

HW1 Report

2013-11421 이은서

1.Problem Statement

과제1은 간단한 채팅 프로그램을 만드는 것이다.

Specification

Server

- 서버는 여러 클라이언트에게 서비스를 제공해야 한다.
- 클라이언트가 deactivated인 경우 서버는 메시지를 저장했다가 클라이언트가 activate 되었을 때 그 동안 받은 메시지를 전송해주어야 한다.

Client

- 클라이언트는 할당된 id로 로그인 할 수 있다.
- 메시지를 보낼 수 있다. 이 때 메시지 길이는 140자로 제한하기로 한다.
- 메시지를 받을 수 있다. 이 때 한 번 읽은 메시지는 다시 볼 수 없는 것으로 한다.

계획

Server

- 여러 클라이언트의 요청을 처리해야 하기 때문에 select()를 이용한 multiplexing 서버를 만들어 서비스하도록 한다.
- 한 id는 한 곳에서만 activate되어야 한다. 이미 activate된 아이디로 로그인을 시도할 경우, 로그인에 실패했다는 메시지를 보낸다.
- 클라이언트의 상태(activate, deactivate)에 대해 알아야 한다.
- 스펙 문서에 클라이언트의 IP와 포트 번호를 알고 있어야 한다는 명시가 있다.
- 클라이언트가 deactivated일 때 메시지를 저장하기 위해 메시지 큐를 구현해야 한다. 구현의 편의를 위해 각 클라이언트별로 로그 파일을 만들어 관리하기로 했다.

Client

- 로그인 시도 할 때 올바른 아이디 형태인지 체크한다.
- 메시지를 140자까지 보낼 수 있다.
- Activate된 상태일 때 자신에게 오는 메시지는 항상 받되, 읽기 명령이 있을 때만 지난번에 읽은 마지막 메시지 다음부터 읽기 명령 전까지 받은 메시지를 표시하도록 한다.
- 이를 위해서 스레드를 나누어 한 스레드에서는 자신에게 오는 메시지를 계속 받고, 다른 스레드에서는 메시지 보내기 등의 명령을 처리하는 것으로 한다.

2.Implementation Details

Protocol Design

아래와 같이 프로토콜을 디자인했다.

- 프로토콜 사이즈는 공통적으로 144byte이다.

Client to Server

1. action code
 - activate/logon : 0
 - message send : 1
2. log on protocol
 - [action code | id]
3. message send
 - [action code | destination | source | message content]
 - message content는 140자까지 허용한다.

Server to Client

1. action code
 - activate : 0
 - message deliver : 1
2. log on 반응
 - log on 성공 : [action code | 0]
 - log on 실패 : [action code | 1]
3. message deliver
 - [action code | source | message content]

Client Manual

- log on

- i. 로그인 하려면 1~9 사이의 한자리 숫자를 id로 입력하면 됩니다.
 - ii. 정상적으로 로그인 되면 login success 가 콘솔에 나타납니다.
- 메시지 보내기
 - i. 콘솔에 s 를 입력합니다.
 - ii. 콘솔에 To : 이 보이면 받을 사람의 id(0~9 사이의 한자리 숫자)를 입력합니다.
 - iii. 콘솔에 Content : 이 보이면 보내고 싶은 메시지를 입력합니다.
- 메시지 읽기
 - i. 콘솔에 r 를 입력합니다.
 - ii. 콘솔에 지금까지 받은 메시지들이 나타납니다.
- deactivate/log off
 - i. 콘솔에 q 를 입력합니다.

3.Results

로그인

로그인 성공

- 로그인이 성공한 경우, 서버는 클라이언트에게 deactivate된 동안 온 메시지를 보낸다.

```

~/Documents/snucse/ComputerNetwork/hw1 > asdf • ./client
Enter your ID : 3
login success

[ s : send message
  r : read message
  q : exit ]

~/Documents/snucse/ComputerNetwork/hw1 > asdf • ./server
client connected : fd4
get msg from fd4
fd4 가 id(3)로 로그인 함
저장된 메시지를 fd4에게 모두 전달 함

```

로그인 실패

- 로그인이 실패한 경우, 클라이언트가 종료된다.

```

~/Documents/snucse/ComputerNetwork/hw1 > asdf • ./client
Enter your ID : 3
login failed

```

```
~/Documents/snucse/ComputerNetwork/hw1 > asdf • > ./server
client connected : fd4
get msg from fd4
fd4 가 id(3)로 로그인 함
저장된 메시지를 fd4에게 모두 전달 함
client connected : fd5
get msg from fd5
Invalid login : client disconnected : fd5
█
```

메시지 보내기

Client

- deactivate상태인 5와 activate상태인 1에게 메시지를 보냈다.

```
s
=====Send Message=====
To : 5
Content :
sending message. hello, world! World of Warcraft!
=====
[ s : send message |
  r : read message |
  q : exit          |
=====
s
=====Send Message=====
To : 1
Content :
hello, world!
=====
[ s : send message |
  r : read message |
  q : exit          |
=====
█
```

Server

- deactivate상태인 5에게 보내는 메시지는 저장하고 activate상태인 1에게 메시지를 바로 전달한다.

```

get msg from fd4
메시지가 왔어요! fd4가 보냄
store message.
client connected : fd5
get msg from fd5
fd5 가 id(1)로 로그인 함
저장된 메시지를 fd5에게 모두 전달 함
get msg from fd4
메시지가 왔어요! fd4가 보냄
메시지 바로 전달 함 : 13hello, world!
□

```

메시지 읽기

5에서 메시지 읽기

- 5로 로그인 후 메시지 읽기를 실행했다.

```

r
=====Read Message=====
From : 3
Content :
sending message. hello, world! World of Warcraft!
=====
| s : send message |
| r : read message |
| q : exit         |
|-----|

```

1에서 메시지 읽기

- 1에서 메시지 읽기를 실행했다.

```

r
=====Read Message=====
From : 3
Content :
hello, world!
=====
| s : send message |
| r : read message |
| q : exit         |
|-----|

```

클라이언트 종료

Client

```
| s : send message |  
| r : read message |  
| q : exit         |  
└──────────────────┘  
q  
~/Documents/snucse/ComputerNetwork/hw1 > asdf
```

Server

```
get msg from fd6  
client disconnected : fd6  
█
```

4.Discusion

1. 입출력 처리의 어려움 C로 구현했기 때문에 input, output을 처리하는 과정이 번거로웠다.
2. 프로토콜 디자인
 - 구현을 쉽게 하기 위해서 모든 프로토콜을 일정한 길이로 정했는데, login같은 경우 실제로 정보를 담고 있는 부분이 매우 적어서 비효율적이다. 하지만 클라이언트 당 한 번만 하기 때문에 그냥 모든 프로토콜을 일정하게 하기로 했다.
 - 메시지 보내기/받기의 경우는 메시지 내용, 보내는 사람, 받는 사람만 있는데 시간을 함께 보내도 좋을 것 같다. 현재 구현에서는 클라이언트의 소켓들을 차례로 돌며 요청 이 있는 것을 처리하도록 되어 있는데 이 때 시간 순서가 뒤바뀌어 처리될 수 있는 가능성이 있다. 예를 들어 c가 1초에 메시지를 b에게 보내고, a가 2초에 메시지를 b에게 보냈는데 서버에서 a의 메시지를 먼저 처리하는 경우 b는 c보다 a의 메시지를 먼저 보게 된다. 시간을 함께 보내면 시간순으로 정렬한 메시지를 받을 수 있을 것이다.