

Lab work:

Change the server program by creating a list of IP addresses from which you do not want to receive any message. Whenever a message is sent from a host from the list you just created, reject the message sending back a message to the sender with content "MESSAGE REJECTED"

```

/* udpserver.c */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
int main()
{
    int sock;
    int addr_len, bytes_read;
    char recv_data[1024];
    struct sockaddr_in server_addr , client_addr;

    if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
        perror("Socket");
        exit(1);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(5000);
    server_addr.sin_addr.s_addr = INADDR_ANY;
    bzero(&(server_addr.sin_zero), 8);

    if (bind(sock, (struct sockaddr *)&server_addr,
        sizeof(struct sockaddr)) == -1)
    {
        perror("Bind");
        exit(1);
    }

    addr_len = sizeof(struct sockaddr);

    printf("\nUDP Server Waiting for client on port 5000");
    fflush(stdout);

    while (1)
    {
        bytes_read = recvfrom(sock, recv_data, 1024, 0,
            (struct sockaddr *)&client_addr, &addr_len);

        recv_data[bytes_read] = '\0';

        printf("\n(%s , %d) said : ", inet_ntoa(client_addr.sin_addr),
            ntohs(client_addr.sin_port));

        printf("%s", recv_data);
        fflush(stdout);
    }
    return 0;
}

```

```

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

int main()
{
int sock;
struct sockaddr_in server_addr;
struct hostent *host;
char send_data[1024];

host= (struct hostent *) gethostbyname((char *)"127.0.0.1");

if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
{
perror("socket");
exit(1);
}

server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(5000);
server_addr.sin_addr = *((struct in_addr *)host->h_addr);
bzero(&(server_addr.sin_zero),8);

while (1)
{

printf("Type Something (q or Q to quit):");
gets(send_data);

if ((strcmp(send_data , "q") == 0) || strcmp(send_data , "Q") == 0)
break;

else
sendto(sock, send_data, strlen(send_data), 0,
(struct sockaddr *)&server_addr, sizeof(struct sockaddr));

}

}

```