CENG 421 IP RESEARCH

Esin Sanem İMAMOGLU

Student ID:240206006

- In this assignment, I created a simple UDP server and client which sends date and time.
- Initially, in server code, I added libraries according to tha functions that I used in the code. I found out some of them with “man (function)” comment whenever there ocurred an error.

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

#define MAXBUF 1024
```

- Next step was defining variables and strings with their lengths.

```
int main(int argc, char* argv[]) {
int udpSocket,r,bn,port;
int returnStatus = 0;
int addrlen = 0;
struct sockaddr_in udpServer, udpClient;
socklen_t cc;
char buf[MAXBUF];
```

- After that, here is checked whether number of arguments are true or not.

```
if (argc < 2) {
fprintf(stderr, "Usage: %s <port>\n", argv[0]);
exit(1);
}
```

- This step is about creating a UDP socket.

```
udpSocket = socket(AF_INET, SOCK_DGRAM, 0);
if (udpSocket == -1) {
fprintf(stderr, "Could not create a socket!\n"); exit(1);
```

```
}
 else {
 printf("Socket created.\n");
}
```

- Taking the port number from the client and creating server address and port.

```
printf("\n Enter the port no:");
scanf("%d",&port);
printf("The port no is:%d\n",port);

udpServer.sin_family = AF_INET;
udpServer.sin_addr.s_addr = htonl(INADDR_ANY);
udpServer.sin_port = htons(atoi(argv[1]));
```

- Binding the socket to the server address.

```
bn = bind(udpSocket, (struct sockaddr*)&udpServer,  sizeof(udpServer));
if (bn == 0) {
fprintf(stderr, "Bind completed!\n");
 }
 else {
fprintf(stderr, "Could not bind to address!\n");
close(udpSocket);
exit(1);
}
```

- By using infinite loop, before transmitting, message is set up.

```
while (1) {
addrlen = sizeof(udpClient);
bn= recvfrom(udpSocket, buf, MAXBUF, 0, (struct sockaddr*)&udpClient, &addrlen);

if (bn== -1) {
fprintf(stderr, "Could not receive message!\n");
}
else {
printf("Received: %s\n", buf);
```

- Sending positive feedback if the message transmitted.

```
strcpy(buf, "OK");
bn = sendto(udpSocket, buf, strlen(buf)+1, 0, (struct sockaddr*)&udpClient,
sizeof(udpClient));
if (bn == -1) {
 fprintf(stderr, "Could not send confirmation!\n");
 }
 else {
printf("Confirmation sent.\n");
}}}

cc=sizeof(udpClient);
```

```
r=recvfrom(udpSocket,buf,sizeof(buf),0,(struct sockaddr*)&udpClient,&cc );
buf[r]=0;
```

- By using tools from time.h library , time and date are determined.

```
time_t tt;
tt = time(NULL);
snprintf(buf,sizeof(buf),"%24s\r\n",ctime(&tt));
sendto(udpSocket,buf,sizeof(buf),0,(struct sockaddr*)&udpClient,sizeof(udpClient));

 return 0;
}
```

- In client code, I followed similar steps with my servent code.

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <string.h>
#include <stdio.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#define MAXBUF 1024
int main(int argc, char* argv[]) {
int udpSocket,returnStatus,addrlen,port,r;
struct sockaddr_in udpClient, udpServer;
char buf[MAXBUF];
if (argc < 3) {
fprintf(stderr, "Usage: %s <ip address> <port>\n", argv[0]);
 exit(1);
 }
/* create a socket */
udpSocket = socket(AF_INET, SOCK_DGRAM, 0);
if (udpSocket == -1) {
 fprintf(stderr, "Could not create a socket!\n");
exit(1);
}
else {
printf("Socket created.\n");
 }
udpClient.sin_family = AF_INET;
udpClient.sin_addr.s_addr = INADDR_ANY;
udpClient.sin_port = htons(atoi(argv[2]));;

while(1){
```

```
returnStatus = sendto(udpSocket, buf, strlen(buf)+1, 0, (struct sockaddr*)&udpServer,
sizeof(udpServer));
if (returnStatus == -1) {
fprintf(stderr, "ERROR!\n");
}
 else {
printf("\nMessage sent.\n");

/* message sent: look for confirmation */
addrlen = sizeof(udpServer);
r= recvfrom(udpSocket, buf, sizeof(buf), 0, (struct sockaddr*)&udpServer, &addrlen);
buf[r]=0;
printf("\n The time received from the server is :%s\n",buf);

}
}

}
```