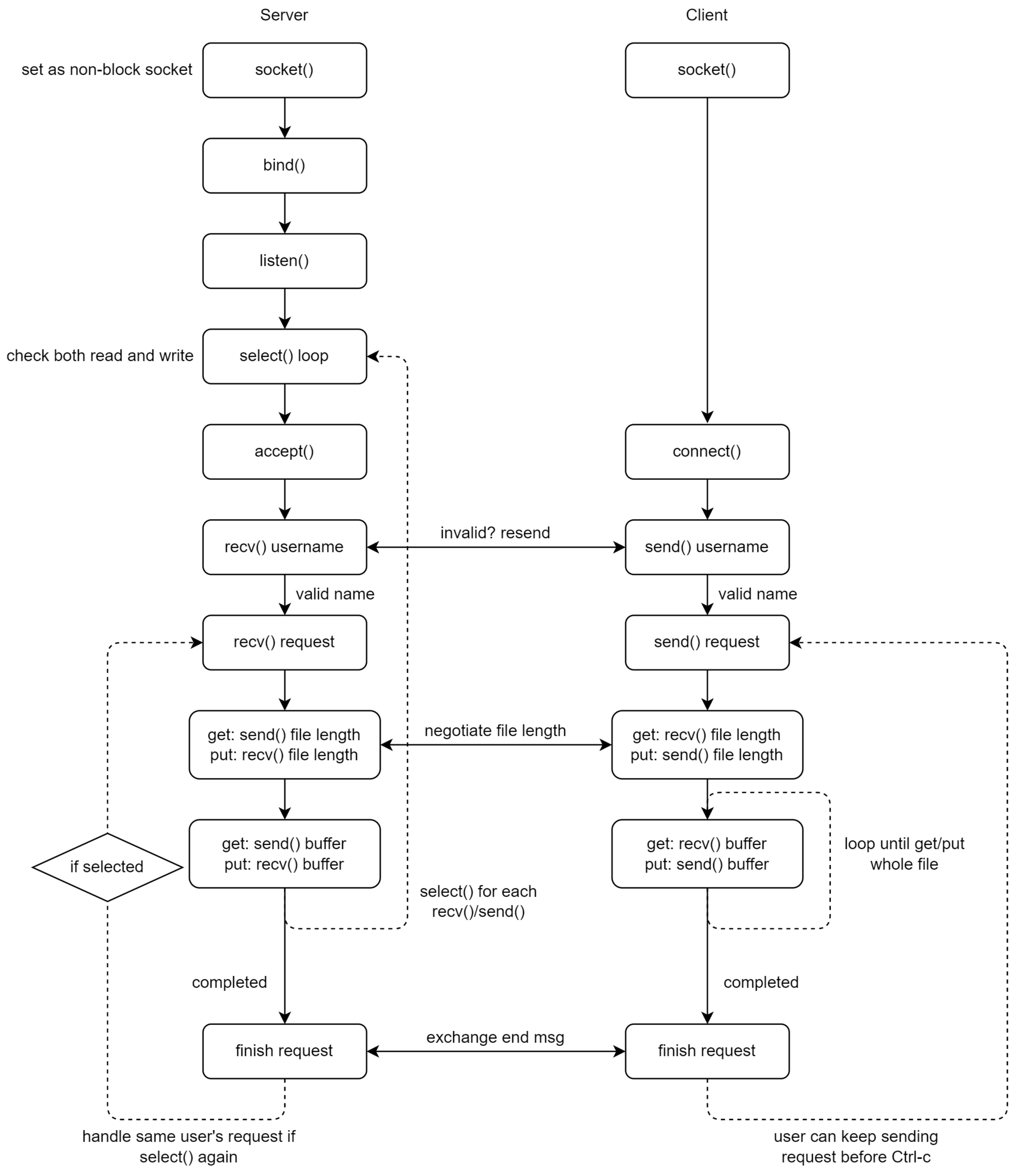
CN Project Phase 1 Report

B08902055 李英華

Flowchart



- For the server, every `send()` and `recv` need `select()` to gaurantee that the client is ready to read/write. But for simplicity, we didn't draw every of it in the above flow chart.

Explanation

- With `select()`, we can monitor multiple client socket fds to see which client is ready instead of blocking on single user, waiting it to become ready. This builds the basis of dealing with multiple users.
- Make all the message and file transfer send/receive in batch. That is, each time a buffer of data is transferred, it needs to wait until next time the fd is `select()` to transfer the next buffer. This prevent the server from being blocked on single user if it transfers a large amount of data.
- To switch between different user, server keep the state of each client. As for `get`, `put`, and `ls`, we will send out file length/number of entries first so that the receiver can know when the data transfer is completed. (This is not appropriate to let the sender send the ending message, because maybe the file content will include the same word as the ending message.)

SIGPIPE

What is SIGPIPE?

`SIGPIPE` is a signal occurs when we attempt to write data to a socket or pipe whose reading end has been closed. (In fact, when the socket has received RST.) As its default action is to terminate the process, we can't deal with it simply by error handling.

Will it occur in my implementation?

Yes, it will. Server can't know if the client has closed the connection when it try to `send()` data to it.

How do we handle it?

Set the `MSG_NOSIGNAL` flag for `send()`:

```
send(sockfd, buf, len, MSG_NOSIGNAL);
```

This will not generate `SIGPIPE`, but `EPIPE` will still return. The server can then handle the error and clear the client data (e.g., `FD_CLR`, `close`).