Daniel Kass
Lab 6
Networking II (CE4960)
May 8, 2014

Experience with Lab

My Experience with this lab went pretty well I had no real problems I was able to just keep moving and finish the lab up without to much hassle. The one problem I did have was with broadcasting the UDP packet, but I was able to figure that one out with a few Internet searches.

Lab 6

```
// Simple UDP echo client.  Reads a single line from STDIN and
// sends to server, including null-terminator

// Compile with:  gcc -o udpechoclient udpechoclient.c

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

#define MAX_MESSAGE 1000
#define HTML_RECEIVE 65000

char* getData(int);

int main(int argc, char **argv) {

	// locals
	int sock;
	struct sockaddr_in server;
	struct hostent *hp;
	int broadcast = 1;
	char* address = "155.92.79.255";
	//char* address = "192.168.43.255";
	//char* address = "10.160.31.255";
	int interval = 0;

	// check command-line args
	if (argc != 3) {
		printf("Usage: <port number> <time interval (minutes)>\n");
		exit(0);
	}

	interval = atoi(argv[2]);
	interval = interval * 60; // convert to seconds
	if (interval == 0) {
		interval = 60;
```

```
}

// create socket
// IP protocol family (PF_INET)
// UDP (SOCK_DGRAM)

if ((sock = socket(PF_INET, SOCK_DGRAM, 0)) < 0) {
      perror("Error creating socket");
      exit(1);
}

// UDP  Using UDP we don't need to call bind unless we
// want to specify a "source" port number.  We really
// do not care - server will reply to whatever port we
// are given

// Make a sockaddr of the server
// address family is IP  (AF_INET)
// server IP address is found by calling gethostbyname with the
// name of the server (entered on the command line)
// note, if an IP address is provided, that is OK too

server.sin_family = AF_INET;

if ((hp = gethostbyname(address)) == 0) {
      perror("Invalid or unknown host");
      exit(1);
}

// copy IP address into address structure
memcpy(&server.sin_addr.s_addr, hp->h_addr, hp->h_length);

// establish the server port number - we must use network byte order!
unsigned short port;

if (sscanf(argv[1], "%hu", &port) != 1) {
      perror("Error parsing port");
      exit(0);
}

// this call is what allows broadcast packets to be sent:
if (setsockopt(sock, SOL_SOCKET, SO_BROADCAST, &broadcast, sizeof broadcast)
            == -1) {
      perror("setsockopt (SO_BROADCAST)");
      exit(1);
}

server.sin_port = htons(port);

// ready to send
printf("Ready to broadcast on port %hu\n", port);

char* buffer = calloc(MAX_MESSAGE, sizeof(char));
int size_to_send;
int size_sent;

while (1) {
      strcpy(buffer,"The Top Story on Reddit.com is:");
```

```
            strcat(buffer,getData(1));
            strcat(buffer,"\n");
            size_to_send = strlen(buffer);

            printf("Broadcasting Top Story\n");
            // send to server
            size_sent = sendto(sock, buffer, size_to_send, 0,
                        (struct sockaddr*) &server, sizeof(server));

            if (size_sent < 0) {
                  perror("Error sending data");
                  exit(1);
            }

            //clear buffer
            //memset(buffer, 0, MAX_MESSAGE);
            sleep(interval);
      }

      return (0);
}

char* getData(int entry) {

      int sock;
      struct sockaddr_in server;
      struct hostent *hp;
      char* website = "www.reddit.com";
      char *temp2 = calloc(500, sizeof(char));

      // create socket
      // IP protocol family (PF_INET)
      // UDP (SOCK_STREAM)

      if ((sock = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
            perror("Error creating socket");
            exit(1);
      }

      // TCP - client will be more like server than UDP example.
      // Since we are going active right away, we can skip bind
      // if we like.

      // Make a sockaddr of the server
      // address family is IP  (AF_INET)
      // server IP address is found by calling gethostbyname with the
      // name of the server (entered on the command line)
      // note, if an IP address is provided, that is OK too

      server.sin_family = AF_INET;

      if ((hp = gethostbyname(website)) == 0) {
            perror("Invalid or unknown host");
            exit(1);
      }

      // copy IP address into address structure
      memcpy(&server.sin_addr.s_addr, hp->h_addr, hp->h_length);
```

```
// establish the server port number - we must use network byte order!
unsigned short port = 80;

server.sin_port = htons(port);

if (connect(sock, (struct sockaddr *) &server, sizeof(server)) < 0) {
      perror("Error calling connect");
      exit(-1);
}

char receive[HTML_RECEIVE];
char* get = "GET / HTTP/1.0\nHost: www.reddit.com\n\n\0";
int insert;
int size_to_send;
int size_sent;
int bytes_read;

size_to_send = strlen(get);

// send to server
size_sent = send(sock, get, size_to_send, 0); // send null

printf("Getting Information\n");

if (size_sent < 0) {
      perror("Error sending data");
      exit(1);
}
while (1) {
      // reset buffer counter
      insert = 0;

      receive[0] = '\0'; // guarantee a null here to break out on a
      // clear buffer
      memset(receive, 0, HTML_RECEIVE);
      // copy null-terminated string into buffer
      // go until we get a null terminator or and endline
      //while(1){
      while (1) {
            if ((bytes_read = read(sock, &receive[insert], 1)) < 0) {
                  perror("Error calling read");
                  exit(-1);
            }
            if(insert == HTML_RECEIVE){
                  break;
            }
            // check for disconnect
            if (bytes_read == 0)
                  break;

            if (receive[insert] == '\0') {
                  //buffer[insert + 1] = '\0';
                  //bytes_read = insert; // will == strlen
                  break;
            }
            insert++;
            //printf(".\n");
```

```
            }

            //got the webpage!
            receive[HTML_RECEIVE] = '\0';

            //now to get the information i want
            //ok so we need to look for text "rank">1<  and then
            //after that we will get the title of the top story on
            //reddit for the time
            int window = 1000;
            char *temp = calloc(window, sizeof(char));

            temp = strstr(receive,"rank\">1<");
            temp[window] = '\0';
            //printf("%s\n",temp);

            temp2 = strstr(temp,"dex=\"1");
            temp2[500] = '\0';
            int i;
            for(i = 0; i < strlen(temp2); i++){
                  if(temp2[i] == '<'){
                        temp2[i] = '\0';
                        break;
                  }
            }
            temp2 = strstr(temp2,">");
            temp2[0] = '\n';

            //printf("%s\n",temp2);
            close(sock);
            break;
      }
      return temp2;

}
```

UDP_Client

```
// Simple UDP echo server
// Compile with:  gcc -o udpechoserver udpechoserver.c

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


// Max message to echo
#define MAX_MESSAGE     1000

/* server main routine */
```

```c
int main(int argc, char** argv) {

      // locals
      unsigned short port = 8675; // default port
      int sock; // socket descriptor

      // Was help requested?  Print usage statement
      if (argc > 1 && ((!strcmp(argv[1],"-?"))||(!strcmp(argv[1],"-h"))))
      {
            printf("\nUsage: udpechoserver [-p port] port is the requested \
 port that the server monitors.  If no port is provided, the server \
 listens on port 8675.\n\n");
            exit(0);
      }

      // get the port from ARGV
      if (argc > 1 && !strcmp(argv[1],"-p"))
      {
            if (sscanf(argv[2],"%hu",&port)!=1)
            {
                  perror("Error parsing port option");
                  exit(0);
            }
      }

      // ready to go
      printf("UDP broadcast client configuring on port: %d\n",port);

      // for UDP, we want IP protocol domain (PF_INET)
      // and UDP transport type (SOCK_DGRAM)
      // no alternate protocol - 0, since we have already specified IP

      if ((sock = socket( PF_INET, SOCK_DGRAM, 0 )) < 0)
      {
            perror("Error on socket creation");
            exit(1);
      }

      // establish address - this is the server and will
      // only be listening on the specified port
      struct sockaddr_in sock_address;

      // address family is AF_INET
      // our IP address is INADDR_ANY (any of our IP addresses)
    // the port number is per default or option above

      sock_address.sin_family = AF_INET;
      sock_address.sin_addr.s_addr = htonl(INADDR_ANY);
      sock_address.sin_port = htons(port);

      // we must now bind the socket descriptor to the address info
      if (bind(sock, (struct sockaddr *) &sock_address, sizeof(sock_address))<0)
      {
            perror("Problem binding");
            exit(0);
      }

      // go into forever loop and echo whatever message is received
```

```
  // to console and back to source
  char* buffer = calloc(MAX_MESSAGE,sizeof(char));
  int bytes_read;
struct sockaddr_in from;
  int from_len;

while (1) {

        from_len = sizeof(from);

        // read datagram and put into buffer
        bytes_read = recvfrom( sock ,buffer, MAX_MESSAGE,
                    0, (struct sockaddr *)&from, &from_len);

        // print info to console
        //printf("Received message from %s port %d\n",
        //    inet_ntoa(from.sin_addr), ntohs(from.sin_port));
        printf("%s",buffer);
        if (bytes_read < 0)
        {
              perror("Error receiving data");
        }

        // clear buffer
        memset(buffer, 0, MAX_MESSAGE);
}

  // will never get here
  return(0);
}
```