UDP Socket Programming

**What is UDP?**

UDP is a connectionless, unreliable, datagram protocol unlike the connection-oriented, reliable byte stream provided by TCP.

**How does UDP work?**

In a UDP client/server, the client does not establish a connection with the server. Instead, the client just sends a datagram to the server using the *sendto* function, which requires the address of the destination (the server) as a parameter. Similarly, the server does not accept a connection from a client. Instead, the server just calls the recvfrom function, which waits until data arrives from some client and recvfrom returns the protocol address of the client, along with the datagram, so the server can send a response to the correct client.

**What are Sockets?**

A process on one machine can use sockets to communicate with a process on another, which allows for client/server systems that are distributed across a network. Sockets may also be used between processes on the same machine.

***Problem description***:

A simple client-server program where the server and client can send messages to each other using UDP sockets.

***Functions used***:

1) int **socket**(int *domain*, int *type*, int *protocol*);

Returns socket file descriptor.

Arguments :   
domain – Specifies the communication  
domain ( AF\_INET for IPv4/ AF\_INET6 for IPv6 )  
type – Type of socket to be created  
( SOCK\_STREAM for TCP / SOCK\_DGRAM for UDP )  
protocol – Protocol to be used by socket.  
0 means use default protocol for the address family.

2) int **bind**(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen)

The bind function assigns a local protocol address to a socket. With the Internet protocols, the protocol address is the combination of either a 32-bit IPv4 address or a 128-bit IPv6 address, along with a 16-bit UDP port number.

Arguments :   
sockfd – File descriptor of socket to be binded  
addr – Structure in which address to be binded to is specified  
addrlen – Size of addr structure

3) ssize\_t **sendto**(int sockfd, const void \*buf, size\_t len, int flags,

const struct sockaddr \*dest\_addr, socklen\_t addrlen)

Send a message on the socket.

Arguments :   
sockfd – File descriptor of socket  
buf – Buffer containing the data to be sent  
len – Size of buffer  
flags – Bitwise OR of flags to modify socket behaviour  
dest\_addr – Structure containing address of destination  
addrlen – Size of dest\_addr structure

4) ssize\_t **recvfrom**(int sockfd, void \*buf, size\_t len, int flags,

struct sockaddr \*src\_addr, socklen\_t \*addrlen)

Receive a message from the socket.

Arguments :   
sockfd – File descriptor of socket  
buf – Buffer in which to receive data  
len – Size of buffer  
flags – Bitwise OR of flags to modify socket behaviour  
src\_addr – Structure containing source address is returned  
addrlen – Variable in which size of src\_addr structure is returned

***C++ Program***

**1) server.cpp**

#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>

#include <iostream>

#include <vector>

#define MAX 255

using namespace std;

int main(int argc, char const \*argv[])

{

if(argc < 2){

printf("Port number not provided. \n");

return 0;

}

struct sockaddr\_in servaddr, cliaddr;

int port\_no = atoi(argv[1]);

char buffer[MAX]; //holds the string received from client

char str[MAX]; //holds the string to be sent to the client

// Get a socket file descriptor

int sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

if(sockfd < 0){ //returns non negative integer if successful

perror("Failed to create a socket");

return 0;

}

memset(&servaddr, 0, sizeof(servaddr)); // initialize both to 0

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

servaddr.sin\_family = AF\_INET; // IPv4

servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

servaddr.sin\_port = htons(port\_no);

// Bind the socket to the protocol address (IP + port number)

if(bind(sockfd, (struct sockaddr\*) &servaddr, sizeof(servaddr)) < 0){

perror("Bind failed");

return 0;

}

int n;

socklen\_t len = 20;

while(1){

n = recvfrom(sockfd, (char \*)buffer, MAX,

0, (struct sockaddr \*) &cliaddr,

&len);

buffer[n] = '\0';

printf("Client : %s\n", buffer);

fgets(str, MAX, stdin);

sendto(sockfd, (const char \*)str, strlen(str),

0, (const struct sockaddr \*) &cliaddr,

len);

}

close(sockfd);

return 0;

}

2) **client.cpp**

#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>

#include <iostream>

#define MAX 255

using namespace std;

int main(int argc, char const \*argv[]) {

if(argc < 2){

printf("Port number not provided. \n");

return 0;

}

char buffer[MAX];

char str[MAX];

int port\_no = atoi(argv[1]);

struct sockaddr\_in servaddr;

int sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

if(sockfd < 0){ //returns non negative integer if successful

perror("Failed to create a socket");

return 0;

}

//We do not bind the socket to a port for a client because it does not matter

//which port you use to send the data

//The port that receives the data is important so we bind the socket in

// the server program

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

// Filling server information

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = INADDR\_ANY;

servaddr.sin\_port = htons(port\_no);

int n;

socklen\_t len;

while(1){

fgets(str, MAX, stdin);

sendto(sockfd, (const char \*)str, strlen(str),

0, (const struct sockaddr \*) &servaddr,

sizeof(servaddr));

n = recvfrom(sockfd, (char \*)buffer, MAX,

0, (struct sockaddr \*) &servaddr,

&len);

buffer[n] = '\0';

printf("Server : %s\n", buffer);

}

close(sockfd);

return 0;

}