## Status of this Memo

## Copyright Notice

## 1. INTRODUCTION

This document specifies how the R-Type client-server UDP Protocol has been designed. UDP is described in [RFC768].

### 1.1 Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 2. PROTOCOL

This document specifies a precise and exhaustive report about the R-Type project due for our third year at EPITECH.

This project is inspired from the 1987 arcade game “R-Type” from Irem.

This protocol is intended to provide the R-Type communication service, and be used between clients and servers on host computers. Typically the clients are on workstation hosts and the servers on mainframe hosts.

## 2. INTEREST

This RFC is being distributed to members of the Internet community in order to solicit their reactions to the proposals contained in it. While the issues discussed may not be directly relevant to the research problems of the Internet, they may be interesting to a number of researchers and implementers.

## 3. STATUS REPORT

In response to the need for maintenance of current information about the status and progress of various projects in the Internet community, this RFC is issued for the benefit of community members. The information contained in this document is accurate as of the date of publication, but is subject to change.  Subsequent RFCs will reflect such changes.

## 4. TECHNICAL INFORMATION

### 4.1 Client-Server Communication Protocol

The Client-Server communication uses the UDP protocol. Even if communication stability is RECOMMENDED, packets losses are negligible because of the repeatability of the data exchanged. In a matter of fluidity of the game, packets SHALL NOT be checked and blocked as it is the case in the TCP protocol [RFC793] in order to optimize the communication between the server and the clients.

### 4.2 Datagram Format

The datagram is formatted to contain an identification byte called “opcode” and a sequence of bytes corresponding to the data of the serialized structure defined by the opcode.

```
+----------+-----------------------------+
|  OPCODE  |  SERIALIZED STRUCTURE DATA  |
+----------+-----------------------------+
|  1 byte  |   structure size x 1 byte   |
+----------+-----------------------------+
```

### 4.3 Data Serialization

Every datagram MUST contain an opcode that identifies a structure known by the two connected entities.
The data of the structure is serialized byte per byte into the datagram immediately after.
The size and type of the serialized structure is hence recognized by the opcode and MAY be interpreted depending of the quality of the connection and the data losses.

## 5. OPCODES

### 5.1 Server to Client

| MACRO | VALUE | DESCRIPTION |
|---|---|---|
| CONNECT_RESPONSE | 0x0000 | Sends success or failed |
| CHOOSEN_NAME_RESPONSE | 0x0001 | Sends success or failed for name setting |
| CREATE_POOL_RESPONSE | 0x0002 | Sends success or failed for game creation |
| LIST_POOL_RESPONSE | 0x0003 | Sends list of the name of the pools |
| PLAYER_JOIN_POOL | 0x0004 | Informs the clients that a player has joined the pool |
| PLAYER_STATUS | 0x0005 | Informs the clients of a player status (ready or not) |
| PLAYER_QUIT | 0x0006 | Informs the clients that a player has exited the pool |
| JOIN_POOL_RESPONSE | 0x0007 | Informs the client if the join request is successful |
| LAUNCH_GAME | 0x0008 | Informs the clients that the game can start |
| PLAYER_POS | 0x0009 | Informs the players position at start of game |
| ACTION | 0x000A | Informs that a player made an action |
| PLAYER_DIE | 0x000B | Informs that a player died |
| NEW_ENNEMY | 0x000C | Informs the clients that an enemy has spawn |
| END_GAME | 0x000D | Informs the clients that the game is over |
| DATA_MAP | 0x000E | Sends the map data to the clients |
| ENNEMY_DIE | 0x000F | Informs that an enemy is dead |
| DISCONNECT | 0x0010 | Disconnect a client |
| MISSILE_EXPLOSE | 0x0011 | Inform that a missile has exploded |
| LIST_MAP_RESPONSE | 0x0012 | Sends the map list to the clients |
| ENNEMY_ACTION | 0x0013 | Informs that an ennemy has made an action |
| NEW_MISSILE | 0x0014 | Informs that a missile has been launched |
| NEXT_FRAME | 0x0015 | Increments the timer |
| PLAYER_LIFE | 0x0016 | Informs a player he lost a life |
| NEW_BONUS | 0x0017 | Informs a bonus has appeared |

| | | |
|---|---|---|
| BONUS_ERASE | 0x0018 | Informs a bonus has disappeared |

## 5.2 Client to Server

| MACRO | VALUE | DESCRIPTION |
|---|---|---|
| CONNECT_REQUEST | 0x0000 | Request for authentication |
| CHOOSEN_NAME_REQUEST | 0x0001 | Request to choose the name |
| CREATE_POOL_REQUEST | 0x0002 | Request to create a pool |
| CHOOSEN_MAP | 0x0003 | Sends the selected map |
| LIST_PLAYER_REQUEST | 0x0004 | Requests the list of players in the pool |
| LIST_POOL_REQUEST | 0x0005 | Requests the list of names of the pools |
| JOIN_POOL_REQUEST | 0x0006 | Requests to join a pool |
| STATUS | 0x0007 | Sends client status (ready/not ready) |
| QUIT_POOL | 0x0008 | Client has exited the pool |
| PLAYER_ACTION | 0x0009 | Informs that an action was made |
| DIE | 0x000A | Informs that the player died |
| READY_TO_GAME | 0x000B | Informs that the player is ready to play |
| PLAYER_FINISH | 0x000C | Informs that a player arrived to the end |
| LIST_MAP_REQUEST | 0x000D | Requests a list of the maps in the server |
| PLAYER_DISCONNECT | 0x000E | Informs that the client is disconnecting |

## 6. STRUCTURES

### 6.1 Client to Server

#### 6.1.1 NameInfos

Request for changing the player’s name.

| TYPE | NAME |
|---|---|
| char[] | name |

#### 6.1.2 PoolNameInfos

Request for creating and joining a pool.

| TYPE | NAME |
|---|---|
| char[] | poolName |

#### 6.1.3 MapNameInfos

Request for selecting the map.

| TYPE | NAME |
|---|---|
| char[] | mapName |

#### 6.1.4 PlayerStatusInfos

Request for receiving the player’s status in game.

| TYPE | NAME |
|---|---|
| bool | isReady |

#### 6.1.5 ActionInfos

Request for sending player’s moves in game.

| TYPE | NAME |
|---|---|
| e_action | action |

## 6.2 Server to Client

### 6.2.1 ConnectInfos

Request for connection answer.

| TYPE | NAME |
|---|---|
| bool | usSuccess |
| int | id |

### 6.2.2 ChoosenNameInfos

Request for answer to selection of name.

| TYPE | NAME |
|---|---|
| bool | isSuccess |
| char[] | name |

### 6.2.3 CreatePoolInfos

Request for answer to pool creation.

| TYPE | NAME |
|---|---|
| bool | isSuccess |
| char[] | name |

### 6.2.4 PoolInfos

Request for answer to pool listing.

| TYPE | NAME |
|---|---|
| char[] | poolName |

6.2.5 JoinPoolInfos

Request for sending the name of a player who’s joined the pool.

| TYPE | NAME |
|---|---|
| char[] | name |

6.2.6 StatusInfos

Request for sending a player’s status.

| TYPE | NAME |
|---|---|
| char[] | name |

6.2.7 QuitInfos

Request for sending that information that a player has disconnected.

| TYPE | NAME |
|---|---|
| char[] | name |

6.2.8 JoinPoolRespInfos

Request for answer to joining the pool.

| TYPE | NAME |
|---|---|
| char[] | name |

6.2.9 PlayerPosInfos

Request for sending a player’s.

| TYPE | NAME |
|---|---|
| char[] | name |

## X. AUTHORS

F. Abgrall
B. Acca
T. Raballand
V. Thibaud
J. Thompson