(Temp) –

Communication Protocols

# Overview

*(Give a 1-2 paragraph overview of the system -- reference architectural design document for more detail. )*

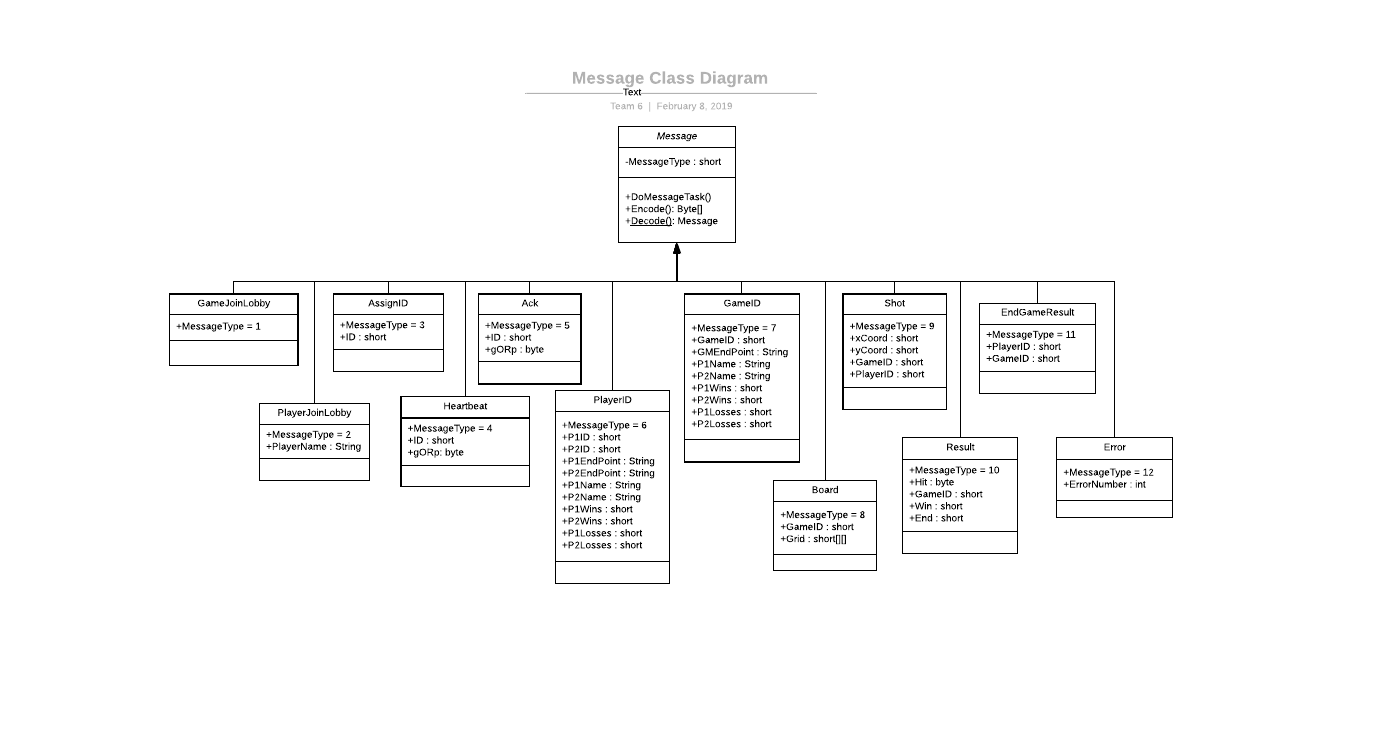
**Table 1 – Protocol List**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Purpose | Initiator | Other Processes | Pattern |
| Game Manger Joins Lobby | Establish a connection between the game manager and lobby | Game Manager | Lobby | Request Reply |
| Player Joins Lobby | Establish a connection between a player and the lobby | Player | Lobby | Request Reply |
| Pass Off | Establish a connection between the game manager and two players | Lobby | Game Manager, Lobby, Player | Secondary Channel |
| Game Ready | Establish player 1 and player 2’s boards | Player | Game Manager, Player | Request Reply Acknowledge |
| Turns | Send players’ shots to be processed for game logic | Player | Game Manager | Request Reply Acknowledge |
| End Game | End the game and reconnect game manager and players to the lobby | Game Manager | Game Manager, Lobby, Player | Request Reply |

# Messages and Shared Objects

*(Describe of messages and any shared objects that they might contain. Use UML Class Diagrams and table to help describe their structure and content.)*

*A board between the player and game manager is a shared object.*

[**](https://www.lucidchart.com/documents/edit/f688e685-58dc-4106-b1f0-c9ddb65a0482/0?callback=close&name=docs&callback_type=back&v=2719&s=692)

Game Manager Join Lobby:

(Short) Message type = 1

Player Join Lobby:

(Short) Message type = 2

(String) Name = Whatever user chose as name.

Assign ID:

(Short/Int) Message type = 3

(Short/Int) ID = Game Id for GM and Player ID for players.

Sent from Lobby to GM and P.

Heartbeat:

(Short/Int) Message type = 4

(Short/Int) ID

(Byte) gORp = game or player

ACK:

(Short/Int) Message type = 5

(Short/Int) ID

(Byte) gORp = game or player

PlayerID:

(Short/Int) Message type = 6

(Short/Int) P1ID

(Short/Int) P2ID

(String) P1 Endpoint

(String) P2 Endpoint

(String) P1 Name

(int) P1 Wins

(int) P1 Loses

(String) P2 Name

(int) P2 Wins

(int) P2 Loses

GameID:

(Short/Int) Message type = 7

(Short/Int) GameID

(String) GM Endpoint

(String) P1 Name

(int) P1 Wins

(int) P1 Loses

(String) P2 Name

(int) P2 Wins

(int) P2 Loses

Board:

(Short/Int) Message type = 8

(Short) gameID

(Short) width

(Short) hieght

(array) Grid

Shot:

(Short/Int) Message type =9

(Short) xCord

(Short) yCord

(Short) GameID

(Short) PlayerID

Result:

(Short/Int) Message type = 10

(bool) hit

(Short/Int) GameID

(bool) win

(bool) end

End Game Result:

(Short/Int) Message type = 11

(Short/Int) PlayerID = who won

(Short/Int) GameID

Error:

(short) Message Type = 12

(int) Error Number

# Communication Patterns

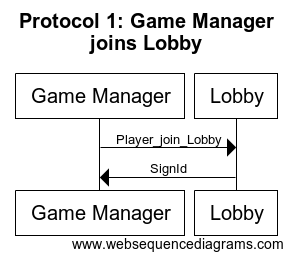
*(Identify and briefly any application-level communication patterns that your system will use. See course notes or commdp.serv.usu.edu for a list.)*

*In our system we are using three communication patterns:* Request Reply, Secondary Channel, Request Reply Acknowledge.

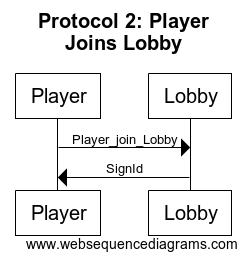
# Communication Protocols

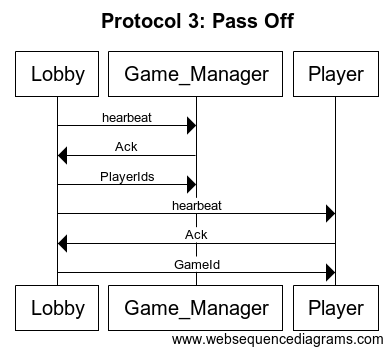
*(Describe each protocol list in Section 1, including the allowed message sequences, semantics, and expected process behaviors)*

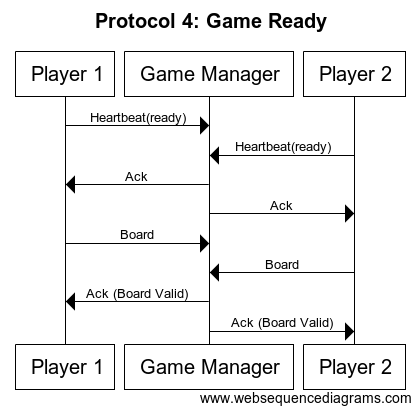
*Below we the sequence diagram describe the semantics of each protocol. Everytime a Game Manager sends a message to the Player it will by over UDP. Everytime someone sends a message to the lobby it will be over TCP.*

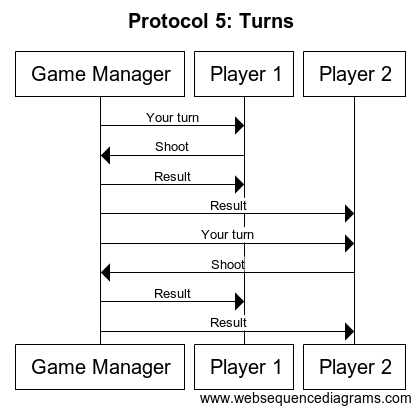
*Protocol 1-Establish a connection between the game manager and lobby*

*Protocol 2-Establish a connection between a player and the lobby*



*Protocol 3-Establish a connection between the game manager and two players*

*Protocol 4-Establish player 1 and player 2’s boards*

*Protocol 5-Send players’ shots to be processed for game logic*

*Protocol 6-End the game and reconnect game manager and players to the lobby*

