**Networks Lab: Broadcasting, Multicasting**

**V Venkataraman 106118106**

**server.c**

#include <unistd.h>

#include <stdio.h>

#include <sys/socket.h>

#include <stdlib.h>

#include <netinet/in.h>

#include <string.h>

#define PORT 8080

typedef enum{ DATA,ACK }   MSGKIND;

struct timeval timeout;

struct MESSAGE

{

    MSGKIND type;

    int seq;

    unsigned int len;

    char msg[100];

    // int parity;

};

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

{

    timeout.tv\_sec = 1;

    timeout.tv\_usec = 0;

    int server\_fd, new\_socket[5], valread;

    struct sockaddr\_in address;

    int opt = 1;

    int addrlen = sizeof(address);

    char buffer[1024] = {0};

    if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == 0)

    {

        perror("socket failed");

        exit(EXIT\_FAILURE);

    }

    if (setsockopt(server\_fd, SOL\_SOCKET, SO\_RCVTIMEO, (char \*)&timeout, sizeof(timeout)) < 0)

    {

        perror("setsockopt");

        exit(EXIT\_FAILURE);

    }

    address.sin\_family = AF\_INET;

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

    address.sin\_port = htons( PORT );

    if (bind(server\_fd, (struct sockaddr \*)&address,

                                 sizeof(address))<0)

    {

        perror("bind failed");

        exit(EXIT\_FAILURE);

    }

    if (listen(server\_fd, 5) < 0)

    {

        perror("listen");

        exit(EXIT\_FAILURE);

    }

    for(int j=0;j<5;j++)

    {

        if ((new\_socket[j] = accept(server\_fd, (struct sockaddr \*)&address,  (socklen\_t\*)&addrlen))<0)

        {

            perror("accept");

            exit(EXIT\_FAILURE);

        }

        if (setsockopt(new\_socket[j], SOL\_SOCKET, SO\_RCVTIMEO, (char \*)&timeout, sizeof(timeout)) < 0)

        {

            perror("setsockopt");

            exit(EXIT\_FAILURE);

        }

    }

    printf("All 5 client sockets accepted!\n");

    int clients[5];

    int n;

    int count=0;

    while(1)

    {

        count++;

        struct MESSAGE\* Message = (struct MESSAGE\*) malloc(sizeof(struct MESSAGE));

        Message->type = DATA;

        char news[100]="KKR beat CSK by 10 runs";

        printf("Msg%d: %s\n",count,news);

        Message->len = strlen(news);

        strcpy(Message->msg, news);

        Message->seq = count;

        scanf("%d",&n);

        if(n==0)

        {

            printf("Closing server...\n");

            break;

        }

        else if(n==5)

        {

            printf("Broadcasting...\n");

            sleep(1);

            for(int i=0;i<n;i++)

            {

                send(new\_socket[i],(void\*)Message, sizeof(struct MESSAGE), 0);

            }

        }

        else

        {

            for(int i=0;i<n;i++)

            {

                scanf("%d",&clients[i]);

            }

            printf("Multicasting to: ");

            for(int i=0;i<n;i++)

                printf("%d ",clients[i]);

            printf("\n");

            sleep(1);

            for(int i=0;i<n;i++)

            {

                send(new\_socket[clients[i]],(void\*)Message, sizeof(struct MESSAGE), 0);

            }

        }

    }

    for(int i=0;i<5;i++)

        close(new\_socket[i]);

    return 0;

}

**client.c**

#include <stdio.h>

#include <stdlib.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <string.h>

#define PORT 8080

typedef enum{ DATA,ACK }   MSGKIND;

struct MESSAGE

{

    MSGKIND type;

    int seq;

    unsigned int len;

    char msg[100];

    int parity;

};

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

{

    int sock = 0, valread;

    struct sockaddr\_in serv\_addr;

    char buffer[1024] = {0};

    if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

    {

        printf("\n Socket creation error \n");

        return -1;

    }

    serv\_addr.sin\_family = AF\_INET;

    serv\_addr.sin\_port = htons(PORT);

    if(inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr)<=0)

    {

        printf("\nInvalid address/ Address not supported \n");

        return -1;

    }

    if (connect(sock, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0)

    {

        printf("\nConnection Failed \n");

        return -1;

    }

    while(1)

    {

                        //send(sock,(void\*)Acknowledge, sizeof(struct MESSAGE), 0);

                    struct MESSAGE\* Message = (struct MESSAGE\*) malloc(sizeof(struct MESSAGE));

                    if(recv(sock, Message, sizeof(struct MESSAGE), 0) > 0)

                    {

                        printf("Seq:%d Message recvd: %s\n",Message->seq,Message->msg);

                    }

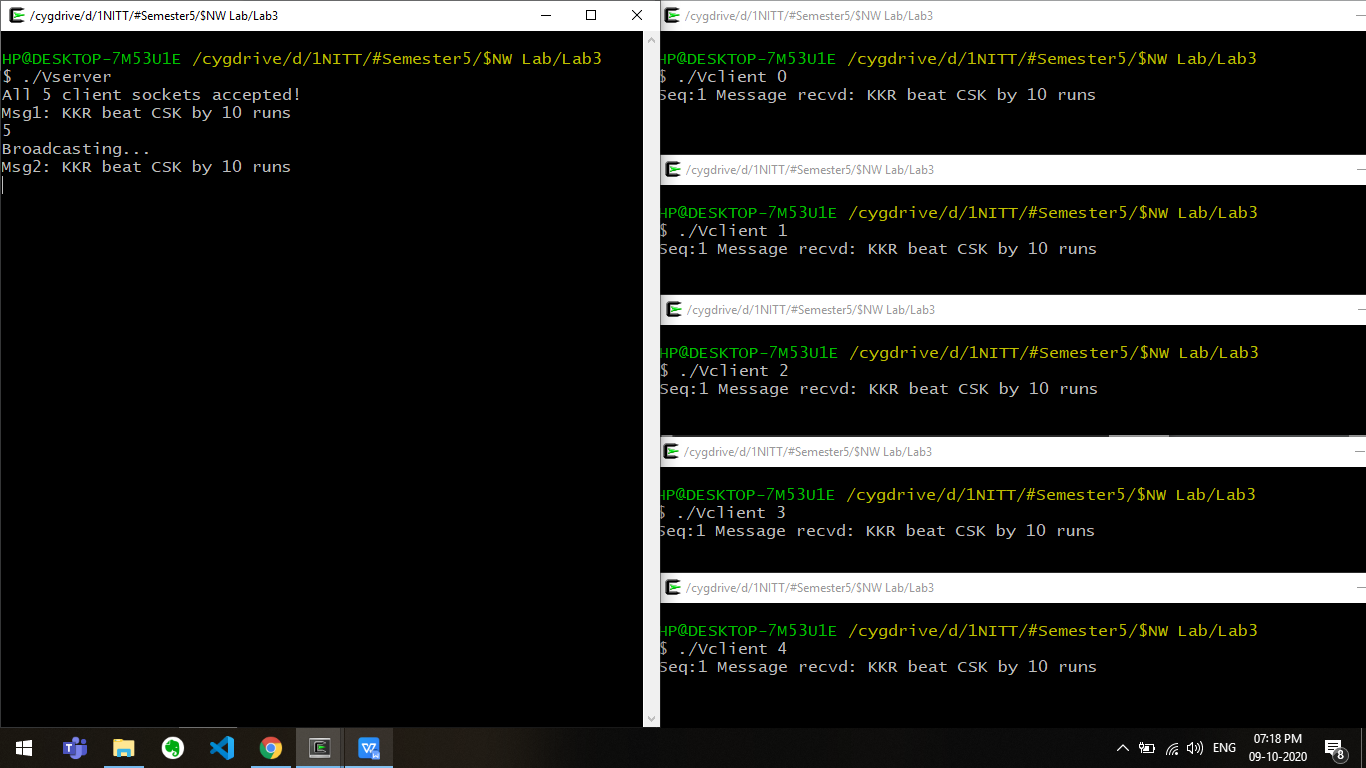
    }

    close(sock);

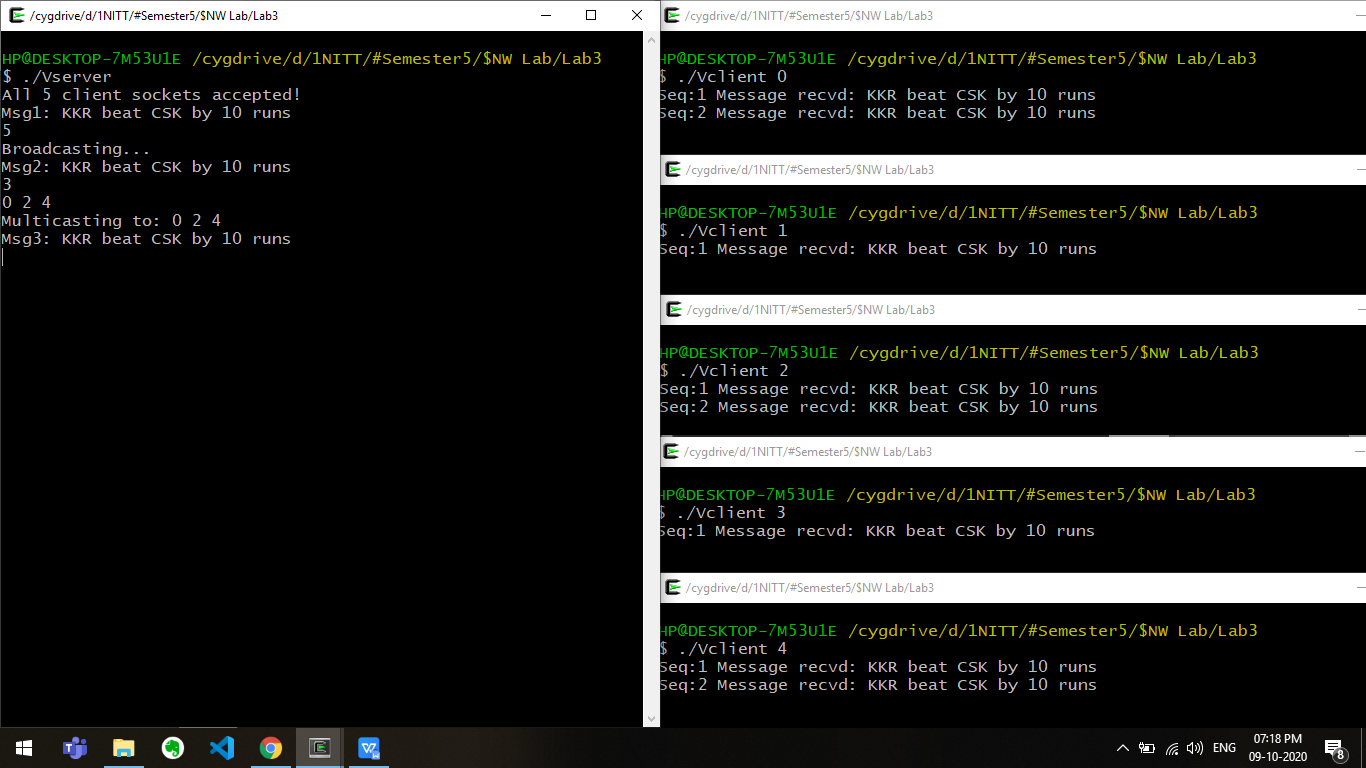
    return 0;

}

Broadcasting(All 5)



Multicasting(0,2,4)



Multicasting(3,4)

