

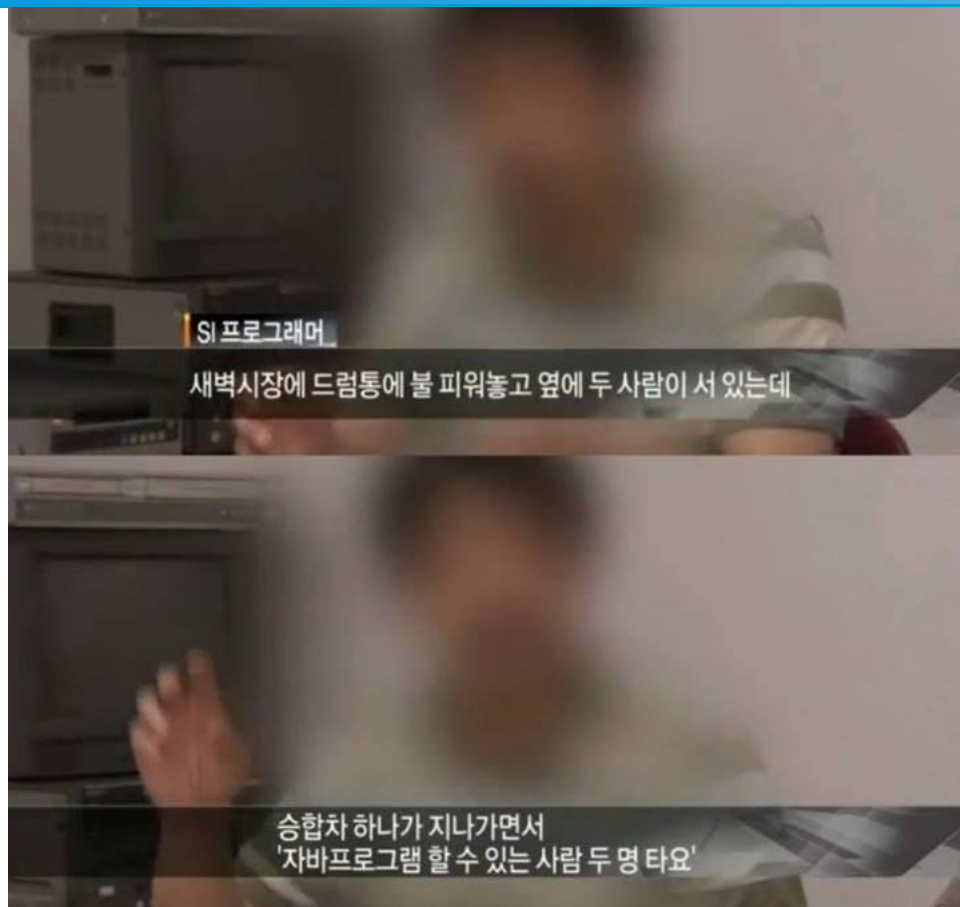
JAVACHAT

Simple Chatting Program

자바타요

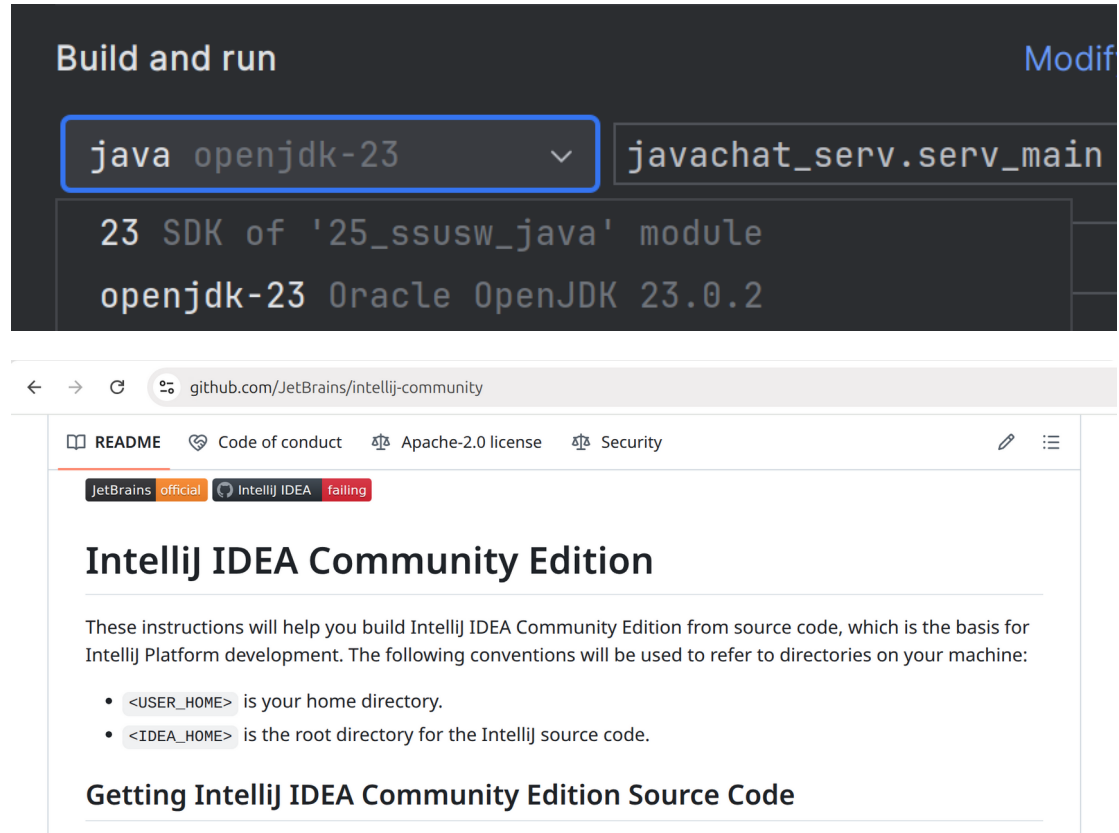
이건희, 황수민, 송민수, 강태우, 장동현, 김진기

자바타요?



Development Environment

- **Various Host OS**
(Windows, Linux, macOS)
... Because of WORA,
different OS does not matter!
- **OpenJDK 23**
- **Jetbrains IntelliJ 2025.1.1**
Community edition
It's Free and Open Source!



What We Focused

- **Implement of Simple Protocol**

Protocols: Rules promised to send and receive data

- Syntax, Semantics, Timing

- **Socket Programming**

Socket : Endpoint of IPC(Inter-Process Communication)

Process : Unit of resource allocation at program execution

Berkeley Socket

- Developed for API of BSD 4.2, in 1983 at UC Berkeley
- Located between Layer 5 (Application) and Layer 4 (Transport) of TCP/IP 5 Layer
- Using the Internet to send and receive data
→ Considering the network topology, this is IPC!

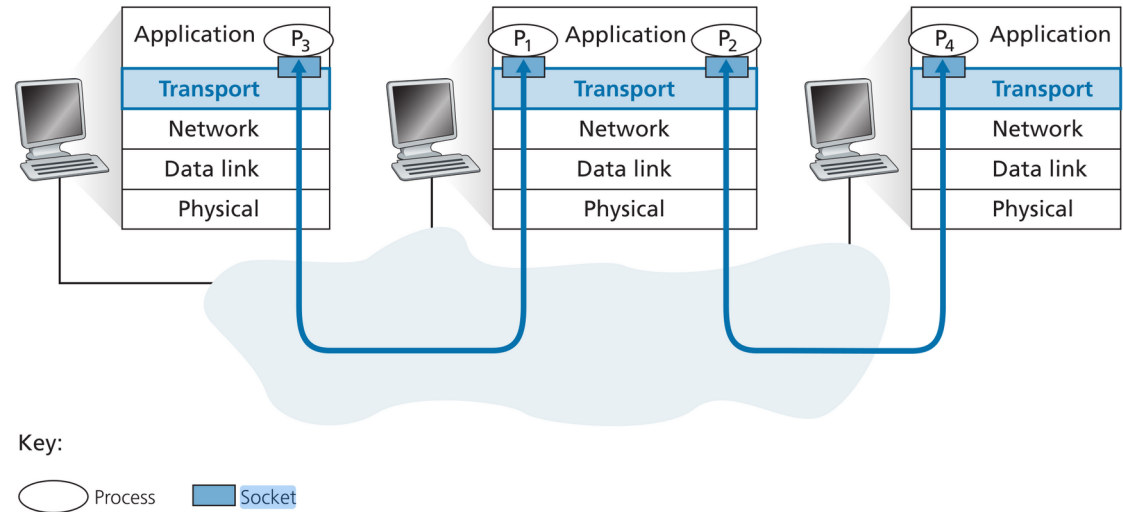


Figure 3.2 ♦ Transport-layer multiplexing and demultiplexing

Socket in Java

- A wrapper built into Java,
not a Berkeley Socket
- Runs the same on all host JVMs
→ Actual implementation
uses system calls, which
provided by each host OS
UNIX - <sys/socket.h>
Windows - <Winsock2.h>
- Java.net.Socket
Java.net.ServerSocket
→ **TCP Communication**



The screenshot shows a terminal window with the title "kh@ThinkPad-T16: /usr/include/sys". It displays the contents of the `socket.h` header file, specifically lines 99 through 119. The code is color-coded: comments are in blue, keywords like `extern` and `int` are in green, and function names and variables are in white. The functions shown are `socket`, `socketpair`, `bind`, `getsockname`, and `connect`. At the bottom of the terminal, a status bar shows "socket.h [R0]" on the left, "117, 1-4" in the center, and "29%" on the right.

```
99 /* Create a new socket of type TYPE in domain DOMAIN, using
100    protocol PROTOCOL.  If PROTOCOL is zero, one is chosen automatically.
101    Returns a file descriptor for the new socket, or -1 for errors.  */
102 extern int socket (int __domain, int __type, int __protocol) __THROW;
103
104 /* Create two new sockets, of type TYPE in domain DOMAIN and using
105    protocol PROTOCOL, which are connected to each other, and put file
106    descriptors for them in FDS[0] and FDS[1].  If PROTOCOL is zero,
107    one will be chosen automatically.  Returns 0 on success, -1 for errors.
108    */
109 extern int socketpair (int __domain, int __type, int __protocol,
110                       int __fds[2]) __THROW;
111
112 /* Give the socket FD the local address ADDR (which is LEN bytes long).  */
113 extern int bind (int __fd, __CONST_SOCKADDR_ARG __addr, socklen_t __len)
114                __THROW;
115
116 /* Put the local address of FD into *ADDR and its length in *LEN.  */
117 extern int getsockname (int __fd, __SOCKADDR_ARG __addr,
118                        socklen_t *__restrict __len) __THROW;
119
120 /* Open a connection on socket FD to peer at ADDR (which LEN bytes long).
```

TCP

- Protocol used in Layer 4 of TCP/IP
- Connection-Oriented

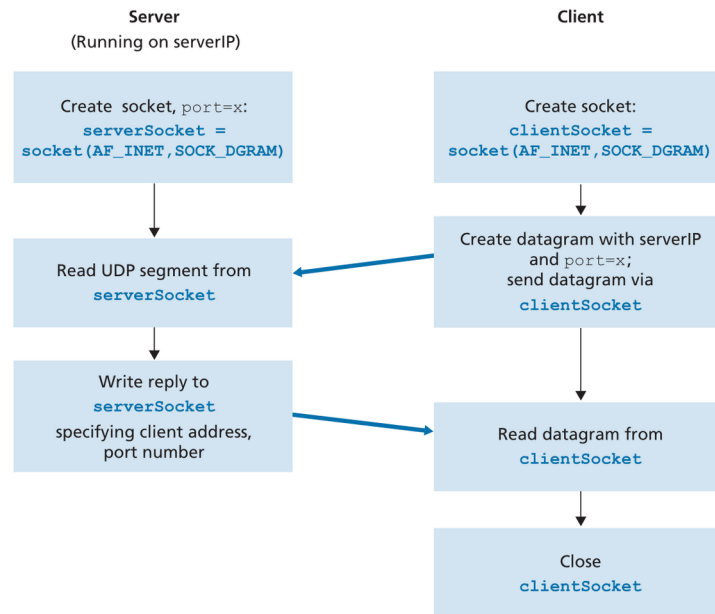


Figure 2.27 ♦ The client-server application using UDP

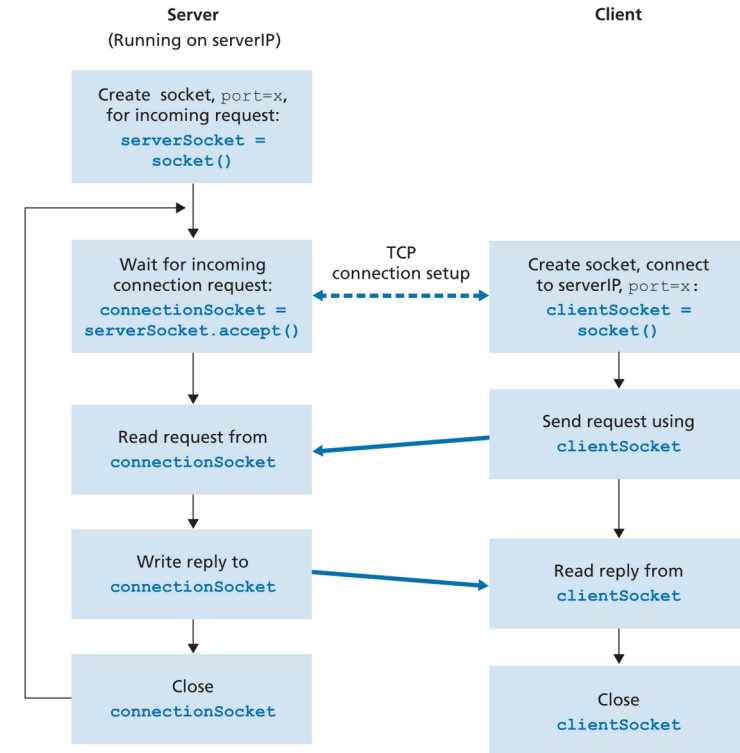


Figure 2.29 ♦ The client-server application using TCP

Basic Syntax

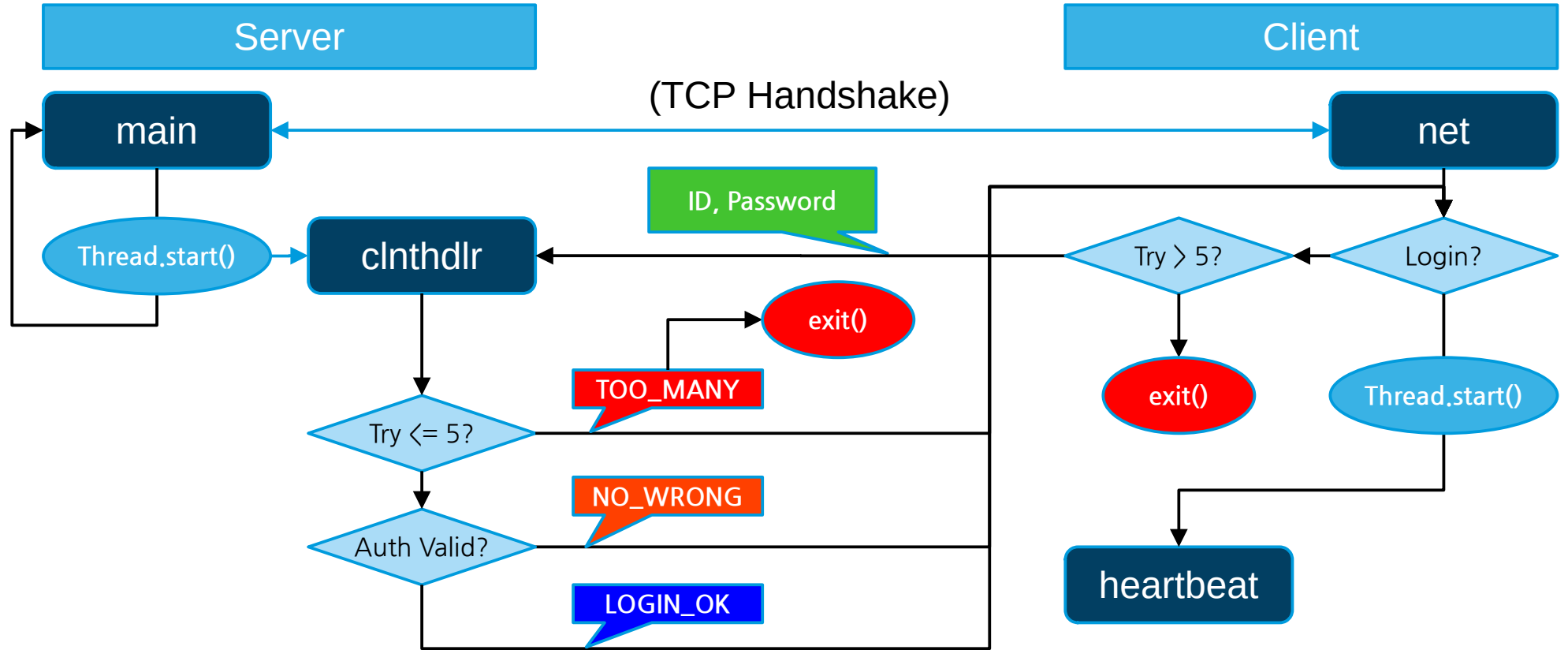
- **Every line ends with ‘\n’**

BufferedReader's `readline()` method reads everything before ‘\n’

- **< Exactly Once >, { None or More }, [None or Once]**

Backus-Naur Form

(1) Login - Timing



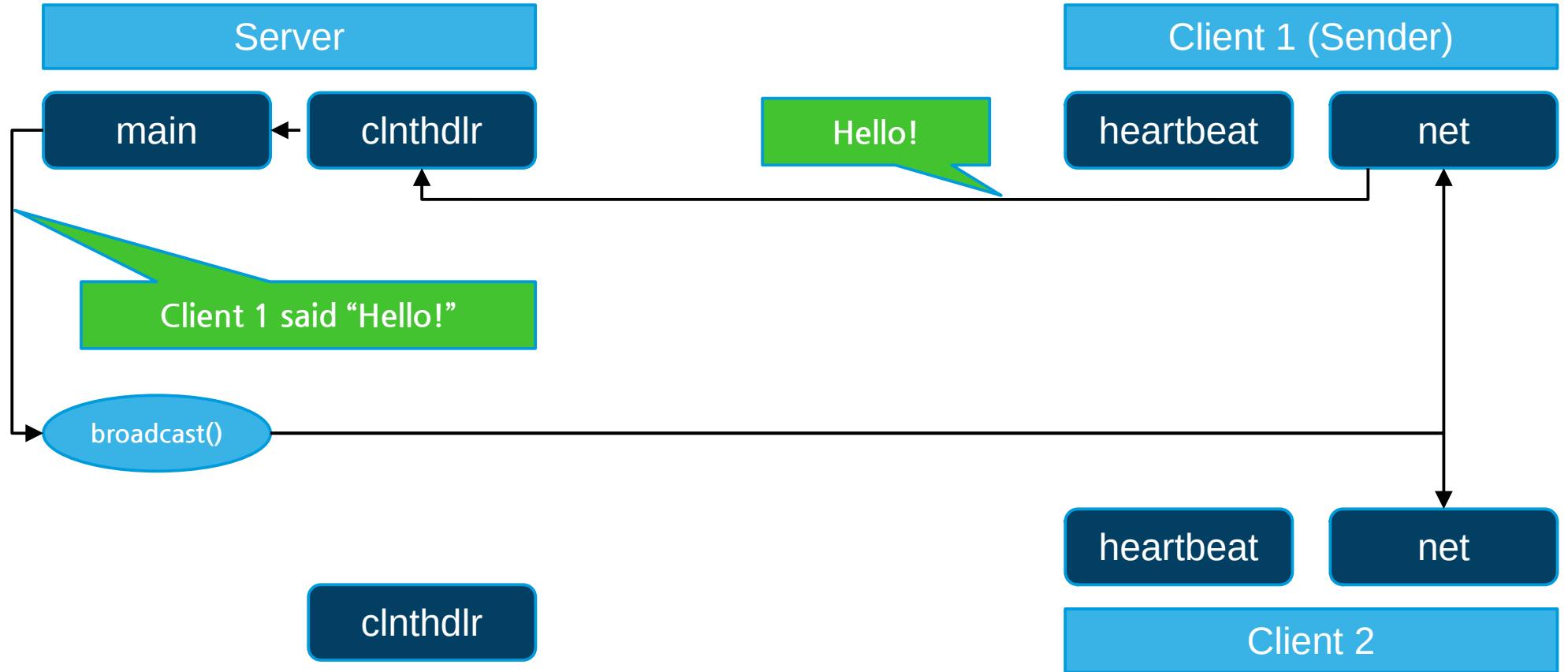
(1) Login - Syntax & Semantics

- **Client.net** → **Server.clnthdlr**
 <String> <String> : id, password
- **Server.clnthdlr** → **Client.net**
 <Int> : response



100 - LOGIN_OK
200 - NO_WRONG
300 - TOO_MANY

(2) Plain Message - Timing



(2) Plain Message - Syntax & Semantics

- (Some) Client.net \rightarrow Server.clnthdlr
- Broadcast (Server.main \rightarrow (All) Client.net)

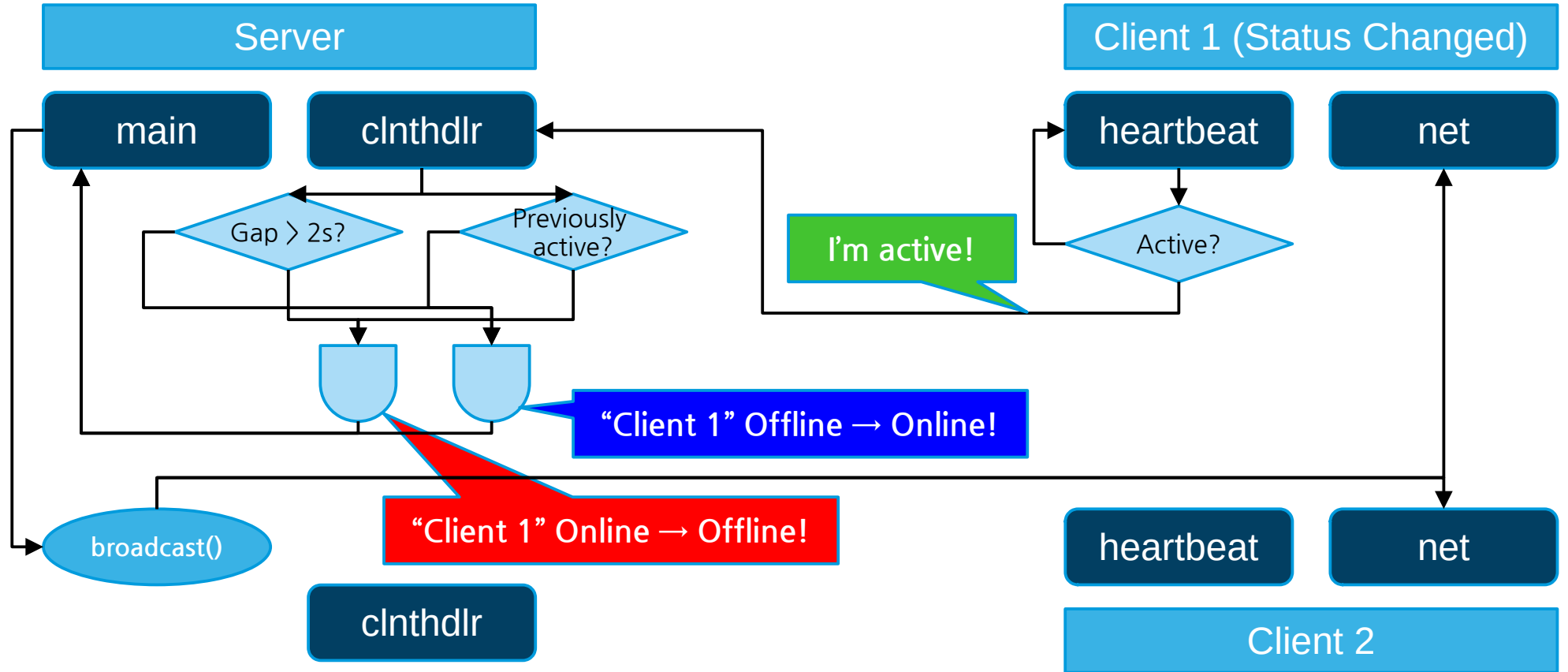
$\langle \text{Int} \rangle [\text{String}]$

: Type, Message

0 : Plain Message

What user want to send
(Empty Message Allowed)

(3) Heartbeat - Timing



(3) Heartbeat - Syntax & Semantics

- **(Some) Client.net → Server.clnthdlr**
 <Int> “true” : Type, No meaning
- **Broadcast (Server.main → (All) Client.net)**
 <Int> <String> <Boolean> : Type, Client Name, Active



1 : Heartbeat

Demonstration

Significance and Limitations

[Significance]

- No usage of external things other than Java
→ Valuable experience to make thought about fundamental of framework

[Limitations]

- All packets are plain text
- Implement without JAAS

JAAS : Java Authentication and Authorization Service

```
127.0.0.1      TCP      66 48016 → 12345 [ACK] Seq=1 Ack=67 Win=512 L...
127.0.0.1      TCP      119 12345 → 48016 [PSH, ACK] Seq=14 Ack=1 Win=...
127.0.0.1      TCP      66 48016 → 12345 [ACK] Seq=1 Ack=67 Win=512 L...

0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00 .....E.
0010 00 69 19 2c 40 00 40 06 23 61 7f 00 00 01 7f 00 .i.,@.@. #a.....
0020 00 01 30 39 bb 90 f5 8a 16 d2 56 96 79 ae 80 18 ..09.....V.y...
0030 02 00 fe 5d 00 00 01 01 08 0a a3 1e 41 84 a3 1e ...].....A...
0040 1a af 6b 68 6c 65 65 20 3e 3e 20 74 68 69 73 20 ..khlee >> this
0050 6d 6f 6e 74 68 27 73 20 6f 66 66 69 63 65 20 64 month's office d
0060 6f 6f 72 20 70 61 73 73 77 6f 72 64 20 69 73 20 oor pass word is
0070 31 32 33 34 35 36 0a 123456.
```

```
** khlee님이 채팅방에 들어오셨습니다 **
** twkang님이 채팅방에 들어오셨습니다 **
khlee >> this month's office door password is 123456
```


Any Questions?

Thank you!

[Reference]

Computer Networking: A Top Down Approach 8th, James F. Kurose, Pearson, 2021