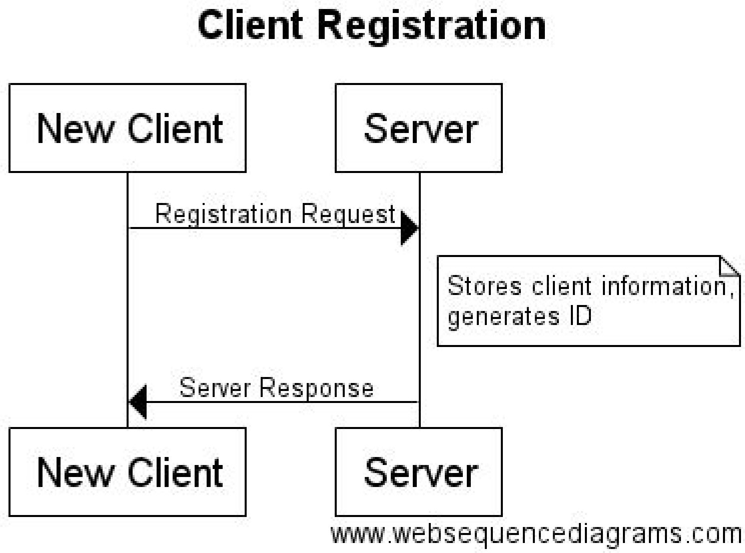
Tank Game Protocol Specification

Written by Jonathan Nelson

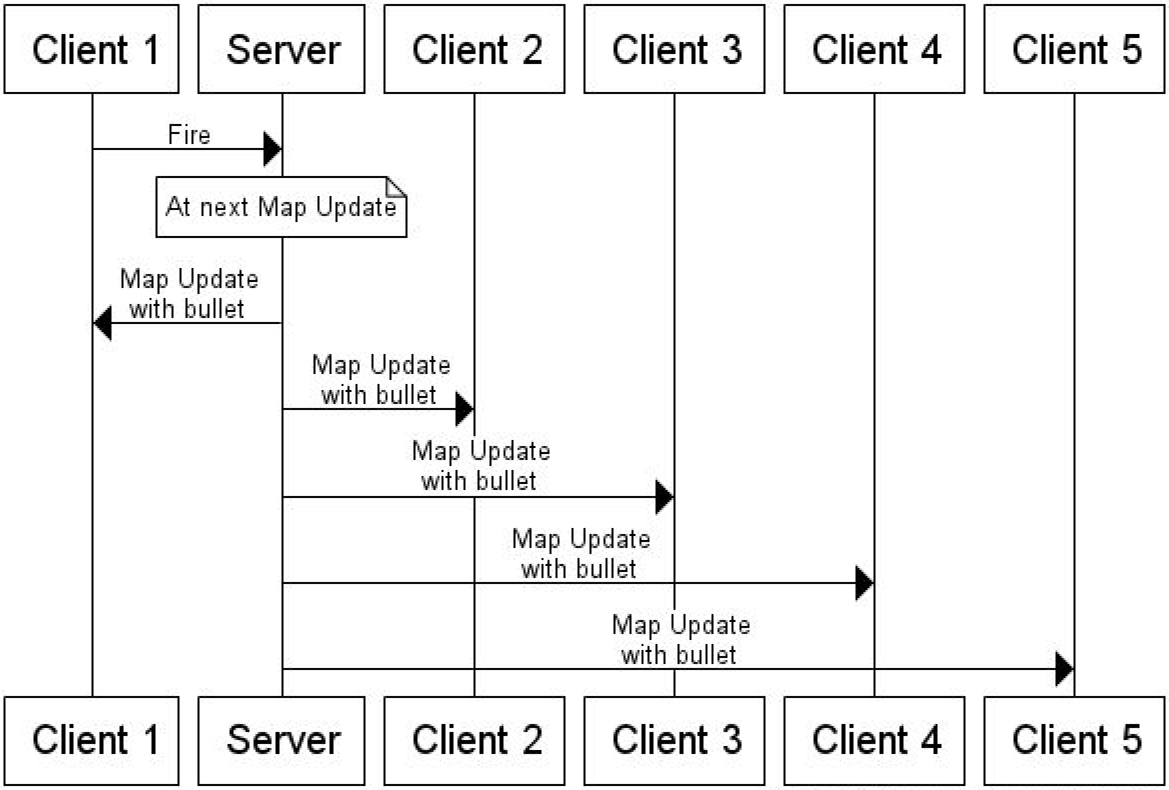
Network Structure

The Tank Game protocol uses a client/server model. The server is the authoritative data source for all clients.
Messages are sent over persistent TCP connections. This ensures that the data arrives from the server. Messages are JSON objects contained in the TCP packet payload. Text formatting is UTF-8.

Sequence Diagrams

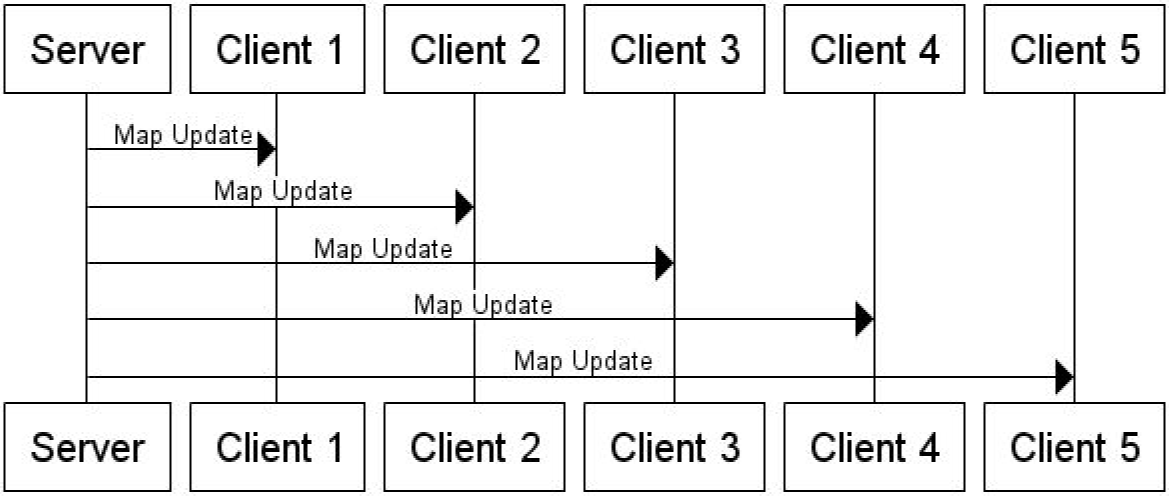


Fire Message



www.websequencediagrams.com

Map Update



www.websequencediagrams.com

Data Formats

This specification only applies to the network messages. The storage and manipulation of this data on the client and server is beyond the scope of this specification.
Messages use JSON format terminated in a newline character (\n). The messages are displayed with line breaks for clarity but should not include breaks when transmitted except to indicate the end of a message.
The center coordinates of the map are at 0, 0. Positive Y is toward the top of the screen, positive X is toward the right of the screen.
Rotation is measured in degrees with 0° being along the positive X axis.

CLIENT: Connection request:

Initial connection message from game client. Sent after network connection established. Username is a user generated name to identify the player; preferably would be unique to a session however not required.

```
{
  "action": "register",
  "username": "<string>"
}
```

SERVER: response:

Connection response from server. Contains server generated ID unique to each client instance.

```
{
  "messageType": "response",
  "id": "<string>"
}
```

CLIENT: Player fire request:

Message from client that player is firing their gun.

```
{
  "action": "fire",
  "id": "<string>"
}
```

CLIENT: continue after death request:

Message requesting to continue play after player is killed: respawn request.

```
{
  "action": "continue",
  "id": "<string>"
}
```

CLIENT: change movement request:

Message from client to server indicating that the client is moving or changing direction.

Drive parameter indicates forward motion with a positive 1, reverse motion with negative 1, and no motion with a zero.
Rotate parameter indicates clockwise motion with a positive 1, counter-clockwise motion with a negative 1, and not rotational motion with a zero.

```
{
  "action": "move",
  "id": "<string>",
  "drive": <forward/none/backward: int>,                 // -1, 0 or 1
  "rotate": <counter_clockwise/none/clockwise: int>      // -1, 0 or 1
}
```

SERVER: Update game state

Game update message from server to client.
Root elements of map element (ie, obstacles, dimensions, players, or bullets, but not including map itself or messageType) are optional. If a root element is present, however, it must be complete. For example, an update message is not required to have the players array, however if present the players array must contain ALL players.

Map element describes the map and all elements.
Obstacles describes the location and size of all obstacles on the map.
Dimensions describes the size of the map itself in floating point units. The origin of the coordinate system is in the center of these dimensions. TankRadius describes the size of all tanks in these floating point units.
Players describes the position, heading, and name of each player as well as whether they are alive at the current moment.
Bullets describes each bullet on the map that has been fired with their position and heading.

```
{
  "messageType": "update",

  "map": {
    "obstacles": [{
      "x": <float>,
      "y": <float>,
      "radius": <float>
    }...],

    "dimensions": {
      "height": <float>,
      "width": <float>,
      "tankRadius": <float>
    },

    "players": [{
      "id": <string>,
      "x" : <float>,
      "y":  <float>,
      "heading": <degrees: float>
      "username": <string>,
      "alive": <boolean>
    }...]

    "bullets": [{
      "id": <string>,
      "x" : <float>,
      "y":  <float>,
      "heading": <degrees: float>
    }...]
  }
}
```