Project – Part 4

1. Objective

1. Be familiar with the TCP protocol.
2. Implement a TCP transmission.

2. Overview

The Transport Control Protocol (TCP) belongs to the transport layer of the Protocol stack. TCP is a reliable, connection-oriented service.

The TCP services that are applicable to this phase of the project are listening and reading. The listen service sets up a port to receive connections. For the purpose of this project, you may make your port listen whenever it is not connected, so you don't really have to provide any interface to the application layer. The read service allows an application to read the bytes that have been delivered over the connection.

For this project, you will need to set up TCP port 5600 to listen for a connection (i.e. respond to a TCP packet with SYN = 1). After the connection is made, your program will have to receive a stream of bytes and hand them off to the application layer. You will need to write a tiny bit of application layer code to read the data from the port and write it to a file. Do not make this project more complicated than it needs to be.

3. Results

- Server IP Address: 192.168.1.20
- Client IP Address: 192.168.1.10

Program Output

SERVER (see code in Appendix 4.1)

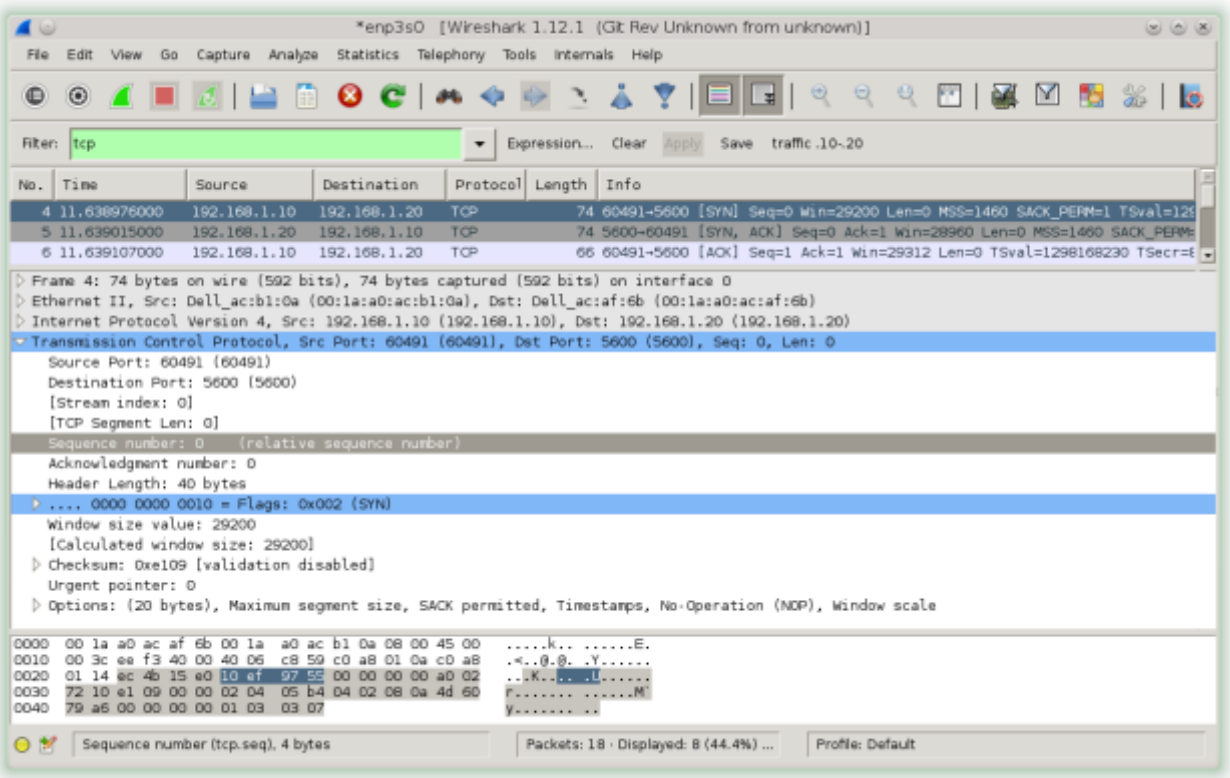
```
netlab20:~/Documents/jmeine # ./server 5600
message received:

'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.


"Beware the Jabberwock, my son!
The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
The frumious Bandersnatch!"


He took his vorpal sword in hand:
Long time the manxome foe he sought--
So rested he by the Tumtum tree,
And stood awhile in thought.


And, as in uffish thought he stood,
The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
And burbled as it came!


One two! One two! And through and through
The vorpal blade went snicker-snack!
He left it dead, and with its head
He went galumphing back.


"And hast thou slain the Jabberwock?
Come to my arms, my beamish boy!
O frabjous day! Callooh! Callay!"
He chortled in his joy.
```

CLIENT (see code in Appendix 4.2)

```
netlab10:~/Documents/jmeine # ./client 192.168.1.20
connect succeeded
socket closed
```

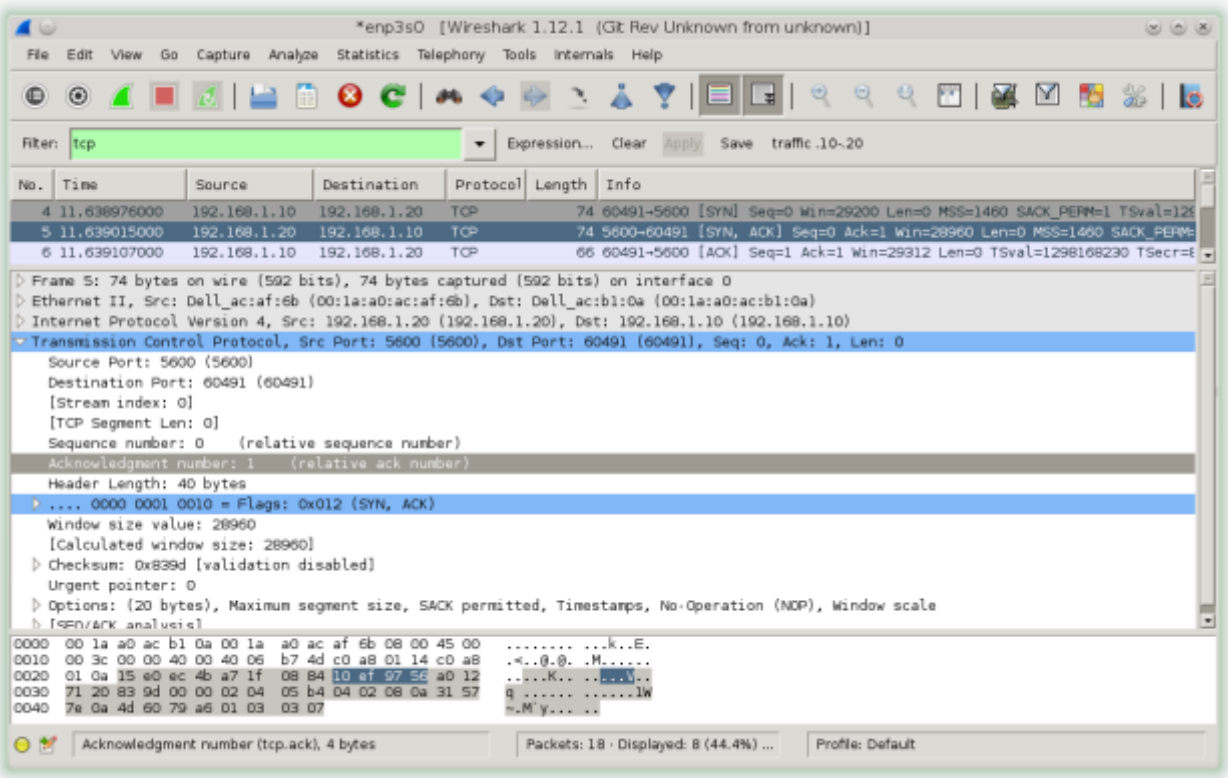
Network Capture

TCP SYN Request from Client Code
Source IP Address = 192.168.1.10
Destination IP Address = 192.168.1.20
SEQ = 0, ACK = 0



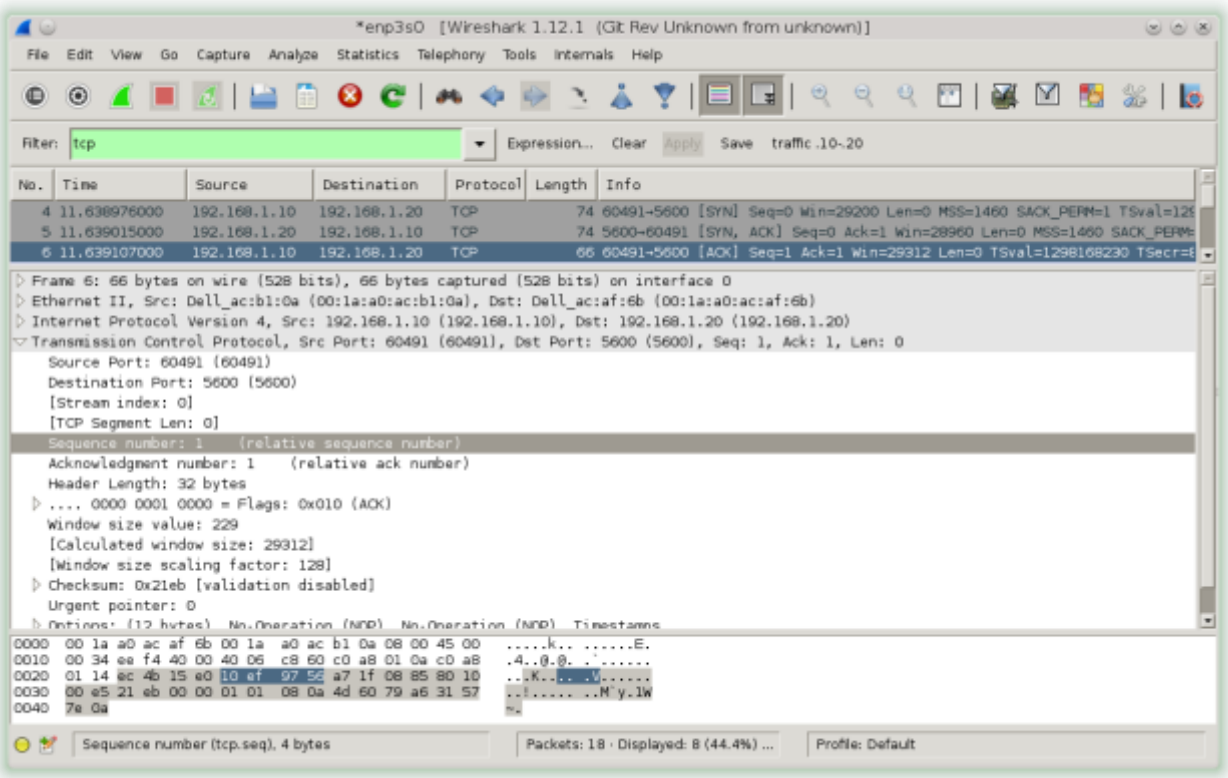
Network Capture

TCP SYN,ACK Reply from Server Code
Source IP Address = 192.168.1.20
Destination IP Address = 192.168.1.10
SEQ = 0, ACK = 1



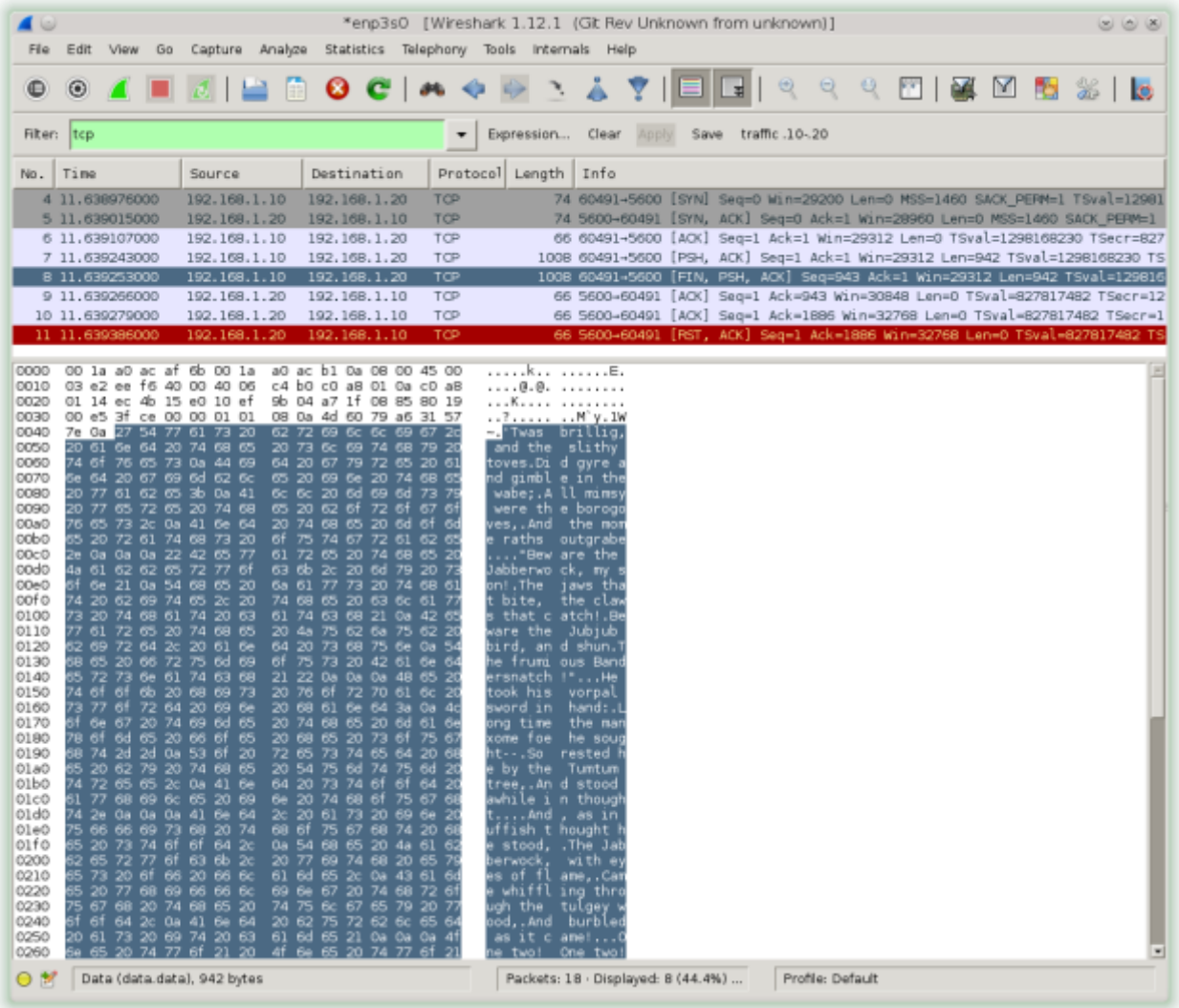
Network Capture

TCP ACK Reply from Client Code
Source IP Address = 192.168.1.10
Destination IP Address = 192.168.1.20
SEQ = 1, ACK = 1



Network Capture

Message Transmitted from Client Code
Source IP Address = 192.168.1.10
Destination IP Address = 192.168.1.20



4. Appendix

4.1 Program Code: Server

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

void error(const char *msg)
{
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[])
{
     int buffer_size = 810;
     int sockfd, newsockfd, portno;
     socklen_t clilen;
     char buffer[buffer_size];
     struct sockaddr_in serv_addr, cli_addr;
     int n;
     sockfd = socket(AF_INET, SOCK_STREAM, 0);
     if (sockfd < 0)
        error("opening socket failed");
     bzero((char *) &serv_addr, sizeof(serv_addr));
     portno = atoi(argv[1]);
     serv_addr.sin_family = AF_INET;
     serv_addr.sin_addr.s_addr = INADDR_ANY;
     serv_addr.sin_port = htons(portno);
     if (bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0)
        error("binding failed");
     listen(sockfd,5);
     clilen = sizeof(cli_addr);
     newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
     if (newsockfd < 0)
        error("accept failed");
     bzero(buffer,buffer_size);
     n = read(newsockfd,buffer,buffer_size-1);
     if (n < 0) error("read failed");
     printf("message received: \n\n%s\n",buffer);
     if (n < 0) error("write failed");
     close(newsockfd);
     close(sockfd);
     return 0;
}
```

4.2 Program Code: Client

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

//
// default IP address (if not specified on the command line
//
#define IP_ADDR "192.168.1.100"

//
// mangled data
//
char text_data[] = {
 32,  83, 112, 102, 116,  39, 101, 117, 110, 107, 107, 110,  96,  43,  39, 102,
105,  99,  39, 115, 111,  98,  39, 116, 107, 110, 115, 111, 126,  39, 115, 104,
113,  98, 116,  13,  67, 110,  99,  39, 96, 126, 117,  98,  39, 102, 105,  99,
 39,  96, 110, 106, 101, 107,  98,  39, 110, 105,  39, 115, 111,  98,  39, 112,
102, 101,  98,  60,  13,  70, 107, 107, 39, 106, 110, 106, 116, 126,  39, 112,
 98, 117,  98,  39, 115, 111,  98,  39, 101, 104, 117, 104,  96, 104, 113,  98,
116,  43,  13,  70, 105,  99,  39, 115, 111,  98,  39, 106, 104, 106,  98,  39,
117, 102, 115, 111, 116,  39, 104, 114, 115,  96, 117, 102, 101,  98,  41,  13,
 13,  13,  37,  69,  98, 112, 102, 117, 98,  39, 115, 111,  98,  39,  77, 102,
101, 101,  98, 117, 112, 104, 100, 108, 43,  39, 106, 126,  39, 116, 104, 105,
 38,  13,  83, 111,  98,  39, 109, 102, 112, 116,  39, 115, 111, 102, 115,  39,
101, 110, 115,  98,  43,  39, 115, 111, 98,  39, 100, 107, 102, 112, 116,  39,
115, 111, 102, 115,  39, 100, 102, 115, 100, 111,  38,  13,  69,  98, 112, 102,
117,  98,  39, 115, 111,  98,  39,  77, 114, 101, 109, 114, 101,  39, 101, 110,
117,  99,  43,  39, 102, 105,  99,  39, 116, 111, 114, 105,  13,  83, 111,  98,
 39,  97, 117, 114, 106, 110, 104, 114, 116,  39,  69, 102, 105,  99,  98, 117,
116, 105, 102, 115, 100, 111,  38,  37, 13,  13,  13,  79,  98,  39, 115, 104,
104, 108,  39, 111, 110, 116,  39, 113, 104, 117, 119, 102, 107,  39, 116, 112,
104, 117,  99,  39, 110, 105,  39, 111, 102, 105,  99,  61,  13,  75, 104, 105,
 96,  39, 115, 110, 106,  98,  39, 115, 111,  98,  39, 106, 102, 105, 127, 104,
106,  98,  39,  97, 104,  98,  39, 111, 98,  39, 116, 104, 114,  96, 111, 115,
 42,  42,  13,  84, 104,  39, 117,  98, 116, 115,  98,  99,  39, 111,  98,  39,
101, 126,  39, 115, 111,  98,  39,  83, 114, 106, 115, 114, 106,  39, 115, 117,
 98,  98,  43,  13,  70, 105,  99,  39, 116, 115, 104, 104,  99,  39, 102, 112,
111, 110, 107,  98,  39, 110, 105,  39, 115, 111, 104, 114,  96, 111, 115,  41,
 13,  13,  13,  70, 105,  99,  43,  39, 102, 116,  39, 110, 105,  39, 114,  97,
 97, 110, 116, 111,  39, 115, 111, 104, 114,  96, 111, 115,  39, 111,  98,  39,
116, 115, 104, 104,  99,  43,  13,  83, 111,  98,  39,  77, 102, 101, 101,  98,
117, 112, 104, 100, 108,  43,  39, 112, 110, 115, 111,  39,  98, 126,  98, 116,
 39, 104,  97,  39,  97, 107, 102, 106, 98,  43,  13,  68, 102, 106,  98,  39,
112, 111, 110,  97,  97, 107, 110, 105, 96,  39, 115, 111, 117, 104, 114,  96,
111,  39, 115, 111,  98,  39, 115, 114, 107,  96,  98, 126,  39, 112, 104, 104,
 99,  43,  13,  70, 105,  99,  39, 101, 114, 117, 101, 107,  98,  99,  39, 102,
116,  39, 110, 115,  39, 100, 102, 106, 98,  38,  13,  13,  13,  72, 105,  98,
 39, 115, 112, 104,  38,  39,  72, 105, 98,  39, 115, 112, 104,  38,  39,  70,
105,  99,  39, 115, 111, 117, 104, 114, 96, 111,  39, 102, 105,  99,  39, 115,
111, 117, 104, 114,  96, 111,  13,  83, 111,  98,  39, 113, 104, 117, 119, 102,
```

```
107,  39, 101, 107, 102,  99,  98,  39, 112,  98, 105, 115,  39, 116, 105, 110,
100, 108,  98, 117,  42, 116, 105, 102, 100, 108,  38,  13,  79,  98,  39, 107,
 98,  97, 115,  39, 110, 115,  39,  99, 98, 102,  99,  43,  39, 102, 105,  99,
 39, 112, 110, 115, 111,  39, 110, 115, 116,  39, 111,  98, 102,  99,  13,  79,
 98,  39, 112,  98, 105, 115,  39,  96, 102, 107, 114, 106, 119, 111, 110, 105,
 96,  39, 101, 102, 100, 108,  41,  13, 13,  13,  37,  70, 105,  99,  39, 111,
102, 116, 115,  39, 115, 111, 104, 114, 39, 116, 107, 102, 110, 105,  39, 115,
111,  98,  39,  77, 102, 101, 101,  98, 117, 112, 104, 100, 108,  56,  13,  68,
104, 106,  98,  39, 115, 104,  39, 106, 126,  39, 102, 117, 106, 116,  43,  39,
106, 126,  39, 101,  98, 102, 106, 110, 116, 111,  39, 101, 104, 126,  38,  13,
 72,  39,  97, 117, 102, 101, 109, 104, 114, 116,  39,  99, 102, 126,  38,  39,
 68, 102, 107, 107, 104, 104, 111,  38, 39,  68, 102, 107, 107, 102, 126,  38,
 37,  13,  79,  98,  39, 100, 111, 104, 117, 115, 107,  98,  99,  39, 110, 105,
 39, 111, 110, 116,  39, 109, 104, 126, 41,  13,  13,  13,  32,  83, 112, 102,
116,  39, 101, 117, 110, 107, 107, 110, 96,  43,  39, 102, 105,  99,  39, 115,
111,  98,  39, 116, 107, 110, 115, 111, 126,  39, 115, 104, 113,  98, 116,  13,
 67, 110,  99,  39,  96, 126, 117,  98, 39, 102, 105,  99,  39,  96, 110, 106,
101, 107,  98,  39, 110, 105,  39, 115, 111,  98,  39, 112, 102, 101,  98,  60,
 13,  70, 107, 107,  39, 106, 110, 106, 116, 126,  39, 112,  98, 117,  98,  39,
115, 111,  98,  39, 101, 104, 117, 104, 96, 104, 113,  98, 116,  43,  13,  70,
105,  99,  39, 115, 111,  98,  39, 106, 104, 106,  98,  39, 117, 102, 115, 111,
116,  39, 104, 114, 115,  96, 117, 102, 101,  98,  41,  13,  13,  13
};

sockaddr sa;
sockaddr_in *sin = (sockaddr_in *) &sa;
int sk;

int main(int argc, char *argv[])
{
   sk = socket(PF_INET, SOCK_STREAM, 0);
   if ( sk < 0 ) return -1;
   memset(&sa,0,sizeof(sa));
   sin->sin_family = PF_INET;
   sin->sin_port = htons(5600);
   if ( argc<2 || inet_pton(PF_INET, argv[1], &(sin->sin_addr)) <= 0 )
      inet_pton(PF_INET, IP_ADDR, &(sin->sin_addr));
   if ( connect(sk, &sa, sizeof(sa)) < 0 )
   {
      printf("connect failed\n");
      return 0;
   }
   printf("connect succeeded\n");
   for ( int i=0; i<sizeof(text_data); ++i )
      text_data[i] ^= 7;
   // send the data
   sendto(sk,text_data,sizeof(text_data),0,&sa,sizeof(sockaddr_in));
   text_data[sizeof(text_data)-54] = 122;
   // send it twice to make network problem more interesting
   sendto(sk,text_data,sizeof(text_data),0,&sa,sizeof(sockaddr_in));
   close(sk);
   printf("socket closed\n");
}
```