(Temp) –

Communication Protocols

# Overview

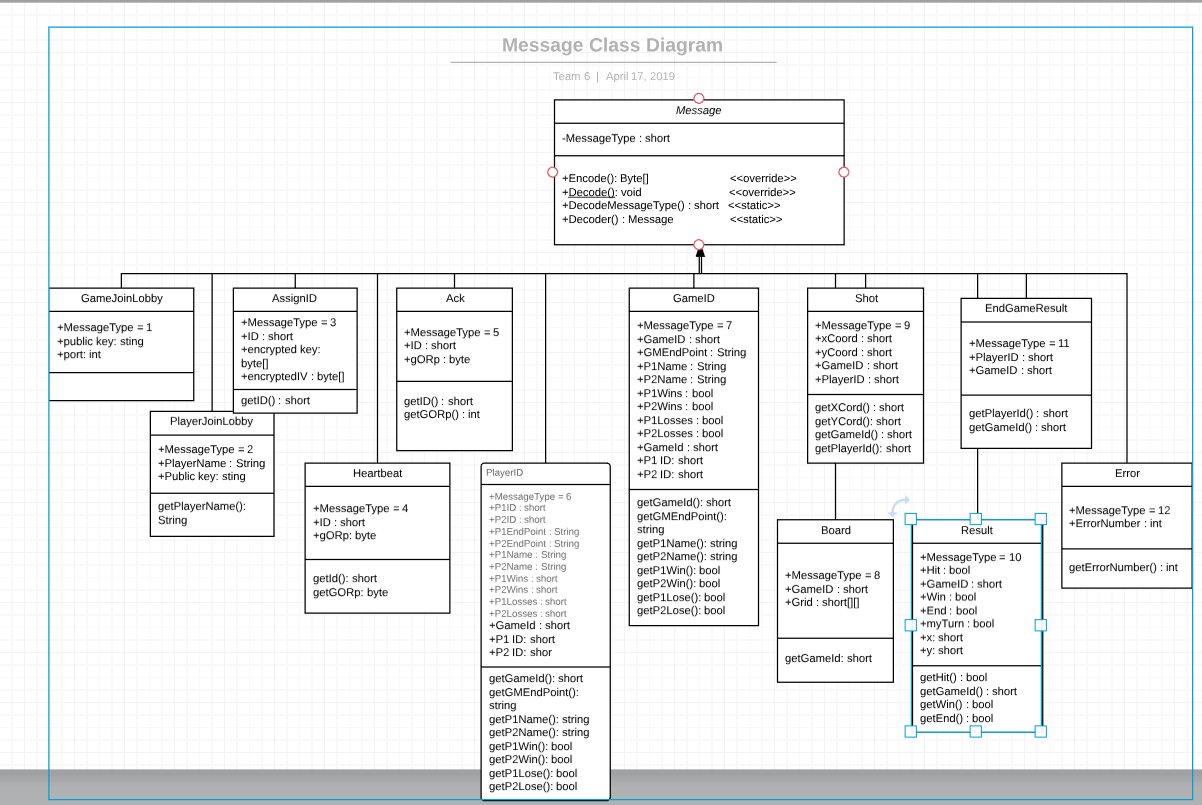
*(Give a 1-2 paragraph overview of the system -- reference architectural design document for more detail.)*

**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 | Multiple Request Reply |
| Board | Establish player 1 and player 2’s boards | Player | Game Manager, Player | Request Reply |
| Turns | Send the result of the opponent players shot | Game Manager | Player | Request Reply |
| Shot | Send players’ shots to be processed for game logic | Player | Game Manager | Request Reply |
| 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.)*



Game Manager Join Lobby:

(Short) Message type = 1

(int) port = Port to join on TCP

(string) publicKey = Public key to encrypt symmetric key in Lobby

Player Join Lobby:

(Short) Message type = 2

(String) Name = Whatever user chose as name.

(int) port = Port to join on TCP

(string) publicKey = Public key to encrypt symmetric key in Lobby

Assign ID:

(Short/Int) Message type = 3

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

(byte array) sysKey = Encrypted symmetric key.

(byte array) sysIV = Encrypted symmetric key.

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

(short) gameID

(short) p1Id

(short) p2Id

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

(short) gameID

(short) P1 ID

(short) p2 ID

Board:

(Short/Int) Message type = 8

(Short) gameID

(Short) width

(Short) height

(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

(short) x

(short) y

(bool) myTurn

End Game Result:

(Short/Int) Message type = 11

(Short/Int) PlayerID = who won

(Short/Int) GameID

Error:

(short) Message Type = 12

(int) Error Number

\*All Messages include a message number and conversation ID identifier

# 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.)*

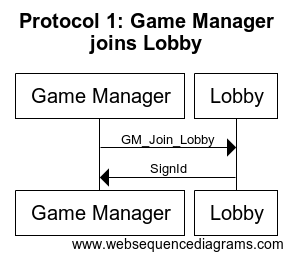
*In our system we are using the request reply communication pattern*.

# Communication Protocols

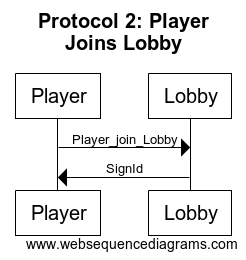
*(Describe each protocol list in Section 1, including the allowed message sequences, semantics, and expected process behaviors)*

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

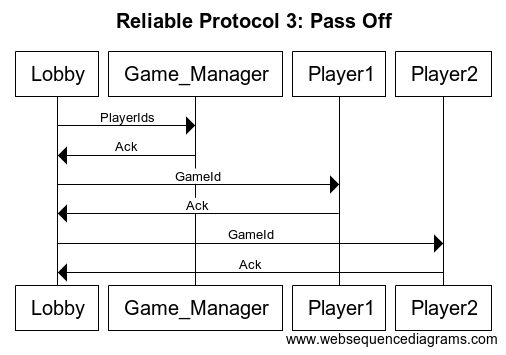
*Protocol 1-Establish a connection between the game manager and lobby*



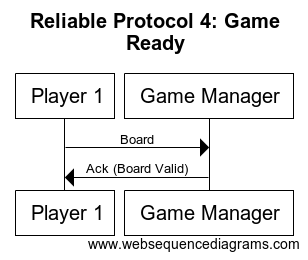
*Protocol 2-Establish a connection between a player and the lobby*



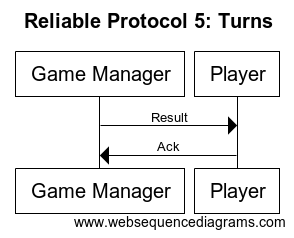
*Protocol 3-Establish a connection between the game manager and two players*



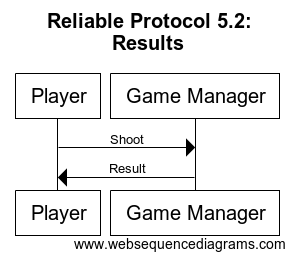
*Protocol 4-Establish player 1 and player 2’s boards*



*Protocol 5.1-Send results from opponent players shot to be processed for player’s logic*



*Protocol 5.2-Send players’ shots to be processed for game logic*



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

