_str());

 //cin.clear();

    fflush(stdin);
 send(newSocket,(char*)&buffer,strlen(buffer),0);
 }
 return 0;
}



//cclient.cpp

#include <iostream>
#include <sys/socket.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
using namespace std;

#define PORT 8052
#define SERVER_ADDRESS "127.0.0.1"
void die(char *error)
{
 perror(error);
 exit(1);
}
int main() {

 struct sockaddr_in server_addr;
 int sock=socket(AF_INET,SOCK_STREAM,0);
 if(sock<0)
            cout<<"Socket Could Not Be Created";
       else
            cout<<"Socket Created Succesfully";
 server_addr.sin_addr.s_addr=INADDR_ANY;
 server_addr.sin_family=AF_INET;
 server_addr.sin_port=htons(PORT);

 int status=connect(sock,(struct sockaddr *)&server_addr,sizeof(server_addr));
  if(status==0)
  cout<<"\nCONNECT SUCCESS!.";
  else
   die("connect");

  char buffer[256];
  while(1)
  {
   bzero((char *)buffer,256);
    cout<<">";
    string data;
    getline(cin, data);
    strcpy(buffer,data.c_str());
```

```
  send(sock,buffer,strlen(buffer),0);
  bzero(buffer,256);
  cout << "Awaiting server response..." << endl;
  recv(sock,(char*)&buffer,sizeof(buffer),0);
  cout<<"Server: "<<buffer<<endl;

 }
return 0;
}


//fserver.cpp


#include <iostream>
#include <sys/socket.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
#include<fstream>
using namespace std;

#define PORT 8566
void die(char *error)
{
 perror(error);
 exit(1);
}
int main() {

 sockaddr_in server_addr,client_addr;

 int sock=socket(AF_INET,SOCK_STREAM,0);
 if(sock<0)
  die("SOCKET CREATE ERROR");
 else
  cout<<"Socket Created.";
 bzero((char *)&server_addr,sizeof(server_addr));
 server_addr.sin_family=AF_INET;
 server_addr.sin_addr.s_addr=INADDR_ANY;
 server_addr.sin_port=htons(PORT);

 if(bind(sock,(struct sockaddr*)&server_addr,sizeof(server_addr))==-1)
  die("ERROR IN BINDING");

 if(listen(sock,10)<0)
 {
  die("ERROR WHILE LISTENING");
 }
 socklen_t socklen=sizeof(client_addr);

 int newSocket=accept(sock,(struct sockaddr*)&client_addr,&socklen);
 if(newSocket<0)
```

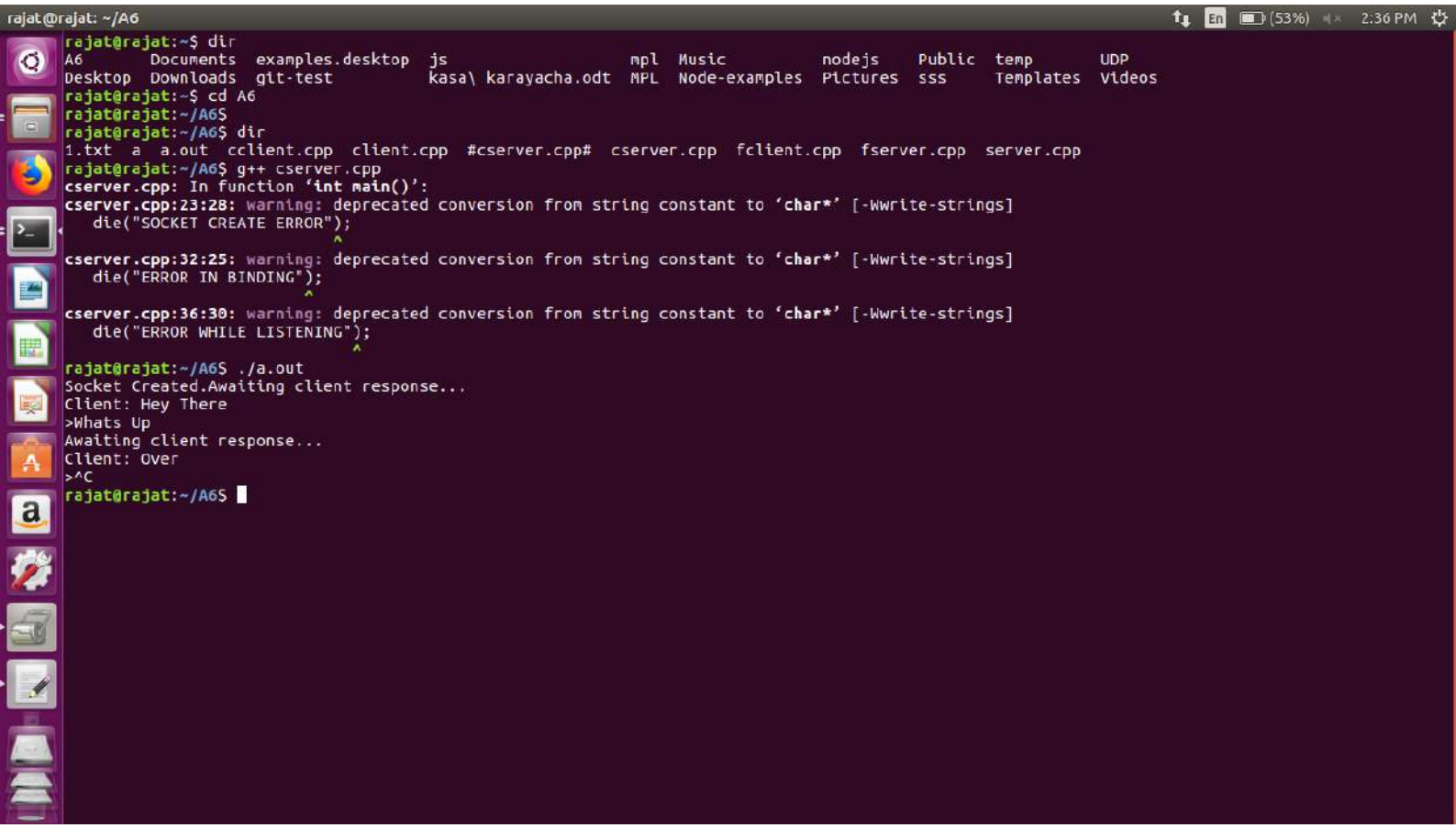
```
 die("ACCEPT ERROR");
else
 cout<<"\nCONNECTION ACCEPTED";
long long int msg_len;


{
 cout<<"\nENter Filename:";
 char filename[100];
 cin>>filename;
 cout<<filename;
 fstream fout;



 msg_len=send(newSocket,filename,100,0); //send filename
 if(msg_len==-1)
  die("Filename error");

 fout.open(filename,ios::in|ios::out|ios::binary);
   fout.seekg(0,ios::end);
   long long int filesize=fout.tellg(); //get file size
   char *filebuff=new char[filesize];
   fout.seekg(0,ios::beg);
   fout.read(filebuff,filesize); //reading file content



 msg_len=send(newSocket,filebuff,filesize,0); //send file conetents
 if(msg_len==-1)
  die("File transmission error");
 else
  cout<<"Transmission Successful";
 fout.close();
}
return 0;
}


//fclient.cpp

#include <iostream>
#include <sys/socket.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
#include<fstream>
using namespace std;
#define SERVER_ADDRESS "127.0.0.1"
#define PORT 8566
void die(char *error)
{
 perror(error);
 exit(1);
```

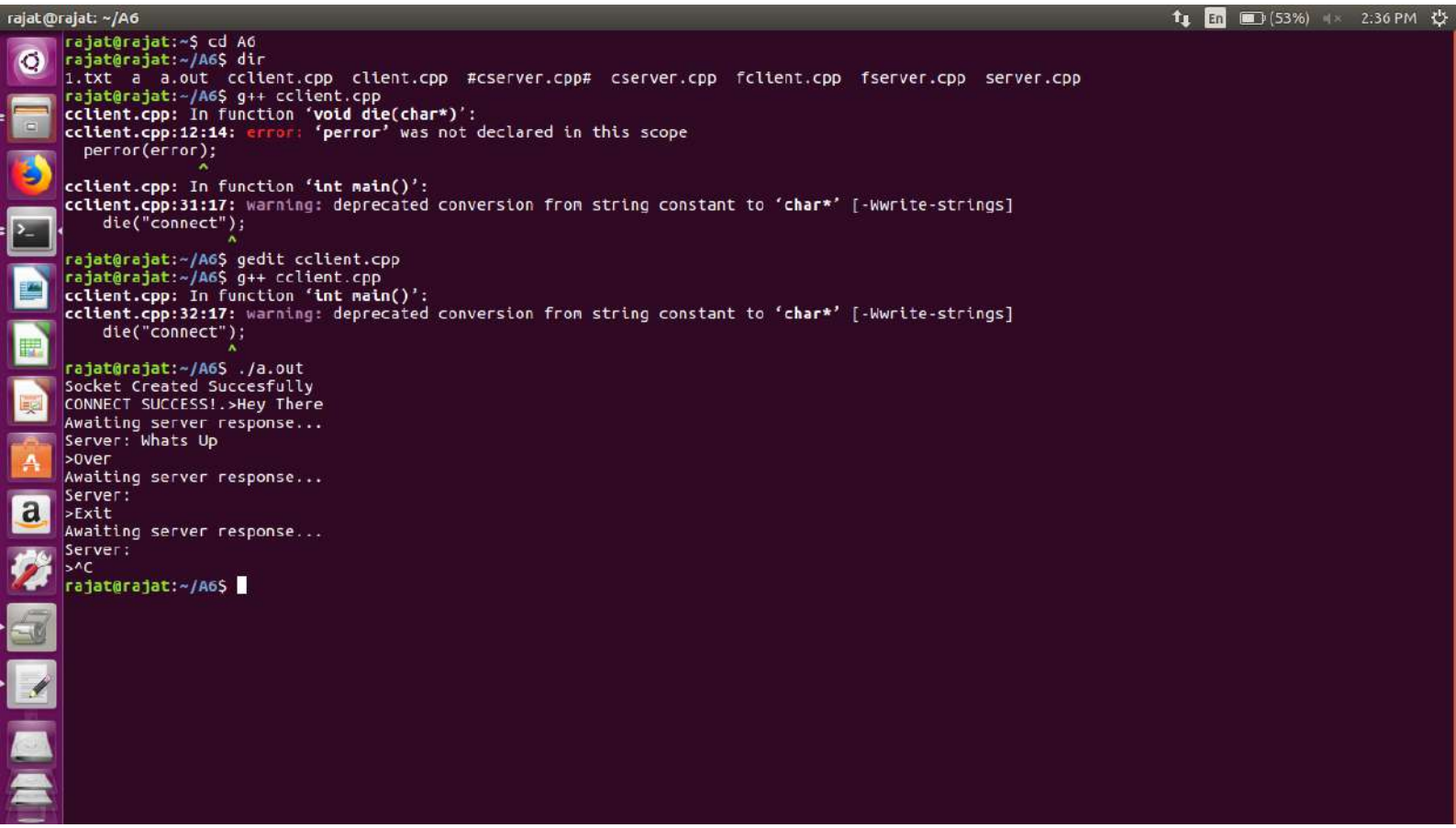
```
}
int main() {

    struct sockaddr_in server_addr;
  int sock=socket(AF_INET,SOCK_STREAM,0);
  if(sock<0)
                cout<<"Socket Could Not Be Created";
          else
                cout<<"Socket Created Succesfully";
  server_addr.sin_addr.s_addr=INADDR_ANY;
  server_addr.sin_family=AF_INET;
  server_addr.sin_port=htons(PORT);

  int status=connect(sock,(struct sockaddr *)&server_addr,sizeof(server_addr));
   if(status==0)
   cout<<"\nCONNECT SUCCESS!.";
   else
    die("connect");
long long int msg_len;
   char buffer[256];

 {
    cout<<"Wating for server to send  filename.";
    char filename[100];
    bzero((char *)filename,sizeof(filename));
    msg_len=recv(sock,filename,100,0);
    if(msg_len==-1)
     die("Filename error");
    cout<<"\nFilename:"<<filename;


    char *filebuff=new char[90000*80];


    bzero((char *)filebuff,sizeof(filebuff));
    msg_len=recv(sock,filebuff,90000*80,0);
    ofstream fout;
    fout.open(filename,ios::out|ios::binary);
    if(!fout)
     die("CANNOT CREATE FILE");
    else
    {
     fout.write(filebuff,msg_len);
     fout.close();
     cout<<"File received";
    }
   }
 return 0;
}



//server.java
```

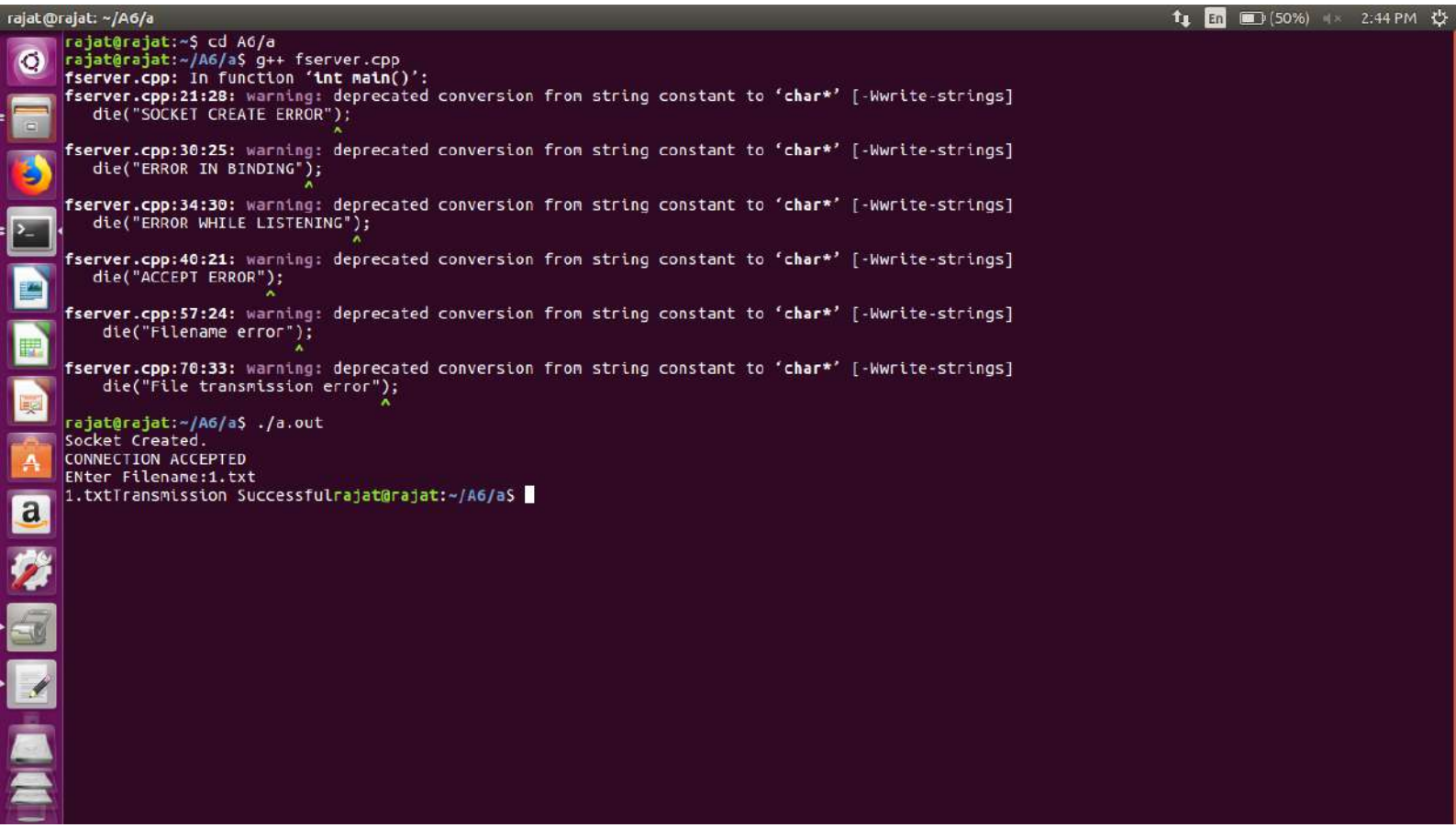
```
#include<iostream>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<stdlib.h>  //for exit
#include<unistd.h>
#include <string.h>
#include <stdio.h>
using namespace std;

int main(int argc, char const *argv[])
{
 int sock = socket(AF_INET,SOCK_STREAM,0);
 struct sockaddr_in server,client;

 server.sin_family = AF_INET;
 server.sin_port   = htons(8003);
 server.sin_addr.s_addr = INADDR_ANY;

 if(bind(sock,(struct sockaddr*)&server,sizeof(server))){
  cout<<"\nBIND ERROR\n";
 }

 if (listen(sock,5)<0){
  cout<<"\nERROR ON LISTEN\n";
 }

 socklen_t clientlen = sizeof(client);
 int newsock = accept(sock,(struct sockaddr *)&client,&clientlen);

 float number1,number2,answer;
 char _operator[2],num1[20],num2[20];

 while(1){
  recv(newsock,num1,20,0);
  cout<<"\nThe first number is "<<num1<<endl;
  number1 = atof(num1);
  bzero((char*)num1,sizeof(num1));

  recv(newsock,num2,20,0);
  cout<<"\nThe second number is "<<num2<<endl;
  number2 = atof(num2);
  bzero((char*)num2,sizeof(num2));

  recv(newsock,_operator,2,0);
  cout<<"\nThe operator  is "<<_operator<<endl;

  switch(_operator[0]) {
   case '+':
   {
    char ans[20];
    answer = number1 + number2;
    bzero((char*)ans,sizeof(ans));
    sprintf(ans,"%f",answer);
```

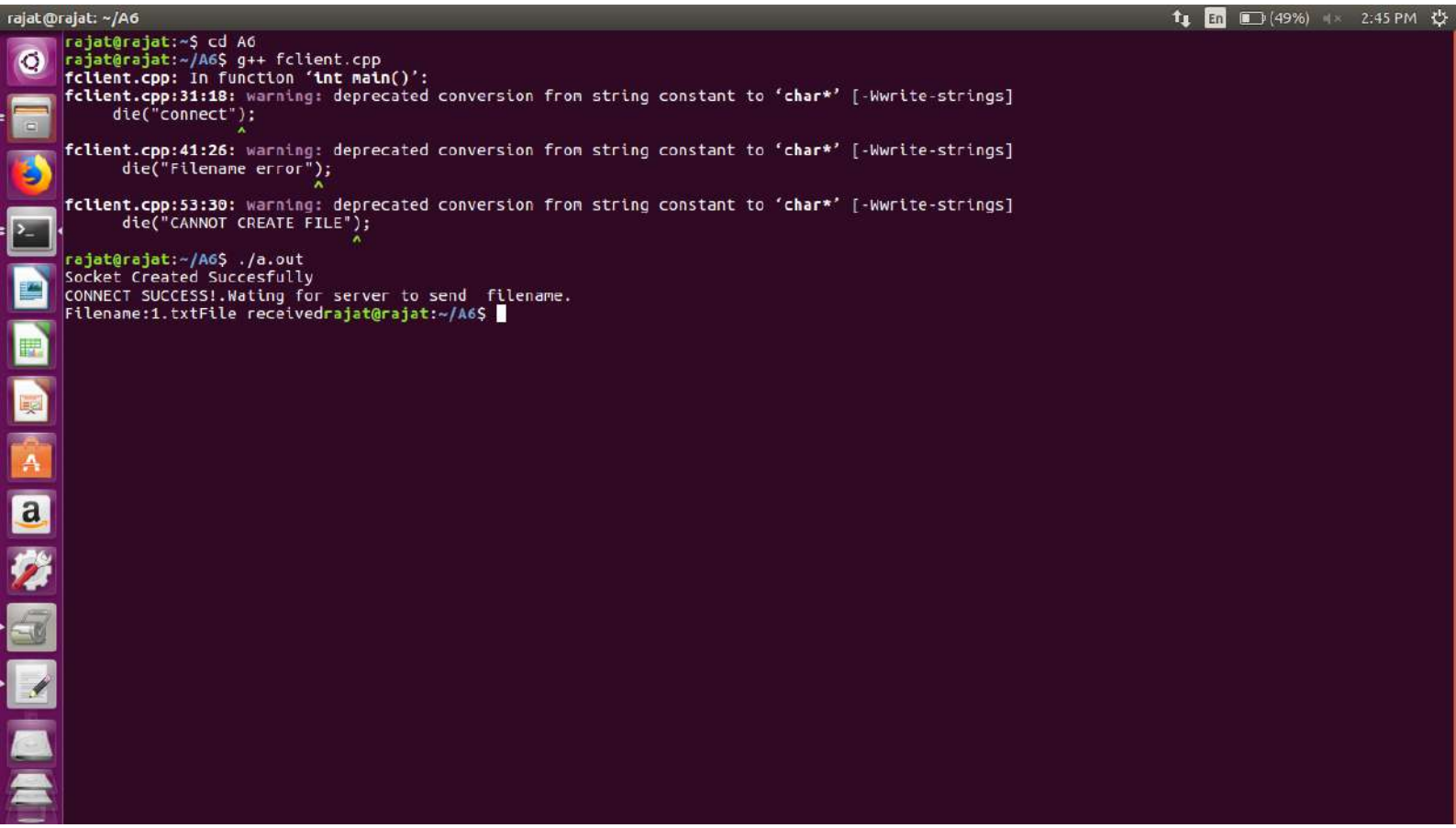
```
    send(newsock,ans,strlen(ans),0);
    break;
   }
   case '-':
   {
    char ans[20];
    answer = number1 - number2;
    bzero((char*)ans,sizeof(ans));
    sprintf(ans,"%f",answer);
    send(newsock,ans,strlen(ans),0);
    break;
   }
   case '*':
   {
    char ans[20];
    answer = number1 * number2;
    bzero((char*)ans,sizeof(ans));
    sprintf(ans,"%f",answer);
    send(newsock,ans,strlen(ans),0);
    break;
   }
   case '/':
   {
    char ans[20];
    answer = number1 / number2;
    bzero((char*)ans,sizeof(ans));
    sprintf(ans,"%f",answer);
    send(newsock,ans,strlen(ans),0);
    break;
   }
  }
  bzero((char*)_operator,sizeof(_operator));
 }
}

// client.cpp

#include<iostream>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<string.h>
#include<stdlib.h>  //for exit
#include<unistd.h>
#include<arpa/inet.h>//for close
using namespace std;

int main()
{
 int n;
 char a[20],b[20],c[20],ans[20];

 int sock = socket(AF_INET,SOCK_STREAM,0);
 struct sockaddr_in server;
```

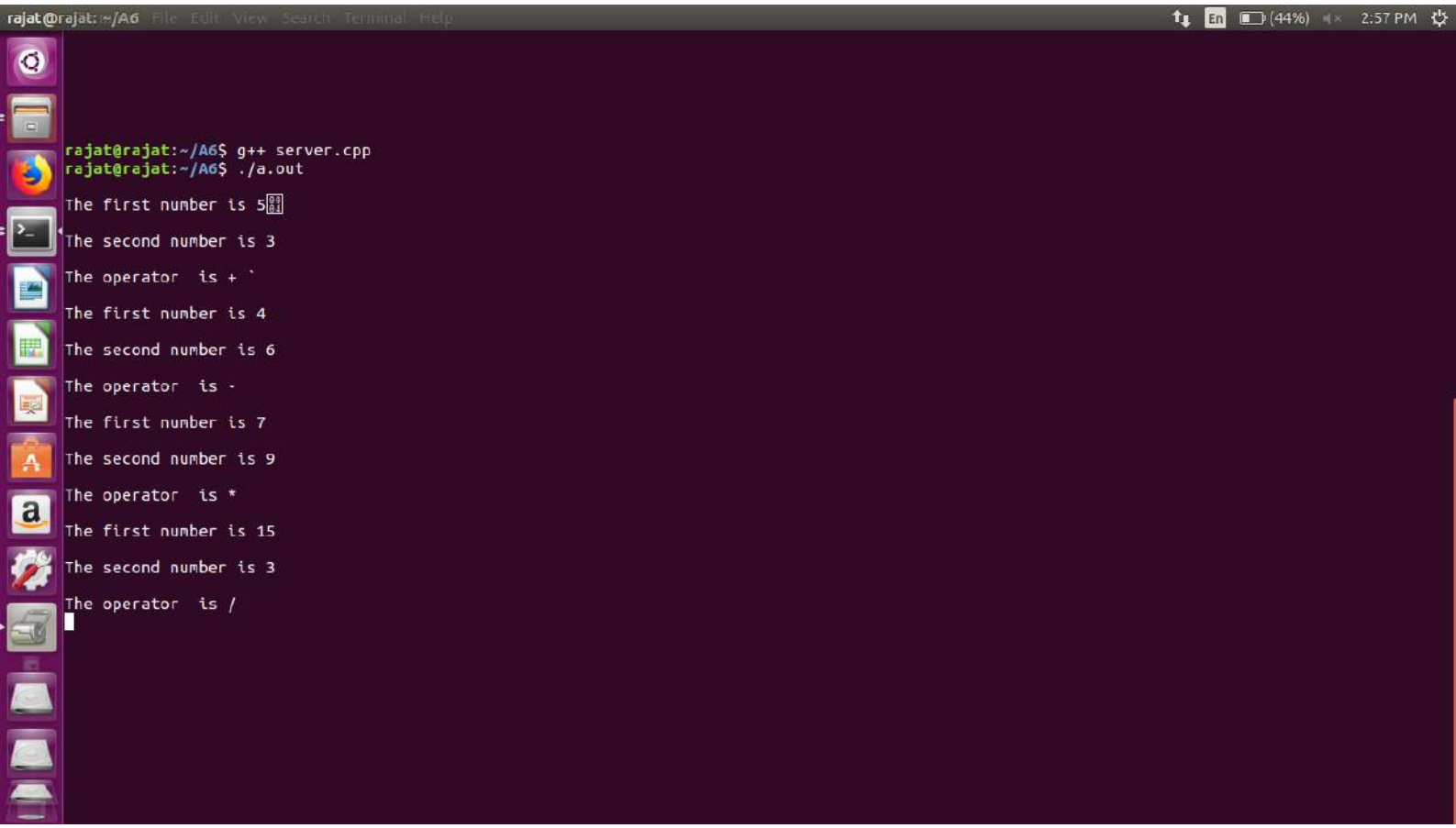
```
server.sin_family = AF_INET;
server.sin_port   = htons(8003);
server.sin_addr.s_addr = INADDR_ANY;
cout<<ntohl(server.sin_addr.s_addr);

connect(sock,(struct sockaddr *)&server,sizeof(server));

while(1){

 cout<<"\nEnter First Number\n";
 cin>>a;
 send(sock,a,strlen(a),0);
 bzero((char*)a,sizeof(a));

 cout<<"\nEnter Second Number\n";
 cin>>b;
 send(sock,b,strlen(b),0);
 bzero((char*)b,sizeof(b));

 cout<<"\nEnter Operator\n";
 cin>>c;
 send(sock,c,strlen(c),0);
 bzero((char*)c,sizeof(c));

 recv(sock,ans,20,0);
 cout<<"Result:"<<(float)atof(ans);
 bzero((char*)ans,sizeof(ans));
 }
}
```

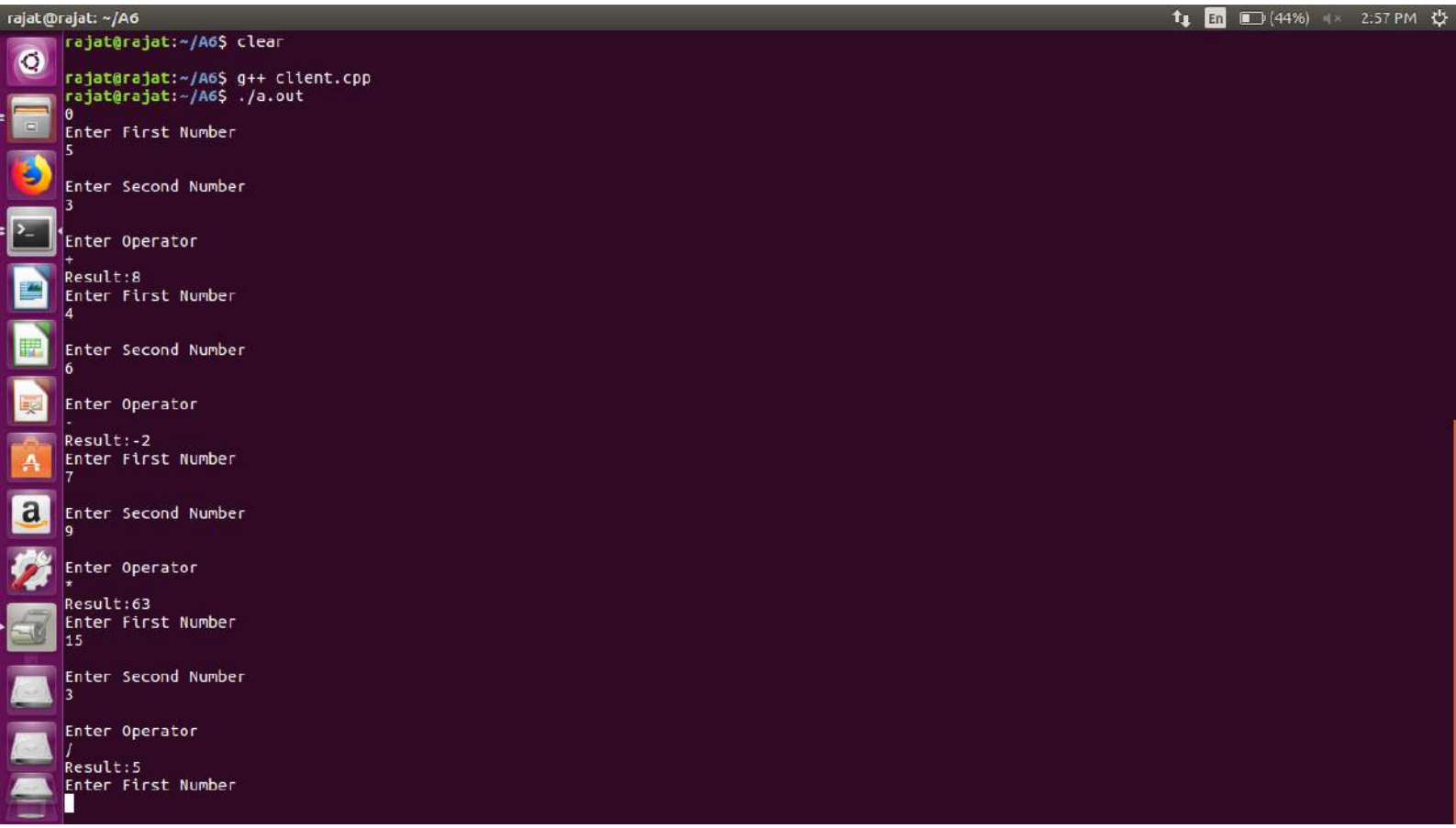

rajat@rajat: ~/A6
(53%)
2:36 PM
rajat@rajat:~$ dir
A6 Documents examples.desktop js mpl Music nodejs Public temp UDP
Desktop Downloads git-test kasa\ karayacha.odt MPL Node-examples Pictures sss Templates Videos
rajat@rajat:~$ cd A6
rajat@rajat:~/A6$
rajat@rajat:~/A6$ dir
1.txt a a.out cclient.cpp client.cpp #cserver.cpp# cserver.cpp fclient.cpp fserver.cpp server.cpp
rajat@rajat:~/A6$ g++ cserver.cpp
cserver.cpp: In function 'int main()':
cserver.cpp:23:28: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
die("SOCKET CREATE ERROR");
^
cserver.cpp:32:25: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
die("ERROR IN BINDING");
^
cserver.cpp:36:30: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
die("ERROR WHILE LISTENING");
^
rajat@rajat:~/A6$ ./a.out
Socket Created.Awaiting client response...
Client: Hey There
>Whats Up
Awaiting client response...
Client: Over
>^C
rajat@rajat:~/A6$

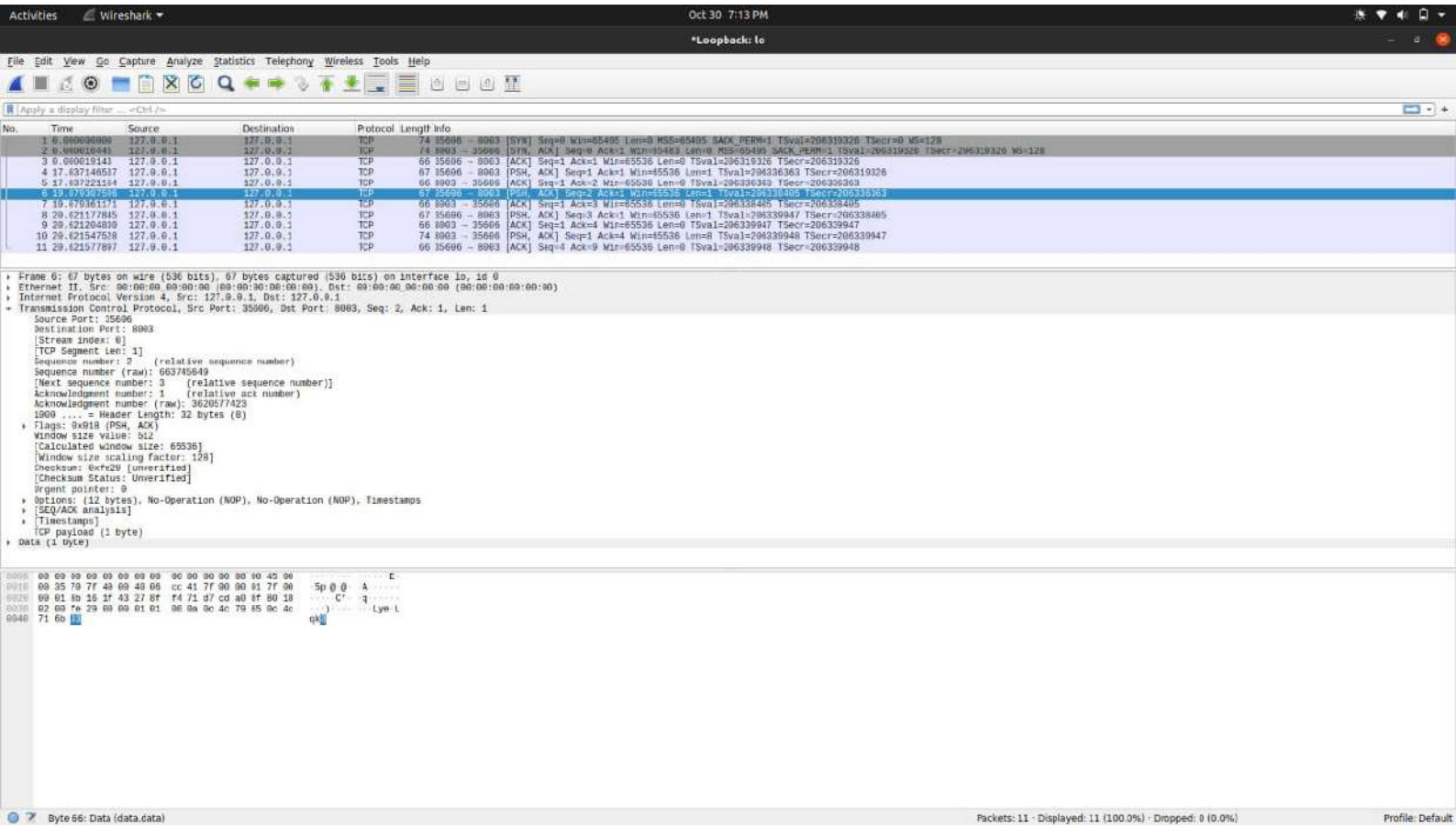

rajat@rajat: ~/A6
(53%)
2:36 PM
rajat@rajat:~$ cd A6
rajat@rajat:~/A6$ dir
1.txt a a.out cclient.cpp client.cpp #cserver.cpp# cserver.cpp fclient.cpp fserver.cpp server.cpp
rajat@rajat:~/A6$ g++ cclient.cpp
cclient.cpp: In function 'void die(char*)':
cclient.cpp:12:14: error: 'perror' was not declared in this scope
  perror(error);
^
cclient.cpp: In function 'int main()':
cclient.cpp:31:17: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
    die("connect");
^
rajat@rajat:~/A6$ gedit cclient.cpp
rajat@rajat:~/A6$ g++ cclient.cpp
cclient.cpp: In function 'int main()':
cclient.cpp:32:17: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
    die("connect");
^
rajat@rajat:~/A6$ ./a.out
Socket Created Succesfully
CONNECT SUCCESS!.>Hey There
Awaiting server response...
Server: Whats Up
>Over
Awaiting server response...
Server:
>Exit
Awaiting server response...
Server:
>^C
rajat@rajat:~/A6$

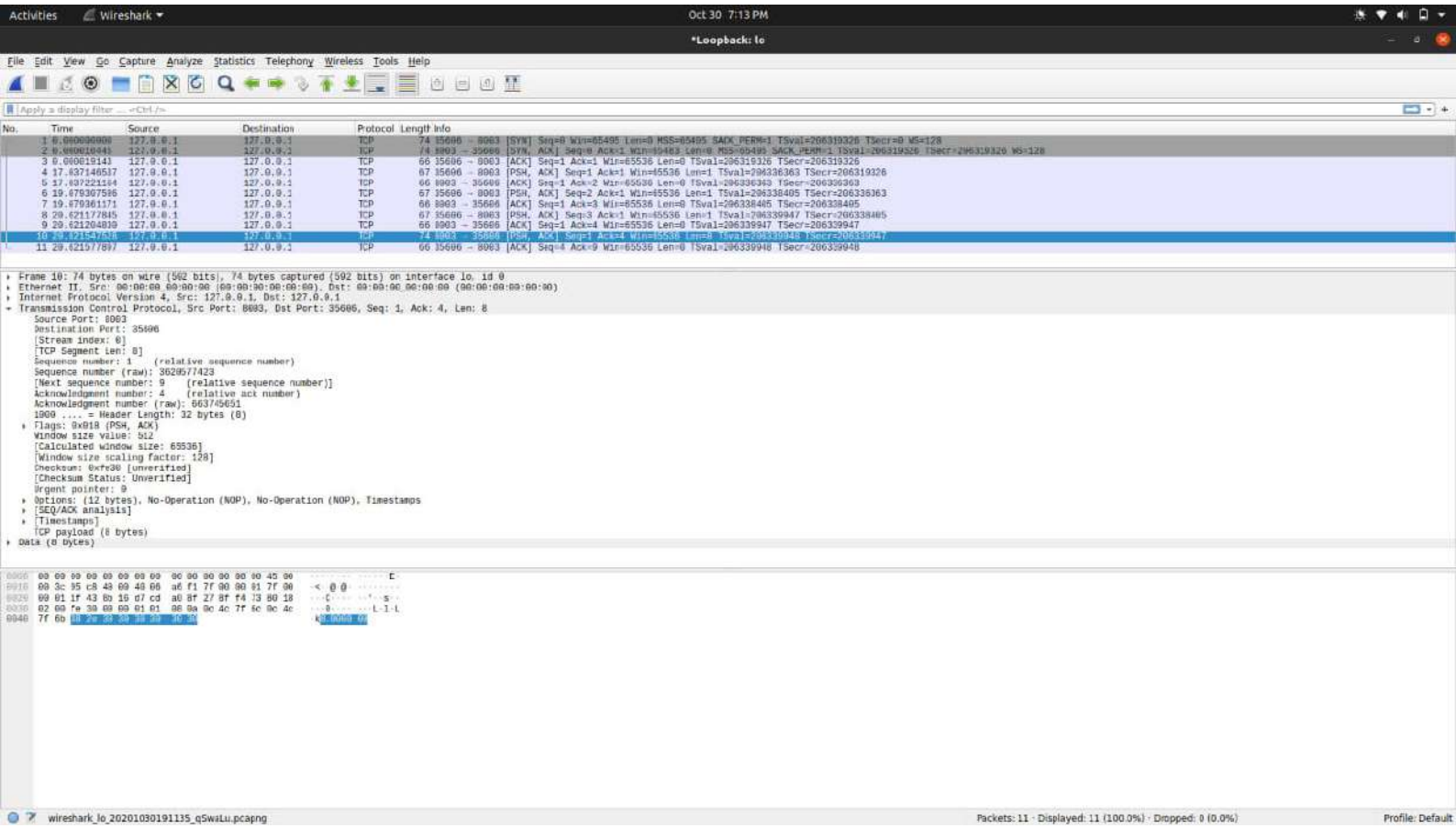

rajat@rajat: ~/A6/a
En
(50%)
2:44 PM
rajat@rajat:~$ cd A6/a
rajat@rajat:~/A6/a$ g++ fserver.cpp
fserver.cpp: In function 'int main()':
fserver.cpp:21:28: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
die("SOCKET CREATE ERROR");
^
fserver.cpp:30:25: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
die("ERROR IN BINDING");
^
fserver.cpp:34:30: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
die("ERROR WHILE LISTENING");
^
fserver.cpp:40:21: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
die("ACCEPT ERROR");
^
fserver.cpp:57:24: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
die("Filename error");
^
fserver.cpp:70:33: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
die("File transmission error");
^
rajat@rajat:~/A6/a$ ./a.out
Socket Created.
CONNECTION ACCEPTED
ENter Filename:1.txt
1.txtTransmission Successfulrajat@rajat:~/A6/a$

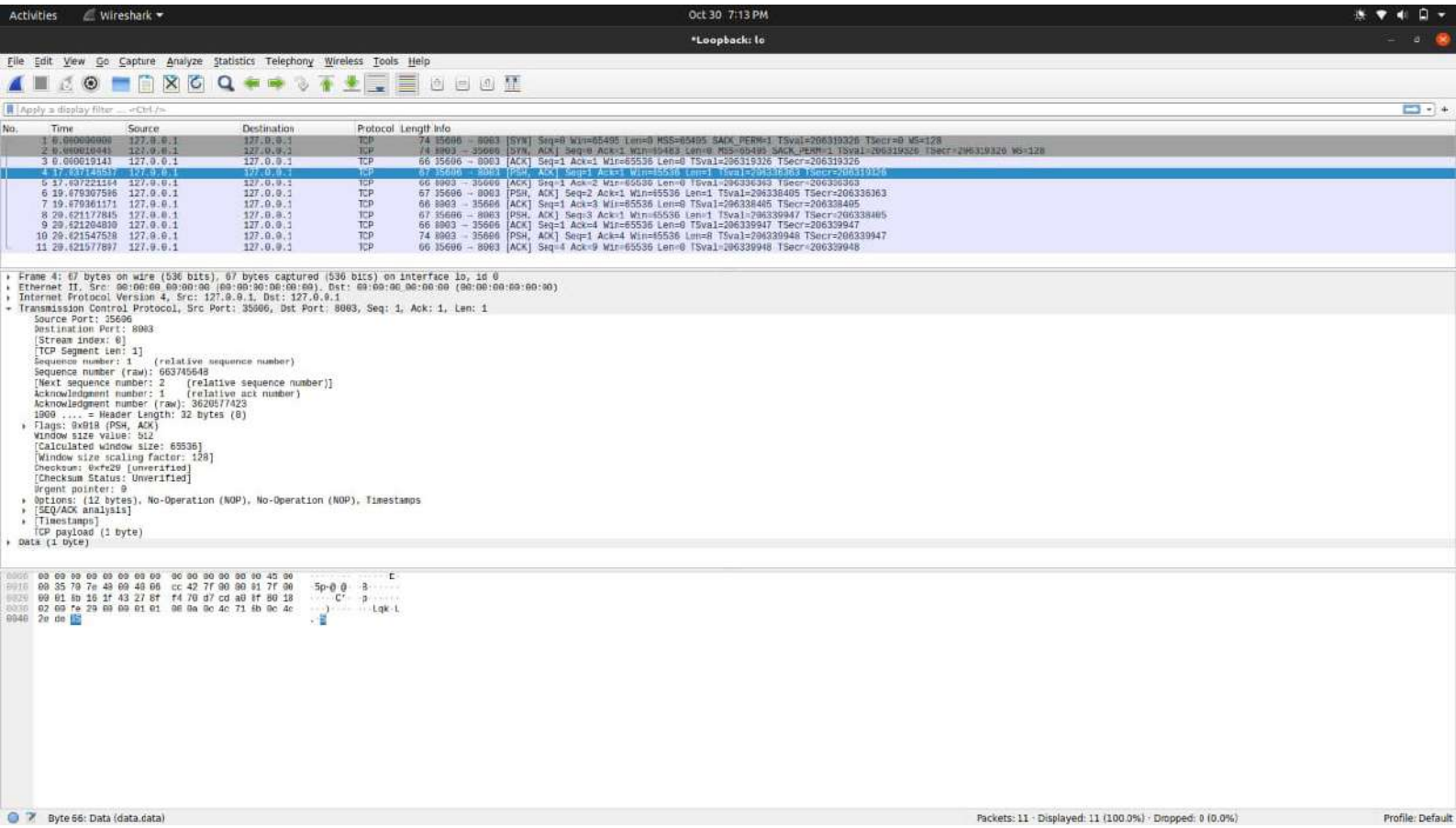

rajat@rajat: ~/A6
En
(49%)
2:45 PM
rajat@rajat:~$ cd A6
rajat@rajat:~/A6$ g++ fclient.cpp
fclient.cpp: In function 'int main()':
fclient.cpp:31:18: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
die("connect");
^
fclient.cpp:41:26: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
die("Filename error");
^
fclient.cpp:53:30: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
die("CANNOT CREATE FILE");
^
rajat@rajat:~/A6$ ./a.out
Socket Created Succesfully
CONNECT SUCCESS!.Wating for server to send filename.
Filename:1.txtFile receivedrajat@rajat:~/A6$

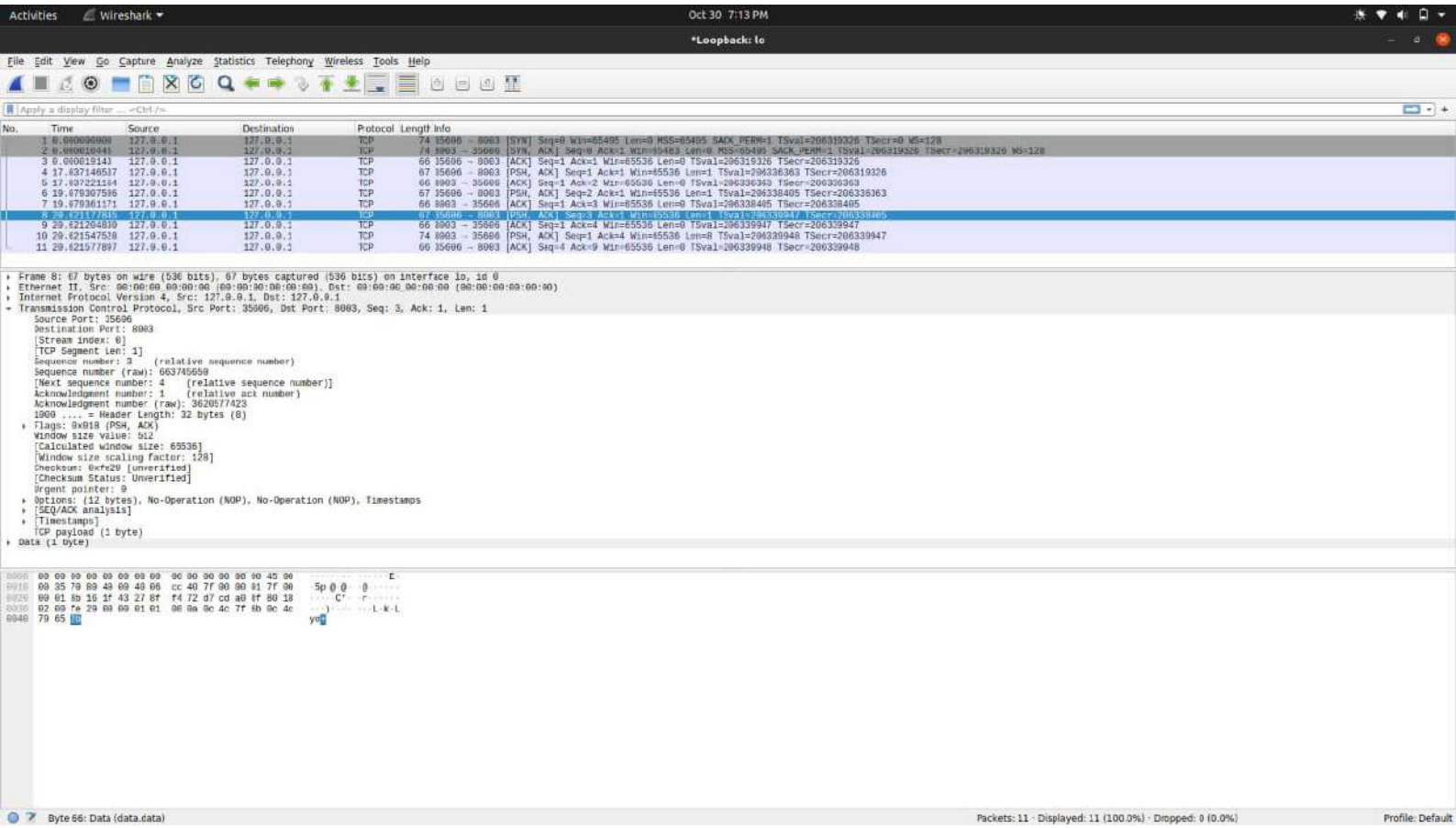

rajat@rajat: ~/A6 File Edit View Search Terminal Help
En (44%) 2:57 PM
rajat@rajat:~/A6$ g++ server.cpp
rajat@rajat:~/A6$ ./a.out
The first number is 5
The second number is 3
The operator is + `
The first number is 4
The second number is 6
The operator is -
The first number is 7
The second number is 9
The operator is *
The first number is 15
The second number is 3
The operator is /

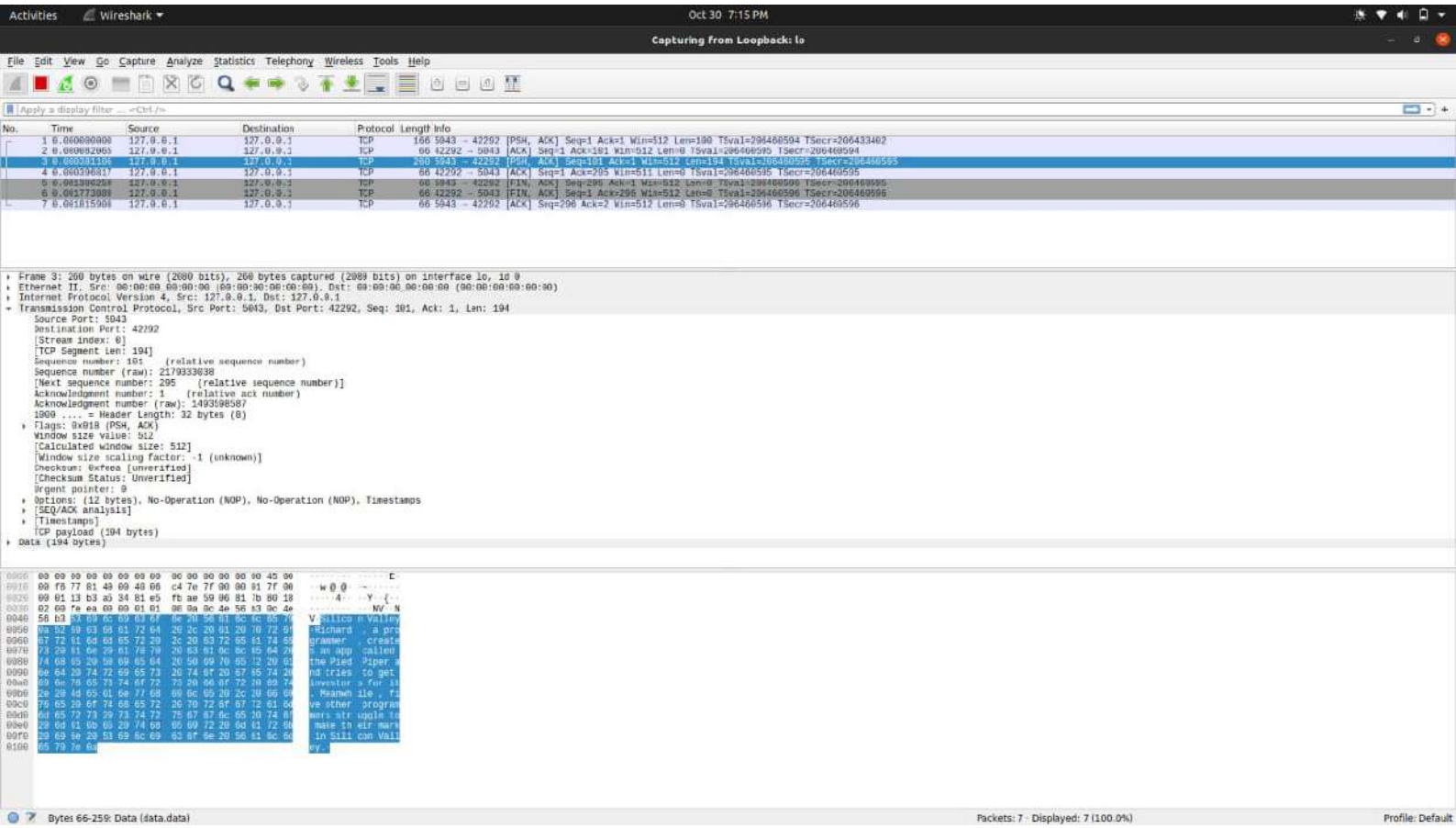

rajat@rajat: ~/A6
En
(44%)
2:57 PM
rajat@rajat:~/A6$ clear
rajat@rajat:~/A6$ g++ client.cpp
rajat@rajat:~/A6$ ./a.out
0
Enter First Number
5
Enter Second Number
3
Enter Operator
+
Result:8
Enter First Number
4
Enter Second Number
6
Enter Operator
-
Result:-2
Enter First Number
7
Enter Second Number
9
Enter Operator
*
Result:63
Enter First Number
15
Enter Second Number
3
Enter Operator
/
Result:5
Enter First Number

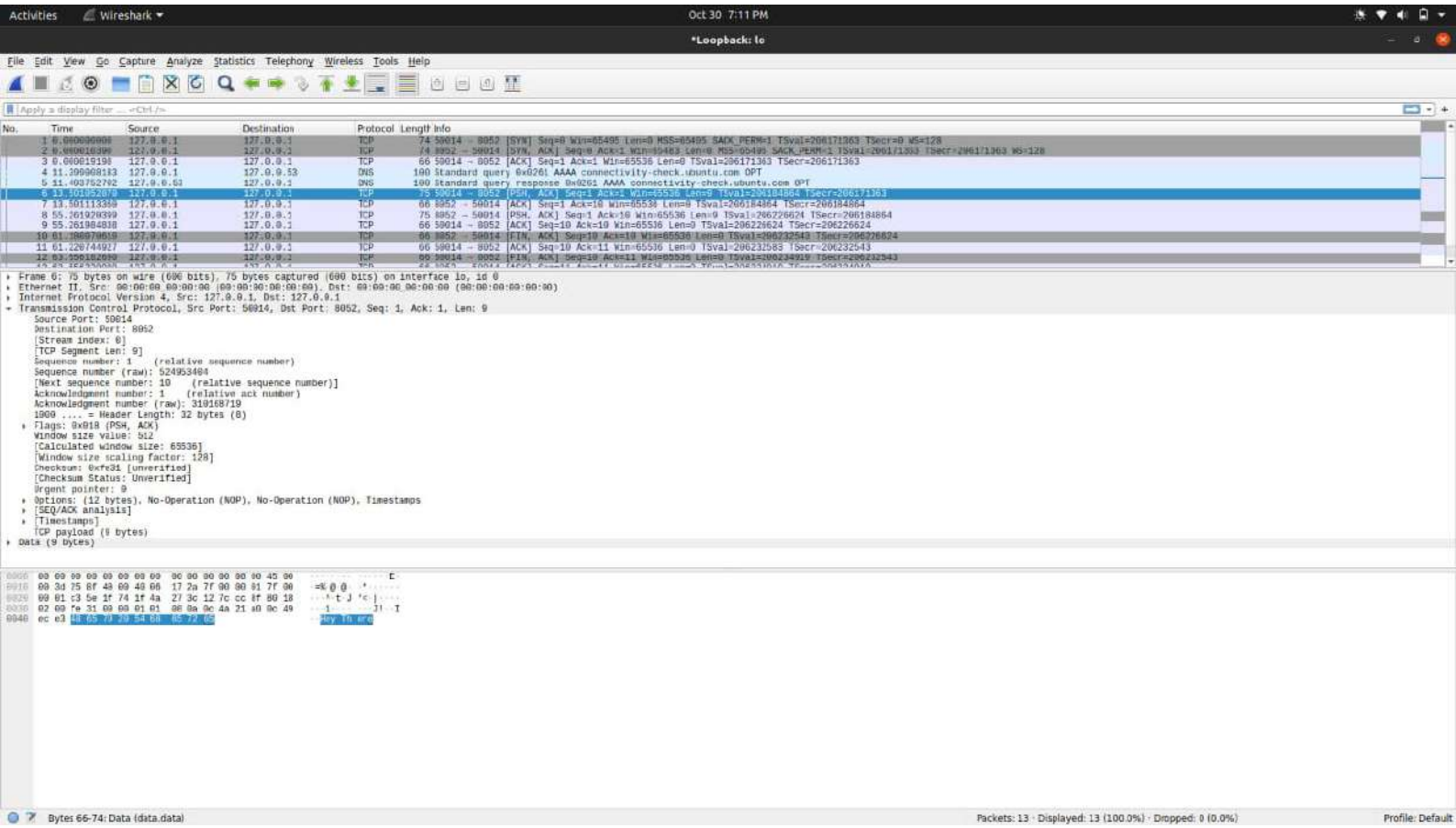

Activities
Wireshark
Oct 30 7:13 PM
*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 127.0.0.1 127.0.0.1 TCP 74 35606 → 8003 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=206319326 TSecr=0 WS=128
2 0.000010441 127.0.0.1 127.0.0.1 TCP 74 8003 → 35606 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=206319326 TSecr=206319326 WS=128
3 0.000019143 127.0.0.1 127.0.0.1 TCP 66 35606 → 8003 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=206319326 TSecr=206319326
4 17.037146537 127.0.0.1 127.0.0.1 TCP 67 35606 → 8003 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=1 TSval=206336363 TSecr=206319326
5 17.037221184 127.0.0.1 127.0.0.1 TCP 66 8003 → 35606 [ACK] Seq=1 Ack=2 Win=65536 Len=0 TSval=206336363 TSecr=206336363
6 19.079307586 127.0.0.1 127.0.0.1 TCP 67 35606 → 8003 [PSH, ACK] Seq=2 Ack=1 Win=65536 Len=1 TSval=206338405 TSecr=206336363
7 19.079361171 127.0.0.1 127.0.0.1 TCP 66 8003 → 35606 [ACK] Seq=1 Ack=3 Win=65536 Len=0 TSval=206338405 TSecr=206338405
8 20.621177845 127.0.0.1 127.0.0.1 TCP 67 35606 → 8003 [PSH, ACK] Seq=3 Ack=1 Win=65536 Len=1 TSval=206339947 TSecr=206338405
9 20.621204810 127.0.0.1 127.0.0.1 TCP 66 8003 → 35606 [ACK] Seq=1 Ack=4 Win=65536 Len=0 TSval=206339947 TSecr=206339947
10 20.621547528 127.0.0.1 127.0.0.1 TCP 74 8003 → 35606 [PSH, ACK] Seq=1 Ack=4 Win=65536 Len=8 TSval=206339948 TSecr=206339947
11 20.621577897 127.0.0.1 127.0.0.1 TCP 66 35606 → 8003 [ACK] Seq=4 Ack=9 Win=65536 Len=0 TSval=206339948 TSecr=206339948
Frame 6: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 35606, Dst Port: 8003, Seq: 2, Ack: 1, Len: 1
Source Port: 35606
Destination Port: 8003
[Stream index: 0]
[TCP Segment Len: 1]
Sequence number: 2 (relative sequence number)
Sequence number (raw): 663745649
[Next sequence number: 3 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Acknowledgment number (raw): 3620577423
1000 .... = Header Length: 32 bytes (8)
Flags: 0x018 (PSH, ACK)
Window size value: 512
[Calculated window size: 65536]
[Window size scaling factor: 128]
Checksum: 0xfe29 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]
[Timestamps]
TCP payload (1 byte)
Data (1 byte)
0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00
0010 00 35 70 7f 40 00 40 06 cc 41 7f 00 00 01 7f 00
0020 00 01 8b 16 1f 43 27 8f f4 71 d7 cd a0 8f 80 18
0030 02 00 fe 29 00 00 01 01 08 0a 0c 4c 79 85 0c 4c
0040 71 6b 6b
Byte 66: Data (data.data)
Packets: 11 · Displayed: 11 (100.0%) · Dropped: 0 (0.0%)
Profile: Default

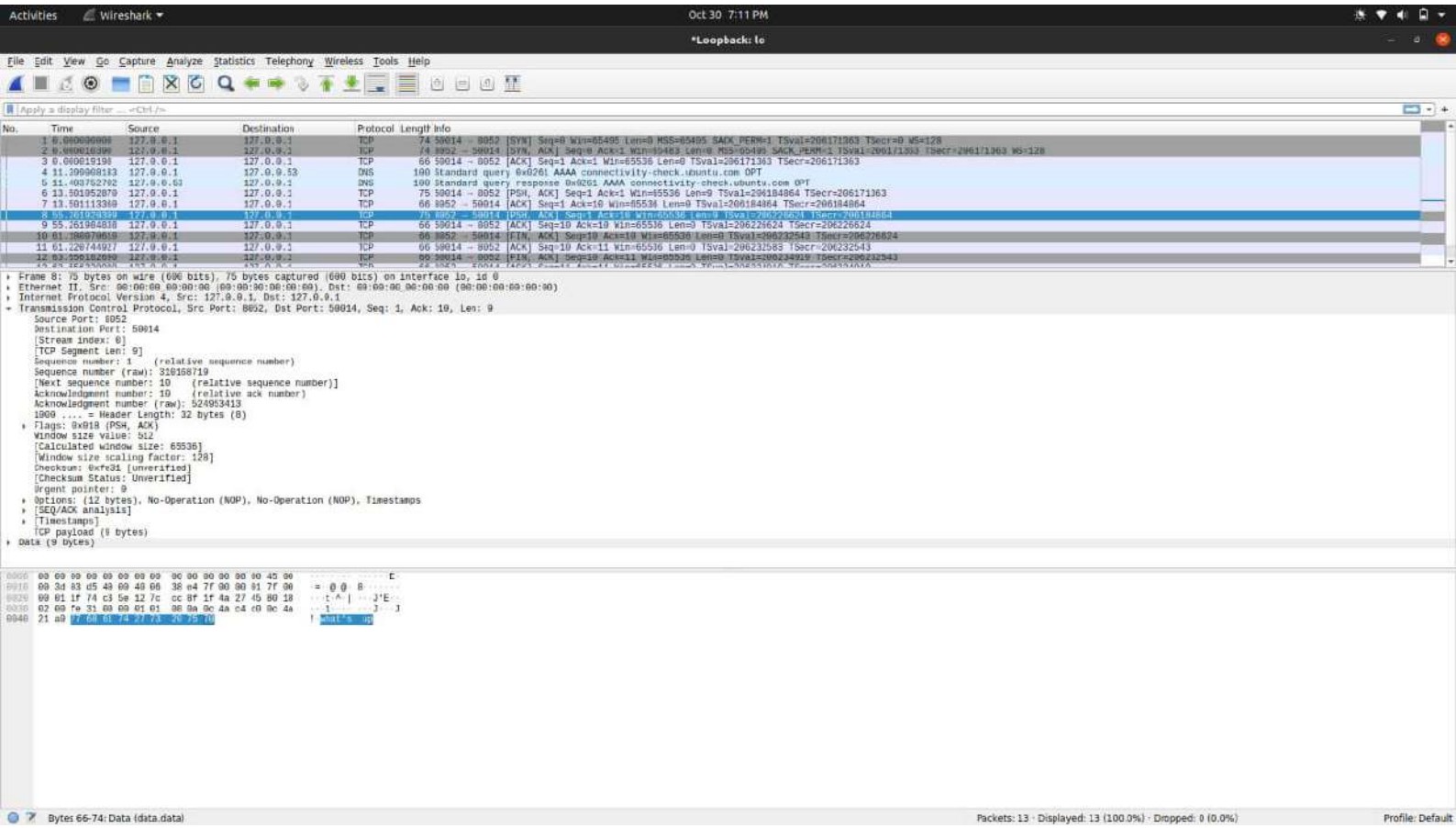

Activities
Wireshark
Oct 30 7:13 PM
*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 127.0.0.1 127.0.0.1 TCP 74 35606 → 8003 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=206319326 TSecr=0 WS=128
2 0.000010443 127.0.0.1 127.0.0.1 TCP 74 8003 → 35606 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=206319326 TSecr=206319326 WS=128
3 0.000019143 127.0.0.1 127.0.0.1 TCP 66 35606 → 8003 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=206319326 TSecr=206319326
4 17.037146537 127.0.0.1 127.0.0.1 TCP 67 35606 → 8003 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=1 TSval=206336363 TSecr=206319326
5 17.037221184 127.0.0.1 127.0.0.1 TCP 66 8003 → 35606 [ACK] Seq=1 Ack=2 Win=65536 Len=0 TSval=206336363 TSecr=206336363
6 19.079307586 127.0.0.1 127.0.0.1 TCP 67 35606 → 8003 [PSH, ACK] Seq=2 Ack=1 Win=65536 Len=1 TSval=206338405 TSecr=206336363
7 19.079361171 127.0.0.1 127.0.0.1 TCP 66 8003 → 35606 [ACK] Seq=1 Ack=3 Win=65536 Len=0 TSval=206338405 TSecr=206338405
8 29.621177845 127.0.0.1 127.0.0.1 TCP 67 35606 → 8003 [PSH, ACK] Seq=3 Ack=1 Win=65536 Len=1 TSval=206339947 TSecr=206338405
9 29.621204810 127.0.0.1 127.0.0.1 TCP 66 8003 → 35606 [ACK] Seq=1 Ack=4 Win=65536 Len=0 TSval=206339947 TSecr=206339947
10 29.621547658 127.0.0.1 127.0.0.1 TCP 74 8003 → 35606 [PSH, ACK] Seq=1 Ack=4 Win=65536 Len=8 TSval=206339948 TSecr=206339947
11 29.621577897 127.0.0.1 127.0.0.1 TCP 66 35606 → 8003 [ACK] Seq=4 Ack=9 Win=65536 Len=0 TSval=206339948 TSecr=206339948
Frame 10: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 8003, Dst Port: 35606, Seq: 1, Ack: 4, Len: 8
Source Port: 8003
Destination Port: 35606
[Stream index: 0]
[TCP Segment Len: 8]
Sequence number: 1 (relative sequence number)
Sequence number (raw): 3628577423
[Next sequence number: 9 (relative sequence number)]
Acknowledgment number: 4 (relative ack number)
Acknowledgment number (raw): 663745651
1000 .... = Header Length: 32 bytes (8)
Flags: 0x018 (PSH, ACK)
Window size value: 512
[Calculated window size: 65536]
[Window size scaling factor: 128]
Checksum: 0xfe30 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]
[Timestamps]
TCP payload (8 bytes)
Data (8 bytes)
0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00
0010 00 3c 95 c8 40 00 40 06 a6 f1 7f 00 00 01 7f 00
0020 00 01 1f 43 8b 16 d7 cd a0 8f 27 8f f4 53 80 18
0030 02 00 fe 30 00 00 01 01 08 0a 0c 4c 7f 6c 0c 4c
0040 7f 6b
wireshark_lo_20201030191135_q5waLu.pcapng
Packets: 11 · Displayed: 11 (100.0%) · Dropped: 0 (0.0%)
Profile: Default


Activities
Wireshark
Oct 30 7:13 PM
*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 127.0.0.1 127.0.0.1 TCP 74 35606 → 8003 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=206319326 TSecr=0 WS=128
2 0.000010443 127.0.0.1 127.0.0.1 TCP 74 8003 → 35606 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=206319326 TSecr=206319326 WS=128
3 0.000019143 127.0.0.1 127.0.0.1 TCP 66 35606 → 8003 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=206319326 TSecr=206319326
4 17.637146537 127.0.0.1 127.0.0.1 TCP 67 35606 → 8003 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=1 TSval=206336363 TSecr=206319326
5 17.637221164 127.0.0.1 127.0.0.1 TCP 66 8003 → 35606 [ACK] Seq=1 Ack=2 Win=65536 Len=0 TSval=206336363 TSecr=206336363
6 19.079307596 127.0.0.1 127.0.0.1 TCP 67 35606 → 8003 [PSH, ACK] Seq=2 Ack=1 Win=65536 Len=1 TSval=206338405 TSecr=206336363
7 19.079361171 127.0.0.1 127.0.0.1 TCP 66 8003 → 35606 [ACK] Seq=1 Ack=3 Win=65536 Len=0 TSval=206338405 TSecr=206338405
8 20.621177845 127.0.0.1 127.0.0.1 TCP 67 35606 → 8003 [PSH, ACK] Seq=3 Ack=1 Win=65536 Len=1 TSval=206339947 TSecr=206338405
9 20.621204810 127.0.0.1 127.0.0.1 TCP 66 8003 → 35606 [ACK] Seq=1 Ack=4 Win=65536 Len=0 TSval=206339947 TSecr=206339947
10 20.621547528 127.0.0.1 127.0.0.1 TCP 74 8003 → 35606 [PSH, ACK] Seq=1 Ack=4 Win=65536 Len=8 TSval=206339948 TSecr=206339947
11 20.621577897 127.0.0.1 127.0.0.1 TCP 66 35606 → 8003 [ACK] Seq=4 Ack=9 Win=65536 Len=0 TSval=206339948 TSecr=206339948
Frame 4: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 35606, Dst Port: 8003, Seq: 1, Ack: 1, Len: 1
Source Port: 35606
Destination Port: 8003
[Stream index: 0]
[TCP Segment Len: 1]
Sequence number: 1 (relative sequence number)
Sequence number (raw): 663745648
[Next sequence number: 2 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Acknowledgment number (raw): 3620577423
1000 .... = Header Length: 32 bytes (8)
Flags: 0x018 (PSH, ACK)
Window size value: 512
[Calculated window size: 65536]
[Window size scaling factor: 128]
Checksum: 0xfe29 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]
[Timestamps]
TCP payload (1 byte)
Data (1 byte)
0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00
0010 00 35 70 7e 40 00 40 06 cc 42 7f 00 00 01 7f 00
0020 00 01 8b 16 1f 43 27 8f f4 70 d7 cd a0 8f 80 18
0030 02 00 fe 29 00 00 01 01 08 0a 0c 4c 71 6b 0c 4c
0040 2e de 42
Byte 66: Data (data.data)
Packets: 11 · Displayed: 11 (100.0%) · Dropped: 0 (0.0%)
Profile: Default


Activities
Wireshark
Oct 30 7:13 PM
*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 127.0.0.1 127.0.0.1 TCP 74 35606 → 8003 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=206319326 TSecr=0 WS=128
2 0.000010441 127.0.0.1 127.0.0.1 TCP 74 8003 → 35606 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=206319326 TSecr=206319326 WS=128
3 0.000019143 127.0.0.1 127.0.0.1 TCP 66 35606 → 8003 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=206319326 TSecr=206319326
4 17.037146537 127.0.0.1 127.0.0.1 TCP 67 35606 → 8003 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=1 TSval=206336363 TSecr=206319326
5 17.037221164 127.0.0.1 127.0.0.1 TCP 66 8003 → 35606 [ACK] Seq=1 Ack=2 Win=65536 Len=0 TSval=206336363 TSecr=206336363
6 19.079307598 127.0.0.1 127.0.0.1 TCP 67 35606 → 8003 [PSH, ACK] Seq=2 Ack=1 Win=65536 Len=1 TSval=206338405 TSecr=206336363
7 19.079361171 127.0.0.1 127.0.0.1 TCP 66 8003 → 35606 [ACK] Seq=1 Ack=3 Win=65536 Len=0 TSval=206338405 TSecr=206338405
8 20.621177845 127.0.0.1 127.0.0.1 TCP 67 35606 → 8003 [PSH, ACK] Seq=3 Ack=1 Win=65536 Len=1 TSval=206339947 TSecr=206338405
9 20.621204810 127.0.0.1 127.0.0.1 TCP 66 8003 → 35606 [ACK] Seq=1 Ack=4 Win=65536 Len=0 TSval=206339947 TSecr=206339947
10 20.621547528 127.0.0.1 127.0.0.1 TCP 74 8003 → 35606 [PSH, ACK] Seq=1 Ack=4 Win=65536 Len=8 TSval=206339948 TSecr=206339947
11 20.621577897 127.0.0.1 127.0.0.1 TCP 66 35606 → 8003 [ACK] Seq=4 Ack=9 Win=65536 Len=0 TSval=206339948 TSecr=206339948
Frame 8: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 35606, Dst Port: 8003, Seq: 3, Ack: 1, Len: 1
Source Port: 35606
Destination Port: 8003
[Stream index: 0]
[TCP Segment Len: 1]
Sequence number: 3 (relative sequence number)
Sequence number (raw): 663745650
[Next sequence number: 4 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Acknowledgment number (raw): 3620577423
1000 .... = Header Length: 32 bytes (8)
Flags: 0x018 (PSH, ACK)
Window size value: 512
[Calculated window size: 65536]
[Window size scaling factor: 128]
Checksum: 0xfe29 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]
[Timestamps]
TCP payload (1 byte)
Data (1 byte)
0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00
0010 00 35 70 89 40 00 40 06 cc 40 7f 00 00 01 7f 00
0020 00 01 8b 16 1f 43 27 8f f4 72 d7 cd a0 8f 80 18
0030 02 00 fe 29 00 00 01 01 08 0a 0c 4c 7f 6b 0c 4c
0040 79 65
Byte 66: Data (data.data)
Packets: 11 · Displayed: 11 (100.0%) · Dropped: 0 (0.0%)
Profile: Default


Activities
Wireshark
Oct 30 7:15 PM
Capturing from Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 127.0.0.1 127.0.0.1 TCP 166 5043 → 42292 [PSH, ACK] Seq=1 Ack=1 Win=512 Len=100 TSval=206460594 TSecr=206433402
2 0.000082065 127.0.0.1 127.0.0.1 TCP 66 42292 → 5043 [ACK] Seq=1 Ack=101 Win=512 Len=0 TSval=206460595 TSecr=206460594
3 0.000381106 127.0.0.1 127.0.0.1 TCP 260 5043 → 42292 [PSH, ACK] Seq=101 Ack=1 Win=512 Len=194 TSval=206460595 TSecr=206460595
4 0.000396817 127.0.0.1 127.0.0.1 TCP 66 42292 → 5043 [ACK] Seq=1 Ack=295 Win=511 Len=0 TSval=206460595 TSecr=206460595
5 0.001396259 127.0.0.1 127.0.0.1 TCP 66 5043 → 42292 [FIN, ACK] Seq=295 Ack=1 Win=512 Len=0 TSval=206460595 TSecr=206460595
6 0.001773000 127.0.0.1 127.0.0.1 TCP 66 42292 → 5043 [FIN, ACK] Seq=1 Ack=295 Win=512 Len=0 TSval=206460596 TSecr=206460596
7 0.001815908 127.0.0.1 127.0.0.1 TCP 66 5043 → 42292 [ACK] Seq=296 Ack=2 Win=512 Len=0 TSval=206460596 TSecr=206460596
Frame 3: 260 bytes on wire (2080 bits), 260 bytes captured (2080 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 5043, Dst Port: 42292, Seq: 101, Ack: 1, Len: 194
Source Port: 5043
Destination Port: 42292
[Stream index: 0]
[TCP Segment Len: 194]
Sequence number: 101 (relative sequence number)
Sequence number (raw): 2179333638
[Next sequence number: 295 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Acknowledgment number (raw): 1493598587
1000 .... = Header Length: 32 bytes (8)
Flags: 0x018 (PSH, ACK)
Window size value: 512
[Calculated window size: 512]
[Window size scaling factor: -1 (unknown)]
Checksum: 0xfeea [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]
[Timestamps]
TCP payload (194 bytes)
Data (194 bytes)
Bytes 66-259: Data (data.data)
Packets: 7 · Displayed: 7 (100.0%)
Profile: Default


Activities
Wireshark
Oct 30 7:11 PM
*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
No. Time Source Destination Protocol Length Info
1 0.000000000 127.0.0.1 127.0.0.1 TCP 74 50014 → 8052 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=206171363 TSecr=0 WS=128
2 0.000016390 127.0.0.1 127.0.0.1 TCP 74 8052 → 50014 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=206171363 TSecr=206171363 WS=128
3 0.000019198 127.0.0.1 127.0.0.1 TCP 66 50014 → 8052 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=206171363 TSecr=206171363
4 11.399908183 127.0.0.1 127.0.0.53 DNS 100 Standard query 0x0261 AAAA connectivity-check.ubuntu.com OPT
5 11.403752702 127.0.0.53 127.0.0.1 DNS 100 Standard query response 0x0261 AAAA connectivity-check.ubuntu.com OPT
6 13.501052870 127.0.0.1 127.0.0.1 TCP 75 50014 → 8052 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=9 TSval=206184864 TSecr=206171363
7 13.501113388 127.0.0.1 127.0.0.1 TCP 66 8052 → 50014 [ACK] Seq=1 Ack=10 Win=65536 Len=0 TSval=206184864 TSecr=206184864
8 55.261920399 127.0.0.1 127.0.0.1 TCP 75 8052 → 50014 [PSH, ACK] Seq=1 Ack=10 Win=65536 Len=9 TSval=206226624 TSecr=206184864
9 55.261984838 127.0.0.1 127.0.0.1 TCP 66 50014 → 8052 [ACK] Seq=10 Ack=10 Win=65536 Len=0 TSval=206226624 TSecr=206226624
10 61.186070619 127.0.0.1 127.0.0.1 TCP 66 8052 → 50014 [FIN, ACK] Seq=10 Ack=10 Win=65536 Len=0 TSval=206232543 TSecr=206226624
11 61.220744927 127.0.0.1 127.0.0.1 TCP 66 50014 → 8052 [ACK] Seq=10 Ack=11 Win=65536 Len=0 TSval=206232583 TSecr=206232543
12 61.220858019 127.0.0.1 127.0.0.1 TCP 66 50014 → 8052 [FIN, ACK] Seq=10 Ack=11 Win=65536 Len=0 TSval=206234919 TSecr=206232543
Frame 6: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 50014, Dst Port: 8052, Seq: 1, Ack: 1, Len: 9
Source Port: 50014
Destination Port: 8052
[Stream index: 0]
[TCP Segment Len: 9]
Sequence number: 1 (relative sequence number)
Sequence number (raw): 524953484
[Next sequence number: 10 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Acknowledgment number (raw): 310168719
1000 .... = Header Length: 32 bytes (8)
Flags: 0x018 (PSH, ACK)
Window size value: 512
[Calculated window size: 65536]
[Window size scaling factor: 128]
Checksum: 0xfe31 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]
[Timestamps]
TCP payload (9 bytes)
Data (9 bytes)
0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00
0010 00 3d 75 8f 40 00 40 06 17 2a 7f 00 00 01 7f 00
0020 00 01 c3 5e 1f 74 1f 4a 27 3c 12 7c cc 1f 80 18
0030 02 00 fe 31 00 00 01 01 08 0a 0c 4a 21 a0 0c 49
0040 ec e3 48 65 79 20 54 68 65 72 65
Bytes 66-74: Data (data.data)
Packets: 13 · Displayed: 13 (100.0%) · Dropped: 0 (0.0%)
Profile: Default


Activities
Wireshark
Oct 30 7:11 PM
*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 127.0.0.1 127.0.0.1 TCP 74 50014 → 8052 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=206171363 TSecr=0 WS=128
2 0.000016390 127.0.0.1 127.0.0.1 TCP 74 8052 → 50014 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=206171363 TSecr=206171363 WS=128
3 0.000019198 127.0.0.1 127.0.0.1 TCP 66 50014 → 8052 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=206171363 TSecr=206171363
4 11.399908183 127.0.0.1 127.0.0.53 DNS 100 Standard query 0x0261 AAAA connectivity-check.ubuntu.com OPT
5 11.403762702 127.0.0.53 127.0.0.1 DNS 100 Standard query response 0x0261 AAAA connectivity-check.ubuntu.com OPT
6 13.501952870 127.0.0.1 127.0.0.1 TCP 75 50014 → 8052 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=9 TSval=206184864 TSecr=206171363
7 13.501113308 127.0.0.1 127.0.0.1 TCP 66 8052 → 50014 [ACK] Seq=1 Ack=10 Win=65536 Len=0 TSval=206184864 TSecr=206184864
8 55.261020389 127.0.0.1 127.0.0.1 TCP 75 8052 → 50014 [PSH, ACK] Seq=1 Ack=10 Win=65536 Len=9 TSval=206226624 TSecr=206184864
9 55.261984838 127.0.0.1 127.0.0.1 TCP 66 50014 → 8052 [ACK] Seq=10 Ack=10 Win=65536 Len=0 TSval=206226624 TSecr=206226624
10 61.186070619 127.0.0.1 127.0.0.1 TCP 66 8052 → 50014 [FIN, ACK] Seq=10 Ack=10 Win=65536 Len=0 TSval=206232543 TSecr=206226624
11 61.220744927 127.0.0.1 127.0.0.1 TCP 66 50014 → 8052 [ACK] Seq=10 Ack=11 Win=65536 Len=0 TSval=206232583 TSecr=206232543
12 61.220858019 127.0.0.1 127.0.0.1 TCP 66 50014 → 8052 [FIN, ACK] Seq=10 Ack=11 Win=65536 Len=0 TSval=206234019 TSecr=206232543
Frame 8: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 8052, Dst Port: 50014, Seq: 1, Ack: 10, Len: 9
Source Port: 8052
Destination Port: 50014
[Stream index: 0]
[TCP Segment Len: 9]
Sequence number: 1 (relative sequence number)
Sequence number (raw): 320168719
[Next sequence number: 10 (relative sequence number)]
Acknowledgment number: 10 (relative ack number)
Acknowledgment number (raw): 524953413
1000 .... = Header Length: 32 bytes (8)
Flags: 0x018 (PSH, ACK)
Window size value: 512
[Calculated window size: 65536]
[Window size scaling factor: 128]
Checksum: 0xfe31 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]
[Timestamps]
TCP payload (9 bytes)
Data (9 bytes)
0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00
0010 00 3d 83 d5 40 00 40 06 38 e4 7f 00 00 01 7f 00
0020 00 01 1f 74 c3 5e 12 7c cc 8f 1f 4a 27 45 80 18
0030 02 00 fe 31 00 00 01 01 08 0a 0c 4a c4 c0 0c 4a
0040 21 a0 77 68 61 74 27 73 20 75 70
Bytes 66-74: Data (data.data)
Packets: 13 · Displayed: 13 (100.0%) · Dropped: 0 (0.0%)
Profile: Default