RTYPE 2023 Kevin NADARAJAH
EPITECH Taha ALANI
PARIS Jonathan YAKAN
RFC 7841 Mehdi DJENDAR
Rayan ES-SEFFAR

R-Type Protocols

Abstract

RFC documents contain a number of fixed elements such as the title page header, standard boilerplates, and copyright/IPR statements. This document describes them and introduces some updates to reflect current usage and requirements of RFC publication. In particular, this updated structure is intended to communicate clearly the source of RFC creation and review. This document obsoletes RFC 5741, moving detailed content to an IAB web page and preparing for more flexible output formats.

Status of This Memo

This memo is the official documentation of the R-Type UDP Protocols. The R-Type is a third-year project from EPITECH.

Table of Contents

1. Introduction

The R-Type architecture is a client-server game architecture. The logic of the game is in both the server and the client; however, the server has authority over the clients. A client connects to the server using the UDP Protocol and once connected to the server he spawns in the game.

2. Protocol

2.1. Packets

When the Player is in the game, the client-server communications are done by using this R-Type UDP Protocol.

The UDP Protocol is used to transmit to all clients when an ENTITY has been updated.

In order to communicate with the server/client, we MUST follow the following Packet containing this 7 datas:

```
MAGIC_NUMBER;
PACKET_TYPE;
TIMESTAMP;
ENTITY_ID;
TYPE_INDEX;
UUID;
component;
```

where "MAGIC_NUMBER" is a 4-byte to know if the Packet is correct, "PACKET_TYPE" is a 1-byte an element of the PacketType enumeration (described below) representation the type of Packet, "TIMESTAMP" is the time when the Packet was sent, "ENTITY_ID" is the data to know which player sent the Packet, "TYPE_INDEX" is to know which component has been updated of the ENTITY_ID, "UUID" is the unique generated for detecting a particul packet.

The PacketType enumeration:

```
PacketType {
      DATA_PACKET = '0',
      REPEAT_PACKET = '1',
      RESPONSE_PACKET = '2',
      NEW_CONNECTION = '3',
      ASK_ENTITY = ‘4’
};
```

To send a Packet, you MUST write the data on UDP socket in this order:

MAGIC_NUMBER, PACKET_TYPE, TIMESTAMP, ENTITY_ID, TYPE_INDEX, UUID, COMPONENT;

To receive a Packet you MUST read the 4 first bytes (corresponding to the MAGIC_NUMBER data), read the 1 following byte (corresponding to the PACKET_TYPE), read the 8 following bytes (corresponding to the TIMESTAMP), read the 4 following bytes (corresponding to the ENTITY_ID), read the 4 following bytes (corresponding to the ENTITY_ID) and finally read the 37 last bytes (corresponding to the UUID).

2.2. Player Events

The player, on the client-side, can execute several actions.

On each action, the client MUST send to the server:

For example: Player 1 move UP

```
{

      MAGIC_NUMBER = 45678
      PACKET_TYPE = DATA_PACKET
      long TIMESTAMP = time (17090988)
      ENTITY_ID = 1 (PLAYER 1)
      TYPE_INDEX = typeid
      UUID = "eyzue1tuyzuyezyizyezyeieiye"
      COMPONENT = POS(15, 10)

 }
```

then the server sends to allClient the query if it is correct, if not the server sends nothing back.

2.3. Packet Types

In order, to specify each packet we need to specify a PACKET_TYPE in the header.

The client sends to server the packet to notify that a component of the player changed.

OR

The server sends to all clients the packet to notify that a component of a player changed.

We have to send the packet containing:

PACKET_TYPE = DATA_PACKET

The server sends, in a laps of 200ms, the same packet if the client does not receive it.

In order to know if the client received it, the client MUST respond when he receives.

The clients have to send the packet containing:

PACKET_TYPE = RESPONSE_PACKET

Until the server does not receive it will receive in loop the packet. With the following header:

PACKET_TYPE = REPEAT_PACKET

R-Type Protocols