OOP Report

Term Project Final Report

Socket Programming in Java and Unix C

과 목 명:	객체지향 프로그래밍
학 과:	컴퓨터학과
학 번:	2000160184
이 름:	이 진 무
제 출 일:	2021년 11월 25일 (목)
담당교수:	이 상 근 교수님

1. 서론

(1) 팀원:

컴퓨터학과 2000160184 이 진 무

(2) 개발환경:

- OS : Linux RedHat 8.0 kernel 2.4.20 (server computer)
  Windows 2000 Profesional (client computer)
- CPU : Pentium 3 800MHz
- RAM : 128 MB

- Java version "1.4.1_01"
- Java(TM) 2 Runtime Environment, Standard Edition
- Java HotSpot(TM) Client VM
- C compiler: GCC version 3.2

(3) 프로젝트의 개요와 동기

이번 프로젝트는 Unix C의 Socket과 Java의 Socket Class를 이용해서 Chatting program을 만드는 과정에서 두 언어 사이의 차이점을 실제적으로 느껴본다. 그리고 Client/Server 방식과 Peer-to-Peer 방식을 이용해서도 각각 chatting program을 구현한다.

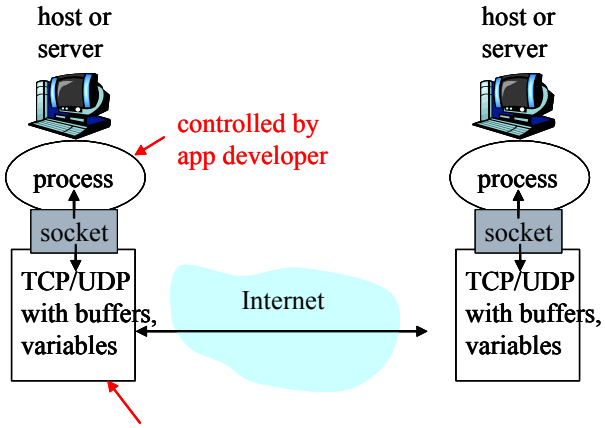
이 프로젝트를 선택하게 동기는 이 수업을 수강한 목적과도 부합한다. 컴퓨터학과에서 2년 정도의 시간동안 프로그래밍을 했으나, Java언어로는 특별히 공부를 해 본 경험이 없었다. 그래서 이번 학기 수업을 수강하게 되었다. 또 이번에 데이터 통신 과목을 같이 수강하게 되었는데, 여기서 Unix C 환경 상에서 Berkeley Socket Programming을 하게 되었다. 그런데, 흔히 Java는 Networking에 강한 언어라는 이야기를 많이 들었고, 두 가지 언어로 동시에 programming 하면서 Unix C에 비교되는 Java의 특징을 직접 느끼기 위해 이번 프로젝트를 진행하게 되었다.

2. 본론

이제 구체적으로 어떤 방식으로 프로젝트를 진행했는지에 대해서 알아보자. 먼저 Socket이 무엇인가에 대해 알아보고, 구체적으로 채팅 프로그램을 어떻게 구현했는지에 관해서 알아보자. 그리고 두 가지 언어로 구현한 결과, 얻어진 차이점에 대해서 알아보자.

(1) Socket의 정의와 동작 원리

이번 프로젝트에서는 C와 Java를 단순 비교하기 위해서 Java의 RMI(Remote Method Invocation)를 이용하지 않고 두 언어에서 공통적으로 가지고 있는 Socket을 이용하였다. Socket은 network programming을 위한 API(Application Program Interface)이다. 1980년대에 University of California at Berkeley에서 개발이 되어서 Unix 환경에서 쉽게 network program을 만들 수 있게 되었다.



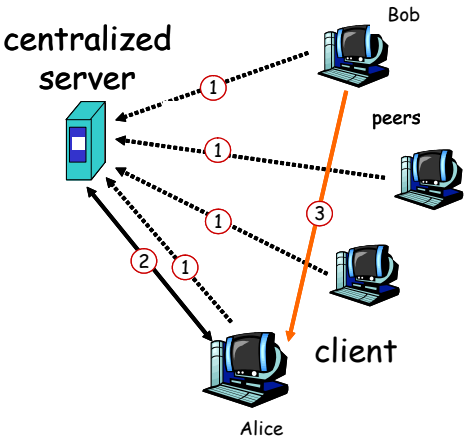
Socket은 위의 그림과 같이 TCP나 UDP 등의 transport layer와 application layer 사이에서 존재하는 networking interface이다. application process가 socket을 통해서 data를 전송하면 TCP나 UDP가 받아서 Internet을 통해서 상대방에게 data를 보내게 된다. 마찬가지로, 상대방에게서 TCP/IP나 UDP/IP를 통해서 data가 오면, socket이 받아서 application process로 넘겨주게 된다.

(2) Chatting program의 두 가지 Model

1) Client/Server(C/S) model

① 동작원리 설명

Client/Server model의 경우에는 Server는 중앙집권적으로 Client의 request를 받고 처리를 한 후에 다시 Client로 service를 하는 방식으로 이루어진다. 현재 모든 server들은 학회 서버에 동작하고 있다.



② Unix C로 구현한 방법

Unix C에서는 특별히 GUI를 구현하지 않았고, X 환경이 아니라 그냥 command line 환경에서 실행되도록 구현하였다.

ChatServer.c (server side)

- command line으로부터 port number를 받아온다.
- socket() system call로 socket을 연다.
- 적당히 server_addr structure를 setting한다.
- 그 server_addr와 socket을 bind()한다.
- client로부터의 connection을 wait한다.
- request를 받으면 무한 loop으로 들어가서 connection을 accpet()한다.
- recv() system call을 이용해서 message를 받고 접속된 모든 client들에게 send() system call을 이용해서 message를 전달한다.

ChatClient.c (client side)

- command line으로부터 server IP address, port number, user의 이름을 argument로 받아서 새로운 socket을 만든다.
- server_addr structure를 setting한다.
- server와 connection을 initiate한다.
- 무한 loop를 돌면서, recv() send() system call을 통해서 message를 주고받는다.

③ Java로 구현한 방법

Server 측은 command line interface에서 돌아가도록 구현하였고, client 측은 AWT를 이용해서 간단한 GUI를 구현하였다.

Class ChatServer (server side)

client로 부터의 connection을 받고, handler thread를 새로 만들어 돌리는일에 관련되 핵심 class이다. prompt로부터 받은 port 번호를 검사하고, ServerSocket instance를 생성하고 무한 loop에 들어간다. 그리고 accept() method를 통해서 client로부터의 connection을 받는다. 각 connection에 대해서 ChatHandler instance를 새로 만들어 각 socket을 parameter로 넘긴다. 그리고 메인 server는 계속 loop을 돌면서 새로운 connection을 기다리게 된다.

Class ChatHandler (server side)

이 class는 server로 들어온 client와의 각각 connection을 handle하는 역할을 수행한다. 연결된 client들에게 각각 message를 보내야 하기 때문에 Vector instance를 이용해서 모든 connection을 관리한다.

start() method
: input output stream을 연 후에, client 처리용 thread인 listener를 새로 만들어 시작시킨다. 도중 발생한 IOException은 무시한다.

stop() method
: listener thread에 interrupt를 걸고 dataOut stream을 닫는다.

broadcast() method
: client로부터 message를 받아서 다른 client들에게 broadcast한다. 우선 handler list에 synchronize를 거는데, 그 이유는 loop을 돌고 있는 동안에 다른 client가 끼어들거나 현재 연결된 client가 끊으면 안 되기 때문이다. 모든 작업은 synchronized block내에서 이루어진다. handler enumerationh class에서 하나의 element를 얻어와서 message를 보낸다.

Class ChatClient implements Runnable, WindowListener, ActionListener

이 client는 command line으로부터 server IP와 port number를 받는다. 그 정보를 이용하여 socket을 만들어서 연 후에, message들을 입출력할 Window를 생성한다. 사용자가 입력하는 문자열을 받아서 socket으로 내보낸다. 텍스트 전송을 위해서 readUTF() writeUTF() method를 사용하였다. character stream은 연속적인 stream을 제공하는 데에 비해, readUTF()와 writeUTF() method는 String instance를 포장해서 전송할 수 있다.

start() method
: server에 대해 socket을 연결하고, buffer stream인 dataIn과 dataOut을 가져온다. listener thread를 가동하여 서버로부터의 message를 받게 되고, chatting client의 frame window를 연다.

stop() method
: listener thread의 실행 중에 interrupt를 걸고 chatting frame을 닫고 network connection을 닫는다.

run() method
: listener thread는 run() method 안으로 진입하여 loop을 돌면서 IO stream으로부터 String instance를 읽어낸다. loop은 interrupt가 걸리거나 Exception이 발생하면 끝난다.

handleIOException() method
: listener thread가 null이면 chatting client를 멈춘다.

windowOpen() method
: frame window가 열리면 자동적으로 입력영역에 focus가 setting된다.

windowClosing() method
: 사용자가 window를 닫으려고 하면, stop() method를 호출한다.

actionPerformed() method
: 입력 영역에서 사용자가 enter를 눌렀을 때의 event가 전달된다. output stream에 사용자의 message를 기록하고 flush() method를 호출하여 message를 전달한다.

Class ChatApplet extends Applet implements Runnable

web browser 상에서 chatting을 할 수 있도록 한 applet이다. thread를 처리하기 위해서 Runnable interface를 구현한다.

init() method
: 입출력 창을 만든다.

start() method
: 새로운 listener thread를 생성하고 start()를 호출한다.

stop() method
: listener thread에서 stop()을 호출하고, null로 바꾼다.

run() method
: connection을 처리할 Socket instance를 생성한다. 입출력의 전달을 위해 DataInputStream, DataOutputStream instance도 생성한다.

execute() method
: 무한 loop를 돌면서 입력 stream에서 message를 읽어온다.

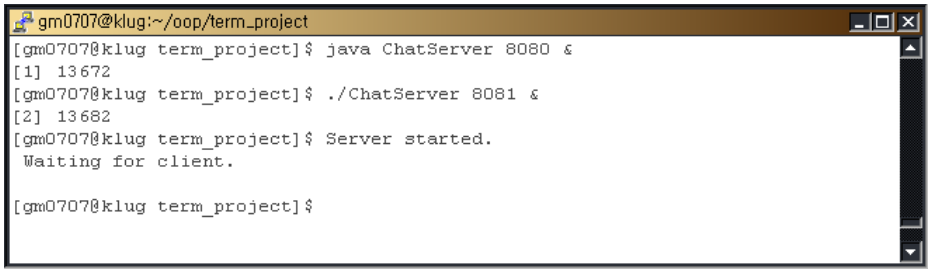
handleEvent() method
: 입력창에서 enter를 받으면, string을 받아서 writeUTF() method를 통해 outputstream으로 보낸다.

④ 실행 결과 화면

a. 먼저, server computer에서 C와 Java로 구현한 ChatServer를 background로 실행시킨다.

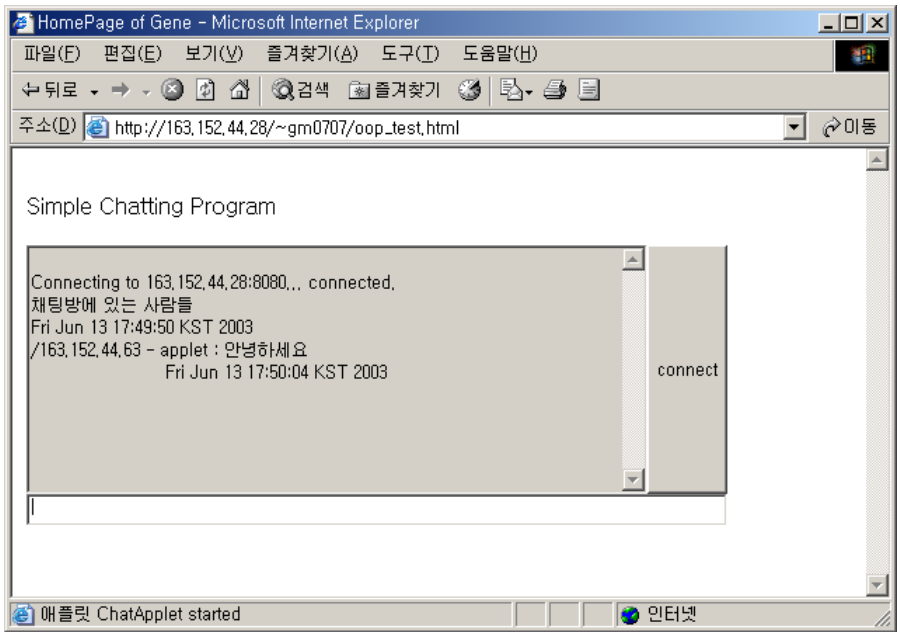
: ChatServer.c => ChatServer (Execute file)

: ChatServer.java, ChatHandler.java => ChatServer.class, ChatHandler.class



b. http://163.152.44.28/~gm0707/oop_test.html
에 있는 Applet의 동작을 확인한다.

: ChatApplet.java => ChatApplet.class

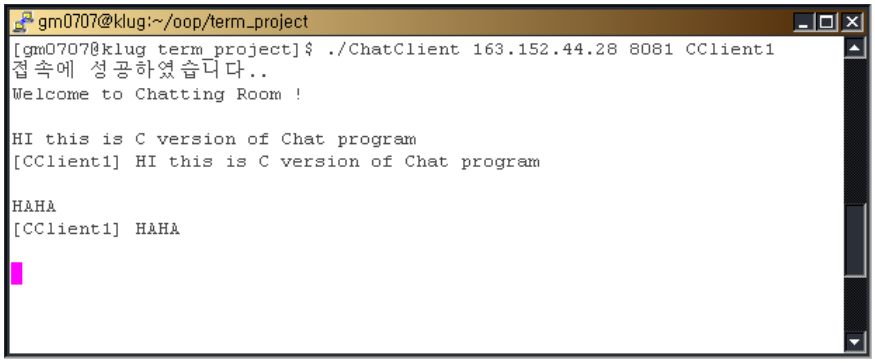


c. Java Application을 실행하여 동작을 확인한다.

=> ChatClient.java => ChatClient.class

d. C application을 실행하여 동작을 확인한다.

: ChatClient.c => ChatClient (execute file)

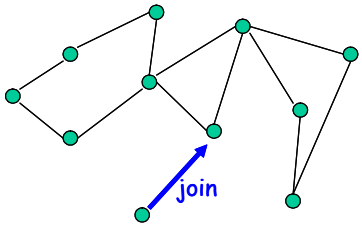


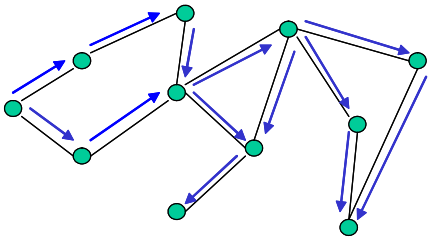
2) Peer-to-Peer(P2P) model

① 동작원리 설명

P2P 방식을 구현하기 위해서 Multicast나 Broadcast를 사용해야한다. 이번 구현에서는 IGMP (Internet Group Management Protocol)을 이용하여서 Multicast하였다. C/S 방식과 달리 중앙서버는 없으며, 그대신 모든 client들이 서로 동등한 입장에서 데이터를 주고받을 수 있다. C/S에서 통신을 수행하는 그룹 안에 속한 client들의 상태를 일관성 있게 유지할 중앙 집중식 application이 여기서는 필요가 없다.

Muticast에 기반을 두기 때문에 UDP protocol로 작동되기 때문에, unreliable data transfer라는 점이 특징적이다.





② Java로 구현한 방법

Class ChatPeer implements Runnable, WindowListener, ActionListener

GUI 처리와 threading을 위해서 위와 같은 interface들을 구현한다. ChatPeer class는 입출력을 할 수 있는 Text 창을 가진 chatting window를 열고, thread를 하나 돌려서 packet이 들어오기는 대기하여 처리한다.

initAWT() method
: chatting window를 만들고 적절히 component들을 배치한다. Frame을 만들고 TextArea type의 output과 TextField type의 input을 만들고 action listener를 등록한다.

start() method
: chatting을 시작하게 한다. network connection을 초기화하고, chatting frame을 띄운다. thread instance인 listener를 만들어 돌린다. 이 thread는 packet을 받아 처리하게 된다.

initNet() method
: 네트워크 연결을 초기화한다. port를 설정해주고 MulticastSocket type의 socket을 생성한다. TTL을 1로 setting함으로써 local network 안에서만 packet이 돌도록 한다. 멀티

캐스트 그룹에 입출력 DatagramPacket instance를 생성한다.

stop() method
: chatting system을 멈춘다. frame window를 닫고 listener thread에 interrupt를 걸고 socket을 닫는다. 물론 group에서도 leave한다.

windowOpened() method
: input text field에 입력 focus를 준다.

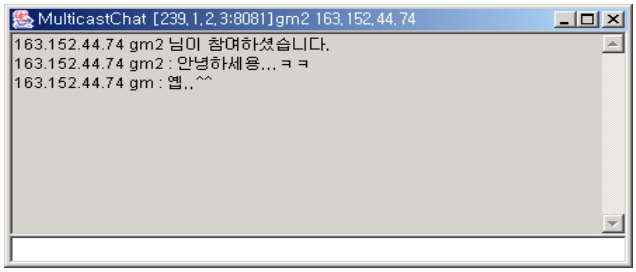
windowClosing() method
: 사용자가 frame을 닫으려고 하면 stop()을 호출한다.

actionPerformed() method
: WindowListener interface에 선언된 것이다. 사용자가 text field에 message를 쓰고 enter를 누르면 발생하는 event를 받는다. 받은 data를 UTF형식으로 만들고 outgoing Socket으로 보낸다.

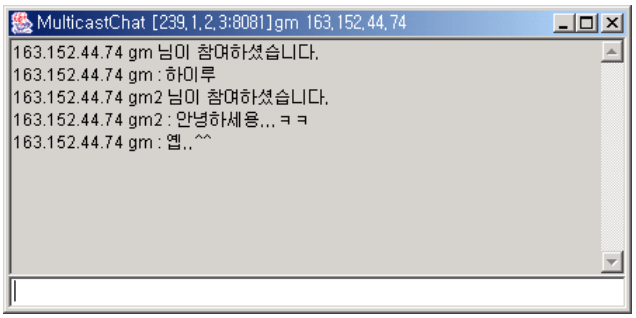
handleIOException() method
: IO 예외 처리용 method이다. listener thread가 null이면 chatting system을 멈춘다.

run() method
: 무한 loop을 돌면서 message를 받아 출력영역에 표시한다. incoming socket으로부터 receive()하여 packet을 받고, message를 뽑아낸다.

③ 실행 결과



ChatPeer.class



(3) Socket programming in Unix C and Java 비교

① Speed

Network program을 구현할 때 중요한 부분 중의 하나가 실행 속도이다. 채팅 프로그램에서 중요한 부분은 사용자가 메시지를 보냈을 때에 빠른 속도로 상대방에게 전달되는 것이다. 얼핏 생각했을 때에는 Java가 C보다 많이 느릴 것이라고 생각할 수 있고, 실제로도 약간 느린 감이 없지 않아 있었다.

compile시에는 C compiler보다 Java compiler가 많이 느린 것을 알 수 있다. 그리고 실행 시에도 Java의 경우에는 compile한 byte code를 Java Virtual Machine이 interpret해야 하므로 compile후 바로 link하면 run이 되는 C code에 비해 느릴 수 있다. 하지만, 일단 실행이 되고 나서는 사용자가 메시지를 보내고 상대방에게 도착하는 시간은 거의 차이 나지 않는 것을 발견할 수 있었다.

② Programming Convenience

이번 프로젝트를 진행하면서 느낀 가장 큰 Java의 장점은 프로그램을 편하게 할 수 있다는 것이었다. Unix C상에서 Socket programming을 하면, 정말 detail에 대한 것에 많은 신경을 써야 한다. pointer 개념을 완벽히 알아야하고 여러 가지 header를 include해야만 한다. 하지만 Java의 경우에는, java.net package에서 제공하는 많은 class들을 import해서 사용하면 간단히 socket programming을 할 수 있다.

예를 들어, server가 하나의 socket을 만들어서 client의 request를 하나 받는 과정을 수행하기 위한 부분을 비교해 보자.

C version

```
server_fd = socket(AF_INET, SOCK_STREAM, 0);
memset((char *)&server_addr, '\0',  sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
server_addr.sin_port = htons(atoi(argv[1]));
bind(server_fd, (struct sockaddr *)&server_addr, sizeof(server_addr));
clilen = sizeof(client_addr);
//accept a connection on the socket
client_fd = accept(server_fd, (struct sockaddr *)&client_addr, &clilen);
```

Java version

```
ServerSocket server = new ServerSocket (port);
```

Socket client = server.accept ();

말이 더 이상 필요 없을 것이다. Java에서 얼마나 쉽게 network programming할 수 있는지 쉽게 알 수 있다.

③ Compile Once and Run Everywhere

Unix C로 구현한 code 같은 경우에는 Unix나 Linux 등의 운영체제에서만 compile이 가능하고, run도 가능하다. 하지만 Java의 경우는, code을 만들어서 한 번 class byte code를 만들면 Java Virtual Machine이 있는 임의의 Operating Systems에서 실행이 가능하다. 이런 점은 Java의 강점 중의 하나이라고 할 수 있다.

④ Easy to get program through Applet

어떤 program을 구현한 후에, 중요한 요소 중에 하나가 그것을 얼마나 쉽게 소비자들에게 전달할 수 있냐이다. Chatting program 같이 가벼운 program은 굳이 file을 받아서 소비자가 compile하는 것은 매우 불편한 일일 것이다. Java Applet을 이용하면, 사용자가 web page에 접속만 하면 web browser를 통해서 쉽게 서비스를 이용할 수 있다. 실제로 이번에 구현한 chatting server와 client를 내 홈페이지가 있는 Server computer에 올려놓았다. 그래서 내 홈페이지를 방문하는 사람은 쉽게 프로그램을 사용할 수 있도록 하였다.

http://163.152.44.28/~gm0707/oop_test.html

3. 결론

(1) 의의

같은 기능을 수행하는 프로그램을 두 가지 언어로 구현하는 기회가 이번이 처음이다. 주로 어떤 문제가 주어졌을 때에, 어떻게 해서든지 그것을 푸는 것에 중점을 많이 두는 경향이 짙었다. 하지만, 같은 기능의 프로그램을 두 가지 관점으로 바라보면서 구현했다는 점에서 좋은 경험이 되었다. 그리고 막연히 어렵게만 느껴졌던 Java가 친숙해 질 수 있는 좋은 기회였다. 기존에 procedural 언어만 다루어서 OO 개념이 딱히 머리에 들어오지 않았었는데, 이번에 개념을 정확히 파악할 수 있었다.

(2) 장단점 및 추가적으로 해결해야할 점들

- C version과 Java version의 연동

Java로 구현한 Server와 Client, 그리고 Applet은 정상적으로 작동되고, C로 구현한 Server와 Client는 서로 잘 동작한다. 하지만, Server를 C로 Client를 Java로 해서 수행시키면, 접속은 정상적으로 이루어지지만, 서로 String을 주고받는 것이 제대로 되지 않는다. 이런 type conversion 문제가 꼭 해결해야하는 과제이다.

- User Interface

이번 project의 관점은 C와 Java의 차이점을 파악하는데 있었다. 자연스럽게 User Interface에 대한 신경은 덜 쓰게 되었다. 그래서 프로그램의 마무리 완성도가 다소 부족한 느낌이 든다.

4. 후기

(1) oop 수업과 term project 수행하면서 느낀 점

앞에서 언급했듯이, 이번 수업은 Java 언어를 접하고자 하는 목적으로 수강하게 되었고, 그 목적을 거의 달성한 것 같다. 2001학년도에 Programming Language 수업을 수강한 적이 있다. 그 때에도, 객체지향이라는 것이 무엇인지 배웠고 또한 시험도 잘 봤던 것으로 기억난다. 하지만, 실제적으로 그 객체지향의 개념을 접목시키는 과정의 부재로 실전적인 감각을 배우지는 못했다. 이번 수업에서도 수업시간 동안에만 배우고 실제 programming을 하지 않았다면, 이론과 실재의 괴리가 여전했을 것이다. 하지만, 이번에는 term project의 수행과 많은 Exercises를 해결해보고, program assignment들을 하면서 실제적인 개념을 터득할 수 있는 기회가 되었다.