Category: Network
R-Type AAA Team

S. Racaud
Epitech Rennes
November 2021

R-Type protocol

Copyright Notice

Table of Contents

1. Introduction

Our project engine is an ECS so the following protocol is adapted to that kind of architecture. An ECS is an architectural design pattern, mostly used in video game development. This pattern follows the principle of composition over inheritance. Instead of defining an inheritance tree as usual in Object Oriented Programming, types are split into small yet highly reusable components.

1.1. Tram types

In order to handle the application requirements, we must define different types of data tram. Each of those trams can be depict by a C data structure.

1. Tram header
2. Get room request
3. Get room reply
4. Create room request
5. Join room request
6. Join and Create room reply
7. Quit room request
8. Create entity request
9. Create entity reply
10. Destroy entity request
11. Sync component

More details in the following sections.

2. Tram header

The tram header must be designed as follows:

- magic number
- tram size
- tram type

2.1. Magic number

The magic number allow to check if the tram received is correct.

2.2. Tram size

Is the total size of the tram in octet.

2.3. Tram type

The tram type describe what kind of data structure is used after the header.
The types recognised are:

- GET_ROOM_LIST
- ROOM_LIST
- CREATE_ROOM
- JOIN_ROOM
- JOIN_ROOM_REPLY
- QUIT_ROOM
- CREATE_ENTITY
- CREATE_ENTITY_REPLY
- DESTROY_ENTITY
- SYN_COMPONENT

3. Game Room listing

Allow to communicate the list of game rooms available from the server to the client.

The data MUST bu organised as follows:

- number of item
- item list

3.1. Number of item

Number of item in the following list.

3.2. Item list

Each item MUST be an 8 bytes number corresponding to the id of a game room.
The items are stocked one after the others.

4. Game Room creation

Ask to create a new game room on the server.

That type of request it sent by a client to the server.

The request’s body is empty.

5. Game Room connection

Ask to join a game room. That request is sent by a client to a server.

The request body must be defined as follows:

- room id | 8 bytes

5.1. Room id

The room id must be a 8 bytes unsigned integer. It contain the id of a valid game room.

5.2. Request reply

The reply must be designed as follows:

- Accept | 1 bytes
- Room id | 8 bytes
- Start timestamp | 8 bytes

5.2.1. Accept

Accept is a Boolean. It tell the client if the server accept that it join a specific game room.

5.2.2. Room id

The room id must be a 8 bytes unsigned integer. It contain the id of a valid game room.

5.2.3. Start timestamp

The moment when the game will start. The client must start it’s game engine at that precise instant. It’s used to synchronize the each clients.

6. Game Room disconnection

Sent by a client to the server if he quit a game room.

The request’s body MUST be empty.

After receiving that request, the server will close the network connection with the client and tell to the others clients to remove that player from their game engine.

7. Game entity creation

Request sent by the clients or the server.

7.1. Request

The request body must be structured as follows:

- Entity local id | 4 bytes unsigned
- Entity type | 10 bytes string
- Timestamp | 8 bytes unsigned

7.1.1. Entity local id

The local id of the source client’s engine. That id isn’t used by the server but it will be integrated in the reply.

7.1.2. Entity type

The type of the created entity. That type allow to call the right entity factory.

7.1.3. Timestamp

The moment when the entity was created. Used by the rollback.

7.2. Reply

The tram body must be structured as follows:

- Accept | 1 byte
- Entity local id | 4 bytes unsigned
- Network id | 4 bytes unsigned

7.2.1. Accept

Accept is a Boolean. It tell the client if the server accept the entity creation.

7.2.2. Entity local id

The client local engine’s entity id sent in the request.

7.2.3. Network id

The new network id of the entity. An network id is an unique id of a shared entity. It allow to avoid entity id collision.

8. Game entity destruction

That request is sent by a client or the server to each others.

The tram must be structured as follows:

- Network id | 4 bytes unsigned

8.1. Network id

The unique id of the removed entity.

9. Game entity component synchronization

Can be sent by a client or the server.

The tram must be structured as follows:

- Body size | 8 bytes unsigned
- Network id | 4 bytes unsigned
- Timestamp | 8 bytes unsigned
- Component type | 8 bytes unsigned
- Component size | 8 bytes unsigned
- Component

9.1. Body size

Size of the body data tram in byte.

9.2. Network id

Id of the entity that contain the component.

9.3. Timestamp

Moment when the update was dispatched by the emitter. Used by the rollback module.

9.4. Component type

Type index of the component contained in the tram.

9.5. Component size

Size in byte of the contained component.

9.6. Component

A pointer to the component data structure. The data structure is located just after the pointer in memory.

10. Authors' Addresses

Simon Racaud
Epitech Rennes
35000 Rennes
France

Phone: +336 42 42 42 42
Fax: +336 42 42 42 42
EMail: simon.racaud@epitech.eu