Network Studio 실습 레포트#3

학과 : 소프트웨어학과

학번 : 2009122172

이름 : 신성철

**3차 실습 문제**

◆ UDP를 이용하여 아래 요구사항을 만족하는 간단한 채팅프로그램을 만드시오.

◆ 서버 프로그램 요구 사항

⦁ 서버에게 문자열을 전송한 클라이언트의 주소와 포트를 최대 4개까지 저장

⦁ 한 클라이언트가 문자열을 전송하면 나머지 클라이언트에게 문자열을 전송

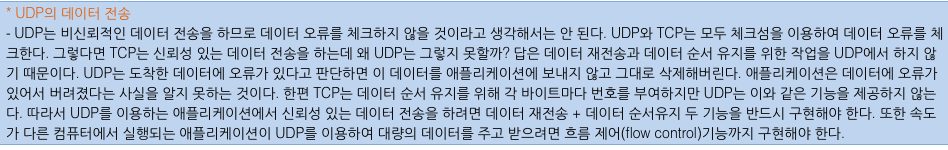
⦁ 한 클라이언트에게 문자열을 전송할 때 에러가 발생하면 해당 클라이언트가 채팅을

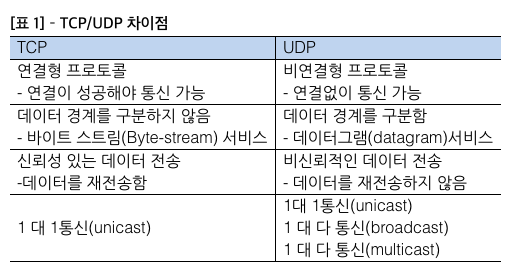
종료한 것으로 간주, 새 클라이언트가 채팅에 참여할 수 있음

◆ 클라이언트 프로그램 요구 사항

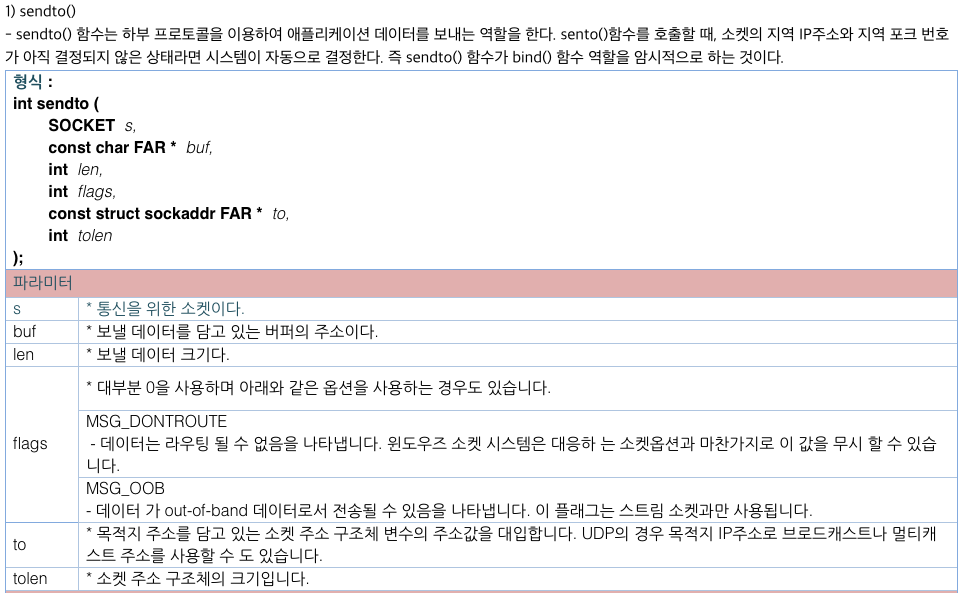
⦁ 사용자로부터 문자열을 입력받으면 서버에게 문자열을 전송 ⦁ 서버로부터 문자열을 받으면 화면에 출력

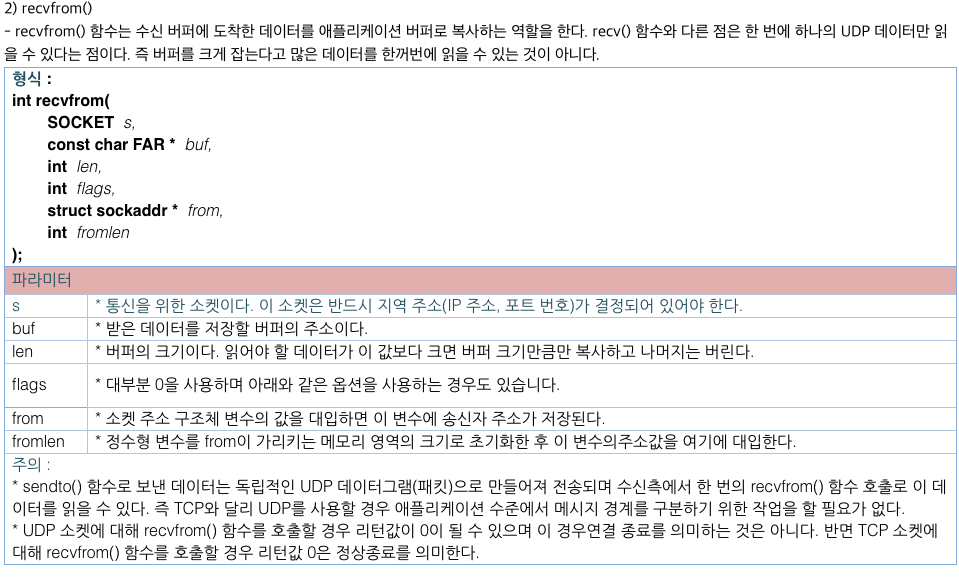
**접근방식**

****

****

함수설명

****

****

**소스코드**

//header.cpp

#ifndef Net03\_header\_h

#define Net03\_header\_h

# include <stdio.h>

# include <string.h>

# include <netinet/in.h>

# include <sys/socket.h>

# include <arpa/inet.h>

# include <unistd.h>

# include <stdlib.h>

# define PORT 20163

# define BUFFER\_SIZE 4096

#endif

//client.cpp

#include "header.h"

int main()

{

struct sockaddr\_in serverAddress;

struct sockaddr\_in fromAddress;

char smessage[BUFFER\_SIZE];

char rmessage[BUFFER\_SIZE];

int addr\_size;

int sentBytes;

int readBytes;

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

serverAddress.sin\_family = AF\_INET;

inet\_aton("127.0.0.1", (struct in\_addr\*) &serverAddress.sin\_addr.s\_addr);

serverAddress.sin\_port = htons(20162);

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

if(client\_socket == -1)

printf("socket error\n");

while(1)

{

memset(smessage, 0,BUFFER\_SIZE);

memset(rmessage,0,BUFFER\_SIZE);

printf("%s","please write a message(QT to QUIT) : ");

scanf("%s",smessage);

sendto(client\_socket, smessage, strlen(smessage), 0, (struct sockaddr\*)&serverAddress, sizeof(serverAddress));

addr\_size = sizeof(fromAddress);

if(smessage[0] == 'Q' && smessage[1] == 'T') break;

readBytes = recvfrom(client\_socket, rmessage, BUFFER\_SIZE,0, (struct sockaddr\*) &fromAddress, (socklen\_t\*)&addr\_size);

printf("%s", " A message from server : ");

printf("%s \n",rmessage);

}

close(client\_socket);

return 0;

}

//server.cpp

#include "header.h"

int main()

{

struct sockaddr\_in serverAddress, clientAddress;

socklen\_t clientAddressLength =0;

char smessage[BUFFER\_SIZE];

char rmessage[BUFFER\_SIZE];

int receivedBytes, addr\_size;

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

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

serverAddress.sin\_family = AF\_INET;

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

serverAddress.sin\_port = htons(PORT);

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

if(server\_socket ==-1)

{

printf("socket() error");

}

if(bind(server\_socket, (struct sockaddr\*) &serverAddress, sizeof(serverAddress)) == -1)

{

printf("bind error");

}

printf("server is running\n");

while(1)

{

memset(rmessage,0,BUFFER\_SIZE);

memset(smessage, 0, BUFFER\_SIZE);

clientAddressLength=sizeof(clientAddress);

receivedBytes=recvfrom(server\_socket,rmessage, BUFFER\_SIZE,0,(struct sockaddr\*) &clientAddress, &clientAddressLength);

printf("%s", "Message from client : ");

printf("%s \n",rmessage);

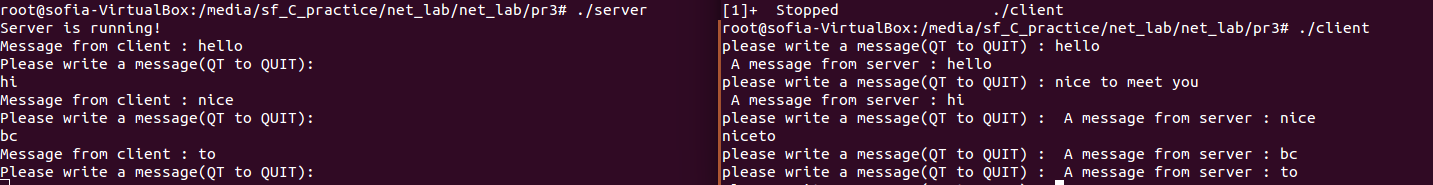
sendto(server\_socket, rmessage, strlen(rmessage),0,(struct sockaddr\*)&clientAddress, sizeof(clientAddress));

}

close(server\_socket);

return 0;

}

**테스트결과**

**느낀점**

UDP로 채팅프로그램을 구현하는 데 많은 어려움을 이번 과제를 하며 느꼈다. TCP는 큐라는 것을 통해 차례로 연결 시도하는 클라이언트들을 관리할 수 있고 또한 connectedFD (스켓 생성리 리턴되는 양수)를 이용하여 에코서버를 쉽게 구현할 수 있는데, UDP는 클라이언트에게 어떻게하면 TCP의 큐를 형성해서 4명의 클라이언트까지 구현 하는지 도무지 생각이 나지 않았다. UDP 소켓통신에 대해 명확한 이해가 필요하다고 느낀 과제였다.