Tanks 2-D

Communication Protocols

# 1 Overview

The Tanks 2-D game will include various protocols to communicate between our main processes, namely the Player (P), Game Manager (GM), and Registry (R). Each protocol listed follows a request-reply communication pattern. Table 1 lists the different protocols used to communicate between the various processes.

# 2 Message Structure

The structure of each message that will be send will have several abstractions that only specific parts of the application will know about. The structure, size, and use will be as follows:

* 4 bytes – Size of message
* 32 bytes – checksum
* Remaining data – Actual message

## 2.1 Checksum

Every message send will contain a checksum using the SHA256 hash algorithm. This will be used to help ensure that the data is received as intended.

**Table 1 – Protocol List**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Protocol | Purpose | Initiator | Other Processes | Pattern |
| Game Move | GM sends the results of a move to each P to render and act out the movement | GM | P | Request-Reply |
| Invalid Move | GM lets the P know that the move requested is not valid | GM | P | Request-Reply |
| Request Move | To send the GM the requested move by P and let the GM decide if it is valid | P | GM | Request-Reply |
| Move Accepted | GM informs P that the move is valid and rendering is started by GM for each P | GM | P | Request-Reply |
| Join Game | P requests to join a game to R | P | R, GM | Request-Reply |
| Join Game Accepted | Registry assigns P to a game with GM, then confirms adding P to game | R | GM, P | Request-Reply |
| Game Status | GM informs P's of the basic status of the game, including whose turn it is | R | P | Request-Reply |
| Heartbeat | GM makes sure each P is still active | R | P, GM | Request-Reply |
| Ack | A response to show active connection | GM, P | R | Request-Reply |

# 3 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.)*

The messages used by this system will all follow a JSON serialization format. In this section, each message will list out the information it contains and what types it includes.

Message Types:

Game Move:

GameId: int

PlayerId: int

MoveId: int

LocationY: int

LocationX: int

GunType: int

ShotAngle: int

ShotPower: int

Invalid Move:

GameId: int

PlayerId: int

MoveId: int

ErrorType: string

RequestMove:

GameId: int

PlayerId: int

MoveId: int

MoveAccepted:

GameId: int

PlayerId: int

MoveId: int

RequestAvailableGames:

GiveAvailableGames:

Games:List of GameId

JoinGame:

GameId: int

PlayerName: String

JoinGameAccepted:

GameId: int

NewPlayerId: int

(Backgroundtype: int)

GameStatus:

GameId: int

PlayerId: int

Status: int

PlayerTurn: int

# 4 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.)*

## C:\Users\Brandon\Downloads\01Request-Reply.pngRequest-Reply pattern

We will be taking advantage of the Request-Reply pattern. The figure shown is a basic sequence diagram of the pattern. The main concept is:

* A initiates request.
* B receives the request.
* A sends a message to B to begin the transfer.
* The transfer is sent from B.

# 5 Communication Protocols

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

Below are the descriptions of the conversations outlined in Section 1.

## 5.1 Game Move (Request-Reply)

This will be used by the Player and Game Manager.

Message Sequence:

Game Manager -> (GameMove) -> Players

Players -> (Ack) -> Game Manager

Behaviors:

* The Game Manager will initiate the conversation by sending a validated game move.
* This message will be sent to every Player that is a part of the game this move corresponds to.
* The GameMove object will contain enough information to properly render the move for each of the players.

## 5.2 Invalid Move (Request-Reply)

This will be sent to the Player from the Game Manager.

Message Sequence:

Player -> (RequestMove) -> Game Manager

(If move is not valid) Game Manager -> (InvalidMove) -> Player

(If move is valid) Game Manager -> (MoveAccepted) -> Player

Behaviors:

* This message will be sent by the Game Manager if a Player sends an invalid move.
* The InvalidMove object will contain the reason why it is an invalid move.
* When a Player receives this, they have two more chances to send a valid move before getting kicked off the game.

## 5.3 Request Move (Request-Reply)

This will be sent by the Player to the Game Manager.

Message Sequence:

Player -> (RequestMove) -> Game Manager

(If move is not valid) Game Manager -> (InvalidMove) -> Player

(If move is valid) Game Manager -> (MoveAccepted) -> Player

Behaviors:

* The Player will send this message with their desired move.
* The message will contain all information needed for the Game Manager to determine if the move is valid. It will also contain enough information that all other players can recreate the move.

## 5.4 Move Accepted (Request-Reply)

This will be sent to the Player from the Game Manager.

Message Sequence:

Player -> (RequestMove) -> Game Manager

(If move is not valid) Game Manager -> (InvalidMove) -> Player

(If move is valid) Game Manager -> (MoveAccepted) -> Player

Behaviors:

* This will be sent to the player that recently sent a Request Move message.
* When this is received, the player that submitted a valid move will also end their turn.

## 5.5 Join Game (Request-Reply)

This will be a request made by the Player to a Game Manager.

Message Sequence:

Player -> (JoinGame) -> Game Manager

Game Manager -> (JoinGameAccepted) -> Player

Behaviors:

* This will be sent to a Game Manager from a Player to request to join a game.

## 5.6 Join Game Accepted (Request-Reply)

This will be sent to the Player that requested to join from a Game Manager.

Message Sequence:

Player -> (JoinGame) -> Game Manager

Game Manager -> (JoinGameAccepted) -> Player

Behaviors:

* The Player will receive this message when they have successfully joined a game.
* The returned object will contain the GameId that player will use in all future messages.
* The Player will begin to accept Game Move messages and Status Messages to determine when they should send the Request Move message.

## 5.7 Game Status (Request-Reply)

This will be a message from the Registry to inform the other processes of the status of the game.

Registry -> (GameStatus) -> Players

Players -> (Ack) -> Registry

Behaviors:

* This will be a message that is periodically sent from the Registry to each of the Players.
* This will contain the information for the Player to know whose turn it is.

## 5.8 Heartbeat (Request-Reply)

This will be used by the Game Manager to determine if the Player is still alive.

Message Sequence:

Game Manager -> (Heartbeat) -> Player

Player -> (Ack) -> Game Manager

Behaviors:

* The Player must respond promptly to each of the heartbeat messages that are sent by the Game Manager.
* If more than 5 in a row are not responded to, the Game Manager will forfeit the Player and inform the Registry of the forfeit.

## 5.9 Ack (Request-Reply)

This will be used by the Game Manager and Player to acknowledge the previous message.

Message Sequence:

Heartbeat:

Game Manager -> (Heartbeat) -> Player

Player -> (Ack) -> Game Manager

Game Status:

Registry -> (GameStatus) -> Players

Players -> (Ack) -> Registry

Game Move:

Game Manager -> (GameMove) -> Players

Players -> (Ack) -> Game Manager

Behaviors:

* This will be the general use reply to all protocols to acknowledge the previous message.
* There won’t be any information transferred with this message aside from the reply code for the previous message.