Documentation: Gestionnaire UDP pour le projet R-Type

Le gestionnaire UDP est une composante clé du projet R-Type, chargé de faciliter une communication rapide et efficace entre le serveur et les clients. Ce document fournit des détails sur sa conception, y compris la structure des messages, le processus de sérialisation/désérialisation et les considérations liées à la bande passante.

## 1. Structure des messages UDP

Le gestionnaire UDP utilise un format de message bien défini pour la communication. Chaque message est représenté par la structure suivante :

```
unsigned int id : 10;    /* 10 bits pour 'id' (jusqu'à 1024 id) */
unsigned int action : 6; /* 6 bits pour 'action' (jusqu'à 64 types) */
std::string params;
std::string secret_key; /* Clé secrète pour l'authentification */
```

### Description des champs

- **id** : Correspond à l'identifiant unique d'une entité sur le serveur (index ECS). Prend en charge jusqu'à 1024 IDs uniques.
- **action** : Spécifie l'action à effectuer, limitée à 64 types (liée à l'Enum des événements).
- **params** : Contient des paramètres supplémentaires sous forme de chaîne de caractères.
- **secret_key** : Une chaîne de caractères utilisée pour authentifier le message entre le client et le serveur.

## 2. Sérialisation et Désérialisation

Pour envoyer et recevoir des messages efficacement via UDP, le gestionnaire convertit les messages en un format binaire. Cette fonctionnalité est implémentée dans la classe `MessageCompressor`.

## Implémentation : Classe `MessageCompressor`

### Méthode `serialize`

```cpp
void MessageCompressor::serialize(const Message &msg, std::vector<char> &buffer)
{
    buffer.clear();

    unsigned short header = (msg.id & 0x3FF) | ((msg.action & 0x3F) << 10);
    buffer.push_back(header & 0xFF);
    buffer.push_back((header >> 8) & 0xFF);

    uint32_t paramsSize = msg.params.size();
    buffer.insert(buffer.end(), reinterpret_cast<const char *>(&paramsSize),
                  reinterpret_cast<const char *>(&paramsSize) + sizeof(paramsSize));
    buffer.insert(buffer.end(), msg.params.begin(), msg.params.end());

    uint32_t secretKeySize = msg.secret_key.size();
    buffer.insert(buffer.end(), reinterpret_cast<const char *>(&secretKeySize),
                  reinterpret_cast<const char *>(&secretKeySize) + sizeof(secretKeySize));
    buffer.insert(buffer.end(), msg.secret_key.begin(), msg.secret_key.end());
}
```

### Méthode de`serialize`

```cpp
void MessageCompressor::deserialize(const std::vector<char> &buffer, Message &msg)
{
    size_t offset = 0;

    unsigned short header = buffer[offset] | (buffer[offset + 1] << 8);
    msg.id = header & 0x3FF;
    msg.action = (header >> 10) & 0x3F;
    offset += 2;

    uint32_t paramsSize;
    std::memcpy(&paramsSize, &buffer[offset], sizeof(paramsSize));
    offset += sizeof(paramsSize);
    msg.params = std::string(buffer.begin() + offset, buffer.begin() + offset + paramsSize);
    offset += paramsSize;

    uint32_t secretKeySize;
    std::memcpy(&secretKeySize, &buffer[offset], sizeof(secretKeySize));
    offset += sizeof(secretKeySize);
    msg.secret_key = std::string(buffer.begin() + offset, buffer.begin() + offset + secretKeySize);
}
```

Processus détaillé

1. **Sérialisation** :
    a. Combine `id` et `action` en un en-tête unique de 16 bits.
    b. Ajoute la taille et le contenu de `params`.
    c. Ajoute la taille et le contenu de `secret_key`.

2. **Désérialisation** :
    a. Extrait l'en-tête de 16 bits pour décoder `id` et `action`.
    b. Récupère la taille et le contenu de `params`.
    c. Récupère la taille et le contenu de `secret_key`.

## 3. Calcul de la bande passante

La bande passante nécessaire pour la communication UDP dépend du nombre de messages envoyés par seconde et de la taille de chaque message.

Répartition de la taille des messages

- En-tête : 2 octets (16 bits pour `id` et `action`)
- Longueur de `params` : 4 octets
- Contenu de `params` : Variable (dépend du message)
- Longueur de `secret_key` : 4 octets
- Contenu de `secret_key` : Variable (dépend de la clé)

Exemple

Pour un message contenant :

- `params` : 100 octets
- `secret_key` : 32 octets

Taille totale = 2 octets (en-tête) + 4 octets (longueur `params`) + 100 octets (`params`) + 4 octets (longueur `secret_key`) + 32 octets (`secret_key`) = **142 octets**

Formule de bande passante

Bande passante (octets/s) = Taille du message × Messages par seconde

Exemple :

- Taille du message : 142 octets
- Taux de ticks du serveur : 40 TPS (Ticks Par Seconde)

- Messages par tick : 100

Bande passante = 142 × 40 × 100 = **568 000 octets/s = 568 Ko/s**

Cette documentation reflète la mise à jour de la structure des messages UDP pour inclure une clé secrète et intègre l'implémentation des méthodes de sérialisation et de désérialisation. Pour toute question ou clarification, référez-vous à l'implémentation ou contactez l'équipe de développement.