EPITECH Paris
Request for Comments : 0
Category : Standard Tracks

S. Jaboulet
A. Mahoussi
P. Lin
A. Lenfant
M. Tonye Mbog
EPITECH
October 2023

R-TYPE Protocol

Status of this Memo

This memo is the official specification of the UDP R-TYPE Protocols. R-TYPE is a third year project from EPITECH.

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as InternetDrafts. Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress." The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html This Internet-Draft will expire on June 2017.

Copyright Notice

Table of contents

1. Introduction

This document describes a protocol used in a copy of the R-TYPE video game. This copy has an implemented online multiplayer feature, where a server and multiple clients can communicate in order to display the game.

The R-TYPE we will be refering to in this memo will thus be the online multiplayer copy of the game.

2. 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 RFC 2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying significance described in RFC 2119.

3. R-TYPE Architecture

The game uses a client-server architecture in order to function. For its game engine, an ECS (Entity Component System) is used for both the client and the server. An UDP protocol is used for the communication between the server and the client. The server can run several lobbies simultaneously at the same time using multithreading.

4. UDP Protocol

The entirety of the communication between the server and the client relies on an UDP protocol in order to share payloads.

The game uses the UDP Protocol over the TCP Protocol since it is more adapted for our purpose. While TCP is more reliable, it transfers data more slowly. UDP is less reliable but works more quickly.

4.1. Contents Of Payload

In order to communicate between the client and the server, these MUST be the contents of a payload:

```
PACKET_TYPE type;
struct data;
```

With "type" being an element of the "PACKET_TYPE" enumeration:

```
enum PACKET_TYPE {
   NEW_CONNECTION,
   DISCONNECTION,
   MOVE_PLAYER,
   SHOOT_PLAYER,
   READY_PLAYER,
   KILL_ENTITY,
   NEW_CLIENT,
   START_GAME,
   NEW_ENTITY,
   ENTITY_MOVE,
   UPDATE_POSITION,
   KILL_ENTITY_TYPE,
   NEW_CLIENT_RESPONSE,
   SEND_LOBBYS,
   NUMBER_PLAYERS_IN_LOBBY,
   UPDATE_ENTITY_HEALTH,
   UPDATE_ENTITY_SCORE,
   END_GAME,
   NEW_LEVEL,
   NEW_CONNECTION_SINGLE
};
```

And with "data" being a structure that changes depending on the PACKET_TYPE so be sent.

The variable "type" MUST be an element of the "PACKET_TYPE" enumeration.
The structure "data" MUST have the necessary information depending on the "type" being sent.

From now on, every time this document mentions "type" and "data", it will refer to the PACKET_TYPE and the structure.

4.2. Connection To Server

When launching the game, the client MAY enter his login information. The client MAY provide a Username, an IP and a Port so as to connect to the server.
If the client doesn't provide these informations, he MUST automatically be given these login informations:

```
Username = player
IP = 127.0.0.1
Port = 8080
```

This causes the client to send to the server a payload containing:

```
type = NEW_CONNECTION
data = Username
```

With "Username" being the username chosen by the player.

This causes the client to automatically join a room. For more information, please refer to *4.3. Connection To Room*.
In response, the server sends to all the clients in the room:

```
type = NEW_CLIENT
data = Username, readyState (of all the players in the room)
```

With "Username" being the username of each player in the room and "readyState" being the status of each player in the room (ready or not ready).

On failure, no message is sent between the client and the server.

4.3. Connection To Room

Now that the client is connected to the server, the client will automatically be assigned to a room. If no room exists or all rooms are full, a new room will be created. This way, the client will join the first non-full room that the server finds for him. Otherwise, the server MUST create a new room.

To leave a room, the client MUST send to the server:

```
type = DISCONNECTION
```

4.4. Start The Game

Now that the client has joined a room, all clients in the room MUST be ready in order to start the game. To do so, each client must send to the server the following:

```
type = READY_PLAYER
```

Once all the players are ready, the server MUST send to all the clients in the room:

```
type = START_GAME
```

Once all the players in a room have died, the server MUST send to the client the following:

```
type = END_GAME
```

4.5. Ingame Communication

During the game, the client is limited down to only a few inputs, such as moving, shooting and leaving. In order to send this information to the server, the client MUST send the following:

```
type = MOVE_PLAYER
data = Direction
```

The client MUST give a direction to move to (UP, DOWN, LEFT, RIGHT).

If the client wishes to shoot, he MUST send the following to the server:

```
type = SHOOT_PLAYER
```

If the client wishes to leave the game, he MUST send the following to the server:

```
type = DISCONNECTION
```

5. References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

Authors' Adresses

Sean Jaboulet
EPITECH Paris
sean.jaboulet@epitech.eu

Pascal Lin
EPITECH Paris
pascal.lin@epitech.eu

Aurélien Lenfant
EPITECH Paris
aurelien.lenfant@epitech.eu

Miles Tonye Mbog
EPITECH Paris
miles-alexandre-nicholas1.tonye-mbog@epitech.eu

Alberick Mahoussi
EPITECH Paris
alberick.mahoussi@epitech.eu