Asynchronous Chat

Foundations of Distributed Systems
Lab Guide 2

2018/2019

Consider a chat server using Java and sockets, where lines sent by any client are broadcast to all currently connected clients.

Steps

1. Implement the server using NIO2 asynchronous sockets.
2. Run clients with different delay configurations.
3. Refactor the server to work on strings internally.
4. Add a login/password validation step to connections.
5. Refactor the lines and login layers to use `CompletableFuture<>`.

Questions

1. How does code in the event driven program map to the equivalent threaded program?
2. How to manage shared structures while minimizing memory allocation?
3. Can the bytes-to-strings layer be reused?
4. What version of the login layer is easier to develop/understand?

Learning Outcomes Apply event-driven programming based on callback functions in the development of distributed programs. Employ shared data structures to reduce memory usage. Organize event driven programs as layers to manage complexity and foster reuse.

Dependency for FutureSockets

```
<dependency>
    <groupId>com.github.spullara.java8</groupId>
    <artifactId>concurrent</artifactId>
    <version>0.1-SNAPSHOT</version>
</dependency>
```