BCIT DATA COMM PROJECT

Instructor: Aman Abdulla

Winter 2015

January 25

ALEX LAM

MANUEL GONZALES

Networking Module

Client-Side Architecture v0.9

Design Document

Calvin Rempel

2015

Table of Contents

[State Diagram 1](#_Toc409979178)

[State Descriptions 2](#_Toc409979179)

[Pseudocode 3](#_Toc409979180)

[START 3](#_Toc409979181)

[DISCONNECTED 3](#_Toc409979182)

[CONNECTING 3](#_Toc409979183)

[CONNECTED 4](#_Toc409979184)

[ASSOCIATING MESSAGE HANDLER 4](#_Toc409979185)

[WAITING FOR DATA IN 4](#_Toc409979186)

[BUFFERING MESSAGE DATA 4](#_Toc409979187)

[MESSAGE RECEIVED 4](#_Toc409979188)

[MANAGE NETWORK ENTITIES 5](#_Toc409979189)

[MANAGING CONNECTION STATE 5](#_Toc409979190)

[CALLING MESSAGE HANDLERS 5](#_Toc409979191)

[SENDING NETWORK MESSAGE 5](#_Toc409979192)

[CONNECTION ERROR 5](#_Toc409979193)

[DISCONNECTING 6](#_Toc409979194)

[EXIT 6](#_Toc409979195)

[CLASS DIAGRAM 6](#_Toc409979196)

[TO DO For VERSION 1 7](#_Toc409979197)

# State Diagram



## State Descriptions

|  |  |
| --- | --- |
| **START** | A state where Network Module is being initialized in Client Mode |
| **DISCONNECTED** | A state where the Client is not connected. |
| **CONNECTING** | A state where the Client is attempting to connect to a Server. |
| **CONNECTED** | A state where the Client has successfully connected to a Server and has prepared to start sending/receiving data. |
| **ASSOCIATING MESSAGE HANDLER** | A state where Client associates a callback mechanism with a networked message of a specified type. |
| **WAITING FOR DATA** | A state where the Client is reading data from the Server on a separate blocking process. |
| **BUFFERING MESSAGE DATA** | A state where the Client has received data from the Server and is buffering it to ensure the full message is received. |
| **MESSAGE RECEIVED** | A state where the Network Read Process has buffered a complete message and has passed it back to the main process for handling. |
| **MANAGING NETWORK ENTITIES** | A state where the Client received a message that concerns Network Entities |
| **MANAGING CONNECTION STATE** | A state where the Client received a message that concerns the status of the Connection itself. |
| **CALLING MESSAGE HANDLERS** | A state where the Client has received a message that is of a user defined type. |
| **SENDING NETWORK MESSAGE** | A state where the Client is sending a command message to the Server. It must be either a non-blocking write, or multi-threaded. |
| **CONNECTION ERROR** | A state where the Client has encountered a connection error of some sort and needs to terminate the connection. |
| **DISCONNECTING** | A state where the Client is closing its’ connection and freeing any allocated resources relating to the connection. |
| **EXIT** | A state where the Network Module is no longer needed and so is freeing any resources it acquired. |

# Pseudocode

## START

Start Client/Game Logic

Initialize Network Module

Go To State: DISCONNECTED

## DISCONNECTED

IF User Registers Message Handler

{

Go To State: ASSOCIATING MESSAGE HANDLER

}

ON Connect()

{

Go To State: CONNECTING

}

## CONNECTING

Create the Socket and configure for connection to Server

Attempt to connect to Server

IF there is an error

{

Go To State: CONNECTION ERROR

}

ELSE

{

Go To State: CONNECTED

}

## CONNECTED

Create Network Read Process IPC Mechanism

FORK Network Read Process

IF new process = CHILD

Go To State: WAITING FOR DATA

ELSE IF new process = PARENT

ON user sends message

Go To State: SENDING NETWORK MESSAGE

ON user registers message handler

Go To State: ASSOCIATING MESSAAGE HANDLER

ON user disconnects

Go To State: DISCONNECTING

## ASSOCIATING MESSAGE HANDLER

Add message handler to map of message types

Go To State: previous state

## WAITING FOR DATA IN

Read data from Socket (Blocking)

IF Read Error

Overwrite receive buffer Message to “NETWORK\_STATUS\_ERROR”

Go To State: BUFFERING MESSAGE DATA

## BUFFERING MESSAGE DATA

ADD received data to buffered messages

WHILE buffer contains complete messages

WRITE Message to Main Process through Network Read IPC

ERASE Message from buffer

## MESSAGE RECEIVED

SWITCH ( MESSAGE TYPE )

CASE CONNECTION\_STATE\_MESSAGE

Go To State: MANAGING CONNECTION STATE

CASE ENTITY\_MESSAGE

Go To State: MANAGING NETWORK ENTITIES

CASE USER\_MESSAGE

Go To State: CALLING MESSAGE HANDLERS

## MANAGE NETWORK ENTITIES

SWITCH ( MESSAGE TYPE )

CASE CREATE\_ENTITY

Create new Client Network Entity

CASE DELETE\_ENTITY

Delete specified Client Network Entity

CASE UPDATE\_ENTITY

Update specified Client Network Entity

## MANAGING CONNECTION STATE

SWITCH ( MESSAGE TYPE )

CASE CONNECTION\_VERIFIED

Set Connection Verified Flag

CASE CONNECTION\_ERROR

Go To State: CONNECTION ERROR

## CALLING MESSAGE HANDLERS

Retrieve message handler from map

IF message handler is set

Call message handler

## SENDING NETWORK MESSAGE

WRITE message to Socket (non-blocking or in thread)

IF error occurred

Go To State: CONNECTION ERROR

Go To State: Connected

## CONNECTION ERROR

LOG error

Go To State: DISCONNECTING

## DISCONNECTING

IF connected

FREE session resources

KILL Network Read Process

FREE connection resources

Go To State: EXIT

## EXIT

FREE all client resources

# CLASS DIAGRAM



# TO DO For DOCUMENT VERSION 1

* Well Defined Task List
* Create Mechanism For ClientNetworkEntity Creation From Server
* Define Internal Message Structure
* Define NETWORK and ENTITY message types
* Define Connection Verification process
* Update Class Diagram
* Specify exact public interface