2015

BCIT DATA COMM PROJECT

Instructor: Aman Abdulla

Winter 2015

January 25

Networking Module
Client-Side Architecture v1.0

DESIGN DOCUMENT

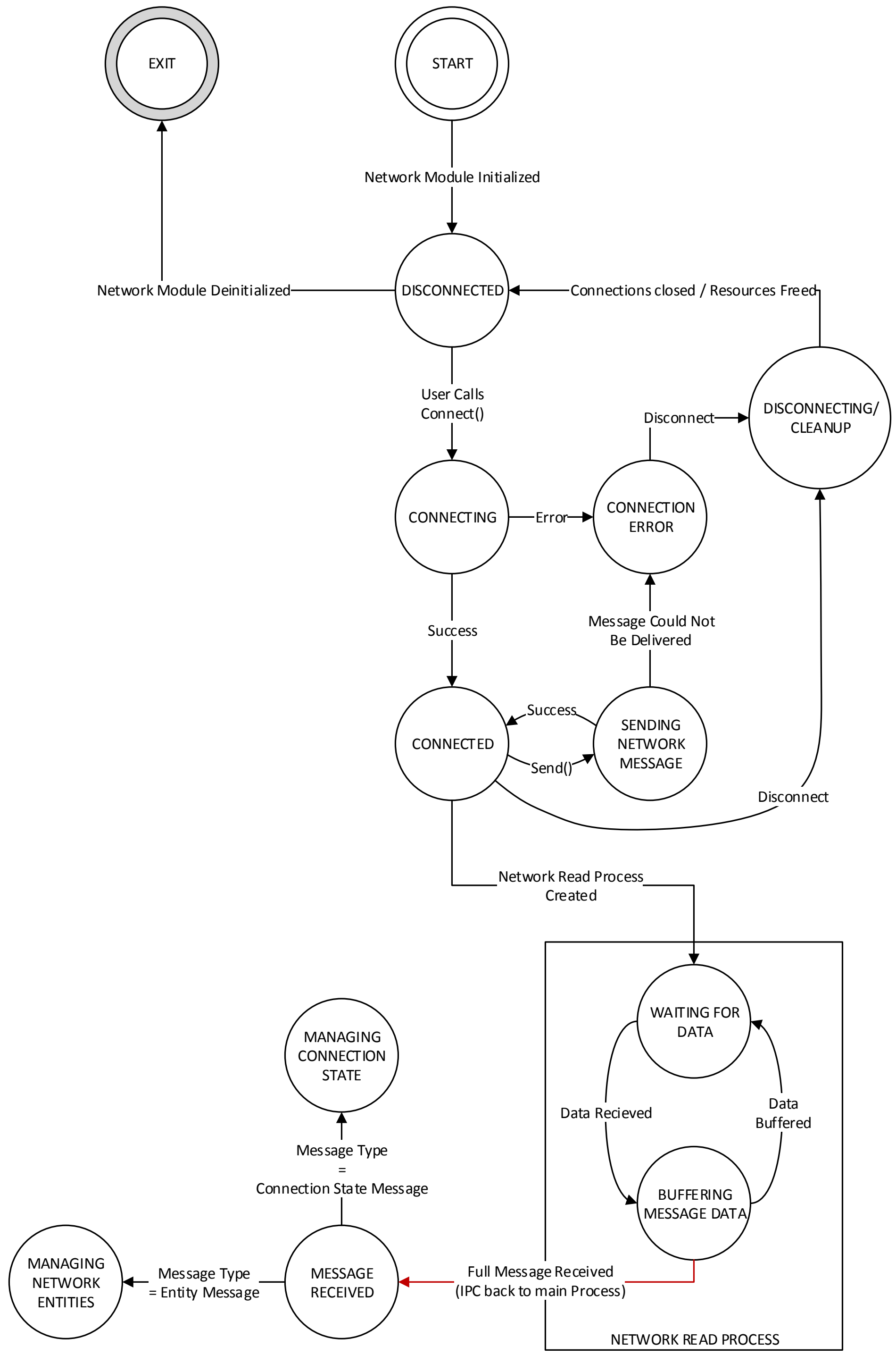
CALVIN REMPEL

ALEX LAM

MANUEL GONZALES

TABLE OF CONTENTS

STATE DIAGRAM


EXIT
START
Network Module Initialized
DISCONNECTED
Network Module Deinitialized
Connections closed / Resources Freed
User Calls Connect()
Disconnect
DISCONNECTING/ CLEANUP
CONNECTING
Error
CONNECTION ERROR
Success
Message Could Not Be Delivered
CONNECTED
Success
Send()
SENDING NETWORK MESSAGE
Disconnect
Network Read Process Created
WAITING FOR DATA
Data Recieved
Data Buffered
BUFFERING MESSAGE DATA
NETWORK READ PROCESS
Full Message Received (IPC back to main Process)
MESSAGE RECEIVED
Message Type = Connection State Message
MANAGING CONNECTION STATE
Message Type = Entity Message
MANAGING NETWORK ENTITIES

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.
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.
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
```

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
```

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
```

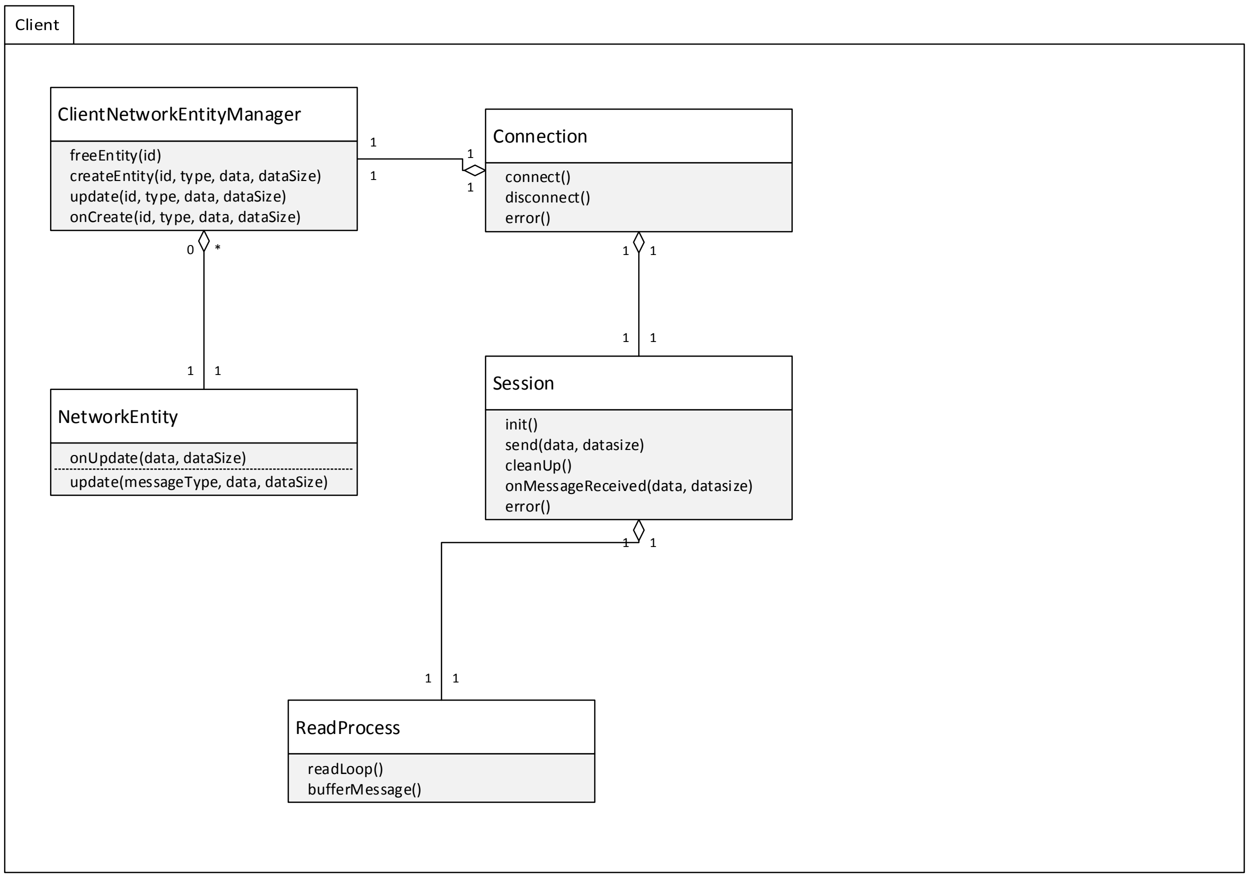
DISCONNECTING

```
IF connected
       FREE session resources
       KILL Network Read Process
FREE connection resources
Go To State: EXIT
```

EXIT

```
FREE all client resources
```

CLASS DIAGRAM


Client
ClientNetworkEntityManager
freeEntity(id)
createEntity(id, type, data, dataSize)
update(id, type, data, dataSize)
onCreate(id, type, data, dataSize)
Connection
connect()
disconnect()
error()
NetworkEntity
onUpdate(data, dataSize)
update(messageType, data, dataSize)
Session
init()
send(data, datasize)
cleanUp()
onMessageReceived(data, datasize)
error()
ReadProcess
readLoop()
bufferMessage()
1
1
0
*

TASK LIST

- Create skeleton interface implementation
- Implement NetworkEntity update pass-through to help other teams
- Implement connection to server with error handling
- Implement “clean” disconnection
- Implement sending messages to the server
- Implement receiving messages
- Implement connection management control messages and handling
- Implement NetworkEntity update sending
- Implement NetworkEntity creation/deletion
- Implement NetworkEntity updating.
- Stress test NetworkEntity system to verify performance requirements.