Design Document

Data Communications: Assignment 3, Comm Audio

Manuel Gonzales, A00866174, 4O  
Georgi Hristov, A00795026, 4O  
Calvin Rempel, A00871348, 4O  
Eric Tsang, A00841554, 4O

|  |
| --- |
| Table of Contents  [State Transition Diagrams 2](#_Toc413218317)  [Server 2](#_Toc413218318)  [Client 3](#_Toc413218319)  [Pseudocode 4](#_Toc413218320)  [Server 4](#_Toc413218321)  [Initialize 4](#_Toc413218322)  [Selecting 4](#_Toc413218323)  [Parse select 4](#_Toc413218324)  [Accept, add client socket 4](#_Toc413218325)  [Remove client socket 4](#_Toc413218326)  [Read 4](#_Toc413218327)  [Echo 4](#_Toc413218328)  [Update client list 4](#_Toc413218329)  [Write to file 4](#_Toc413218330)  [Client 4](#_Toc413218331)  [Initialize 4](#_Toc413218332)  [Selecting 4](#_Toc413218333)  [Parse select 4](#_Toc413218334)  [Read stdin 5](#_Toc413218335)  [Read socket 5](#_Toc413218336)  [Write socket 5](#_Toc413218337)  [Write stdout 5](#_Toc413218338)  [Write file 5](#_Toc413218339) |

# State Transition Diagrams

The state transition diagrams in this section describe the states of the server and client applications.

## Server



The diagram above illustrates the states of the server application:

* **Initialize**; creates the server socket, and sets up a set of sockets for select to monitor.
* **Selecting**; call select to monitor our set of sockets.
* **Parse select**; determine which socket caused select to return.
* **Accept, add client socket**; accept a connection from the server socket, and add them to our socket set so select can monitor it as well.
* **Remove client socket**; remove the client from our socket list, so select will no longer monitor it.
* **Read**; read data from the client socket.
* **Echo**; write the data read from the client socket to all other sockets except the one that sent it.
* **Update client list**; update our list of clients that are being displayed on std out from the select's socket set.
* **Write to file**; write out output to a file for bonus marks.

The server application can be signaled at anytime, causing it to perform a cleanup routine, and terminate.

## Client



The above diagram illustrates the states of the client application:

* **Initialize**; opens a socket that is connected to the server.
* **Selecting**; select monitors stdin, and the socket.
* **Parse select**; select has returned, determine which file descriptor made select return.
* **Handle input**; reads data from stdin, then transmits the data read from stdin to the server through the socket.
* **handle message**; reads data from the socket, then displays the received data on the terminal.
* **Write file**; record inputs and outputs to the file.

The client application can be signaled at anytime, causing it to perform a cleanup routine, and terminate.

# Pseudocode

The pseudo code is a programming-language-agnostic code-like description of what happens in each of the states in the server and client applications.

## Server

This section contains pseudo code for the server application.

### Initialize

### Selecting

### Parse select

### Accept, add client socket

### Remove client socket

### Read

### Echo

### Update client list

### Write to file

## Client

This section contains pseudo code for the client application.

### Initialize

### Selecting

### Parse select

### Read stdin

### Read socket

### Write socket

### Write stdout

### Write file