Server Documentation

Team Members: Andrew Burian (T.L.), Chris Holisky, German Villarreal

1 MILESTONES

1.1 MILESTONE 1 – BASE SYSTEM

This goal for **February 21, 2014**

Set up the basic program structure for the server, including all the IPC, and will be the completed framework for all future feature controllers to be plugged into.

Can be connected to by up to 2 players, and will support movement and game objectives.

Players joining the game will be immediately put on a gameplay area, and the game will start as soon as 2 players have entered the area.

1.2 MILESTONE 2 – FLOORS AND LOBBY SYSTEMS

This goal for **March 6, 2014**

Add the floor concept to gameplay. Players will first spawn in the "Lobby" floor, where they will be able to choose their teams. Gameplay starts after a set time has elapsed and a minimum number of players are present. Players are then moved to a floor by the server. Players can then move freely between gameplay floors.

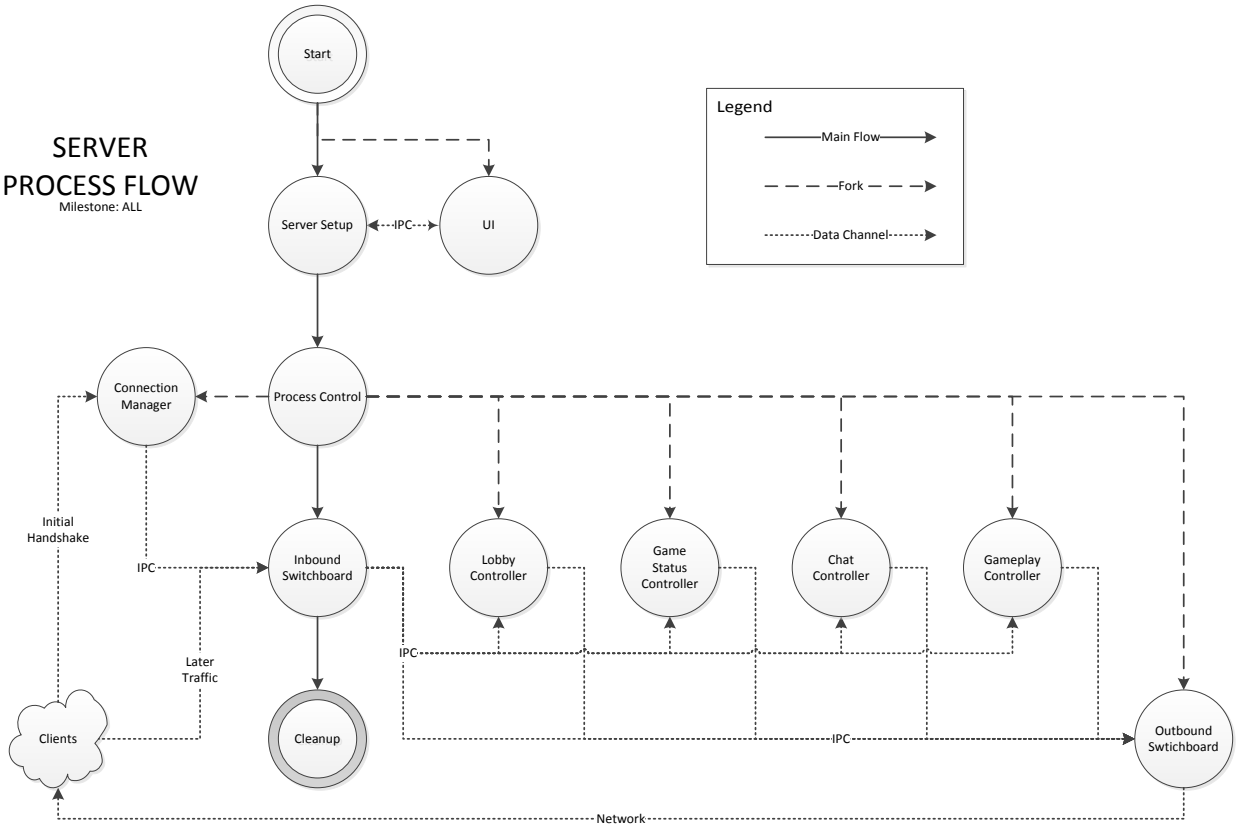
Possibly also introduce the chat feature.

1.3 MILESTONE 3 – FEATURES AND ADDITIONAL SYSTEMS

This goal for **March 20, 2014**

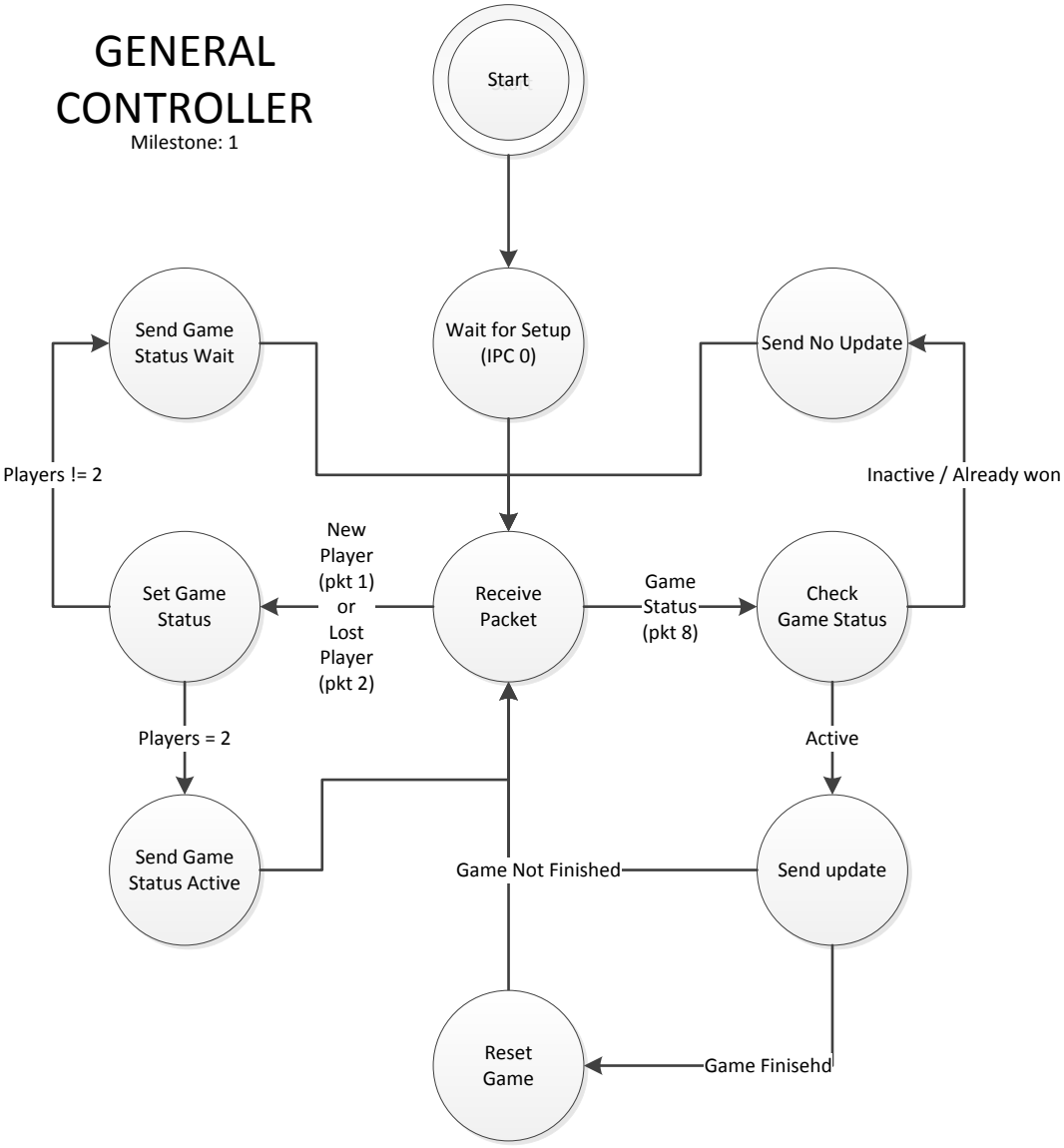
Add any additional controllers for systems including the chat, traps, alarms, and any other features.

2 STATE TRANSITION DIAGRAMS



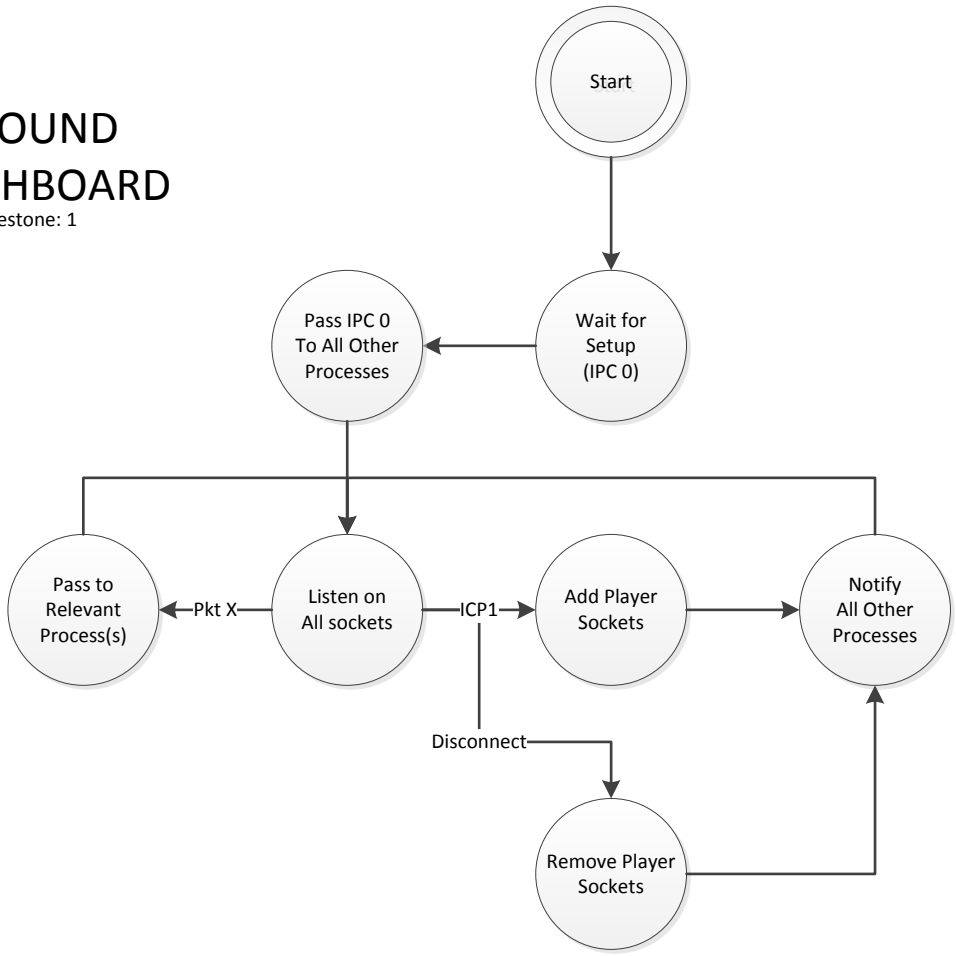
GENERAL CONTROLLER

Milestone: 1



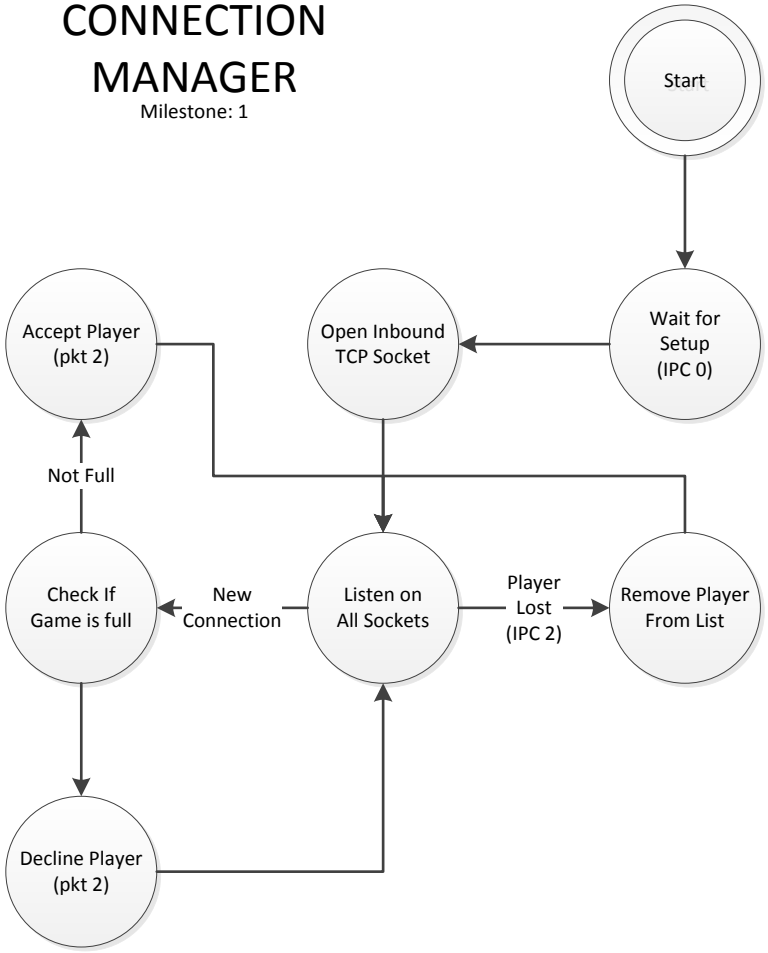
INBOUND SWITCHBOARD

Milestone: 1



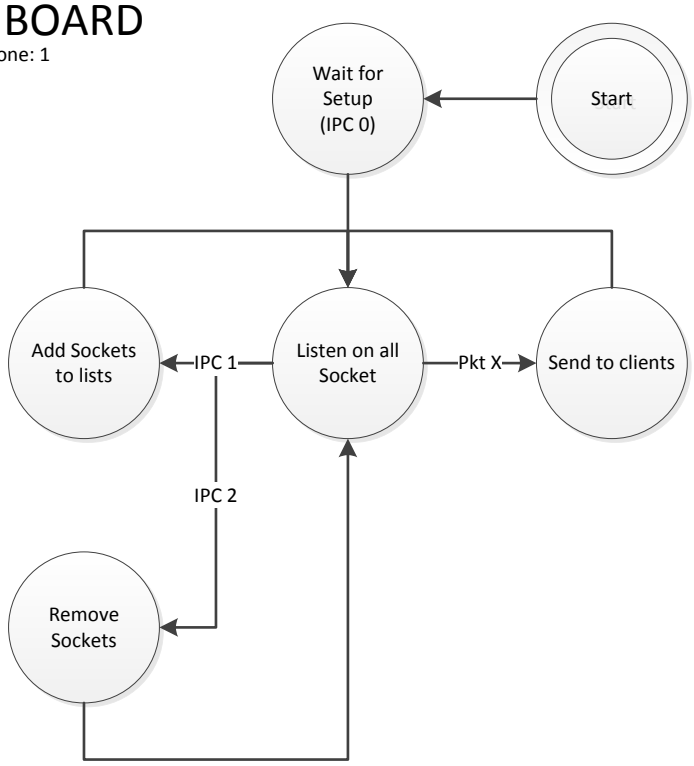
CONNECTION MANAGER

Milestone: 1



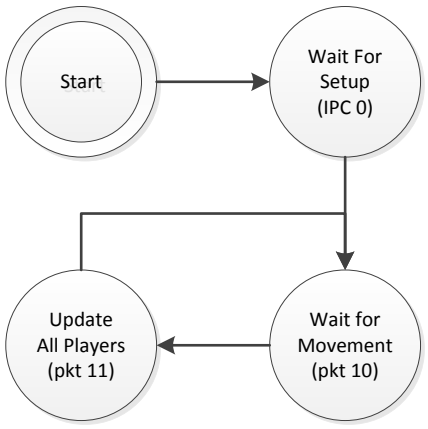
OUTBOUND SWITCHBOARD

Milestone: 1



GAMEPLAY CONTROLLER

Milestone: 1



3 PSEUDO-CODE FOR PROCESSES

```
UI
    prompt setup info
    pass IPC packet 0 to Inbound Switchboard
    while running
        get input
        if quit
            create quit packet
            send to switchboard
            exit
        if other
            other

-------------------------------------------------------------------------------------------------------

CONNECTION MANAGER
    create and bind a TCP listen socket
    loop
        accept new connection
        see if there is enough space in the game to accept a new client
        if game is full
            send pack packet 2 with game connect denied
            FIN connection
            Close socket
        if space is available
            assign new client a player number
            send packet 2 back to client with connect accepted.
            send IPC packet 1 to Inbound Switchboard

-------------------------------------------------------------------------------------------------------

INBOUND SWITCHBOARD
    allocate socket lists
    wait for IPC packet 0 from UI
    pass IPC packet 0 to Connection Manager, Outbound Switchboard, Gameplay, and General

    loop
        listen on all sockets
        if connection manager socket
            add new connection to socket to list of inbound sockets
            create a new UDP socket based on the information
            Pass the new info to Outbound Switchboard, Gameplay, and General

        if udp socket
            receive the size of the largest UDP packet
            determine which packet type it is
            cast the received data to the appropriate structure

            if packet 10 (Movement)
                pass to Gameplay

        if tcp socket
            if socket is closed or terminated
                remove the socket and it's UDP counterpart
                create IPC packet 2 (Player lost) and pass it to Connection Manager, Outbound Switchboard,
General, Gameplay
                close descriptors

            else

            read in the packet type
            fill the appropriate structure

            if packet 8 (gameplay update)
                pass packet to General
```

```
-----------------------------------------------------------------------------------------------------

GAMESTATUS CONTROLLER
    Setup
    Loop
        Run Lobby Controller
        Run in-game controller
        Run end of game controller



-----------------------------------------------------------------------------------------------------

GAMEPLAY CONTROLLER
    *just milestone1*

    wait for IPC packet 0 (setup)

    loop
        listen on ipc-socket
        read player information
        update player information
        on update tick
            send to outbound switchboard


-----------------------------------------------------------------------------------------------------

OUTBOUND SWITCHBOARD
    allocate tcp and udp socket list

    wait for IPC packet 0 (setup)

    loop
        listen on socket

        read packet type
        fill appropriate structure

        if IPC packet 1 (New player)
            add descriptors to lists

        if IPC packet 2 (Lost player)
            remove descriptors from list

        if Packet 1 - 13
            read 32b (4B) more from socket to get send-to-player flags
            send received packet to all players specified in flags
```

4 TEAM RESPONSIBILITIES

ANDREW BURIAN

1. Inbound Switchboard
2. Outbound Switchboard
3. Connection Manager
4. Keep Alive
5. Chat
6. Server-Utils

Created the server framework to support and abstract the core controllers from the details of packet transmission and client statuses.

GERMAN VILLARREAL

1. General Controller
    - 1.1. Running Controller
    - 1.2. Lobby Controller
    - 1.3. End of Game Handler

Created the Game logic and handling component of the server, responsible for all things that would affect the entire game, such as game status, objectives, player count, and chat.

CHRIS HOLISKY

1. Gameplay Controller

Created the game movement system to handle the locations of all connected players across all floors, floor change requests, as well as forced floor moves, and position updates on a timer tick.

5 NOTABLE FEATURES

Modular Design

The server is set up to facilitate quick and powerful changes.
The core of the server is its 2 controllers: general and gameplay, which are abstracted from the network and each other by means of the switchboards. These handle all communication to and from the network and any IPC communication between the controllers.
If a new feature is added that requires a new data packet, it can be added to the list of net packets (NetComm), have its size recorded (startup), set which controller(s) it is to be sent to (Inbound-Switchboard), and what protocol it is to be sent out over (Outbound-Switchboard). All these changes are 1 or 2 lines respectively, and result in a completely new packet routine being put in place.

Switchboards

The inbound switchboard is also notably unconcerned as to incoming protocol, so long as the packet is tagged correctly, it will be received and forwarded to the controller expecting it.
The outbound switchboard prevents the data streams from becoming jumbled together by having multiple controllers writing to sockets simultaneously. A packet type can be set to transmit over TCP or UDP simply by moving one case statement. Any traffic sent over UDP will automatically have a unique sequence number appended to it to prevent out of sequence traffic confusing the clients.

Efficiency

The server is designed to be fast and responsive, while maintaining high reliability.
The switchboards are centered on the Select call, which results in very quick and flexible turnover of data to the clients. The floor distinction system, as well as the outbound masking system prevent unneeded data from being sent to clients, and keep the load down on the network.

Validation

The server anticipates some out of sequence or invalid data from the client, and knows to ignore logically impossible requests or actions.

Predictive Movement

The server is easily capable of handling traffic over the wide area network, however, latency in updates is expected to increase significantly, and the potential for lost data rises sharply. To compensate for this, the server maintains a knowledge of players last known position and velocities, and performs some basic prediction if no new data has been received from the client the last cycle.

Packet Injection

For a combination of testing purposes, and server commands, the server's UI controller is able to simulate any packet that would normally be received from a client, and have the server process it

normally. This results in the “move” and “tag” commands, which simulate players requesting to change floors, or tagging other players. Any packet can be directly inputted using the “pkt <no>” command.

Keep Alive

The server maintains a list of the time of its last interactions with each client, and the time of each client’s last interaction with the server. If the server notes it has not interacted with the client within a set threshold, it will send a no-data packet simply to confirm with the client that it is still present. If no data is received from the client within a threshold, the server assumes the client is lost, and simulates a normal disconnect.

Chat

Chat comes in several forms.
Server chat is a message that comes from the server using either the “say” command, or a pre-defined game message. These messages appear in yellow on the client and have no apparent sender.
Team chat is messages sent from a team which is sent to all other players on the same team. Whist in the lobby, all players are on the same team (none) and this works as a global chat. In game, only the player’s fellow team-mates receive the messages. These appear in red.
Encrypted chat is messages that the guards intercept from the robbers communications. While in-game, a robber’s chat message is intercepted by the guards and attempts to display. The robber’s chat is encrypted with a random scramble. However, as the game progresses, the guards begin to decrypt this chat and it slowly becomes less scrambled and more readable. These messages are in grey to the guards.