R-Type Protocol RFC

NOTES:

- time_t → unsigned long : portable ?
- Threads par ROOMS ou Threads par CLIENTS ? (cote serveur)
- Timestamp utile ou pas ?
- Alignement de la structure avec la donnée reserved.

Partie I : Communication client/serveur

- Liste des actions possibles du client hors game.

```
enum eClientActionTCP
{
        AUTHENTIFICATION = 0,           /* Le client s'authentifie. */
        SHOW_ROOM,                      /* Requête de la liste des rooms. */
        CREATE_ROOM,                    /* Création d'une room sur le serveur. */
        JOIN_ROOM,                      /* Rejoindre une room. */
        LEAVE_ROOM,                     /* Quitter une room. */
        LAUNCH_GAME                     /* Lancer une partie (depuis une room). */
        MAX_CLIENT_ACTION_TCP                   /* Toujours en dernier. */
};
```

- Liste des actions possibles du client ingame.

```
enum eClientActionUDP
{
        MOVEMENT = 100,                 /* Déplacement du joueur. */
        SHOOTING,                       /* Le joueur tire. */
        ASK_DESCRIBE_ENTITY,            /* Demande de description d'une entité. */
        ASK_PLAYER_SCORE,               /* Demande du score d'un joueur. */
        ASK_PLAYER_LIFE,                /* Demande de la vie d'un joueur. */
        MAX_CLIENT_ACTION_UDP                   /* Toujours en dernier. */
};
```

Partie II : Communication serveur/client

- Liste des actions possibles du serveur hors game.

```
enum eServerActionTCP
{
        AUTHENTIFICATION_OK = 200,      /* Succès de l'authentification. */
        AUTHENTIFICATION_KO,            /* Echec de l'authentification. */
        ROOM _DESCRIPTION,              /* Description de la room. */
        CREATE_ROOM_OK,                 /* Succès de la création de room. */
        CREATE_ROOM_KO,                 /* Echec de la création de la room. */
        JOIN_ROOM_OK,                   /* Succès de l'accès à la room. */
        JOIN_ROOM_KO,                   /* Echec de l'accès à la room. */
        LEAVE_ROOM_OK,                  /* Succès du leaving room. */
        LEAVE_ROOM_KO,                  /* Echec lors du leave room. */
        LAUNCH_GAME_OK,                         /* Succès du lancement du jeu. */
        LAUNCH_GAME_KO,                         /* Echec du lancement du jeu. */
        MAX_SERVER_ACTION_TCP           /* Toujours en dernier. */
};
```

- Liste des actions possibles du serveur ingame.

```
enum eServerActionUDP
{
        START_GAME = 300,                       /* Début de la partie. */
        SPAWN_ENTITY,                   /* Apparition d'une entité */
        DESTROY_ENTITY,                 /* Destruction d'une entité. */
        MOVE_ENTITY,                    /* Déplacement d'une entité. */
        LIFE_ENTITY,                    /* Quantité de vie d'une entité. */
        COLLISION,                      /* Collision entre 2 entité. */
        DESCRIPTION_ENTITY,             /* Description d'une entité. */
        ENTITY_SCORE,                   /* Score du joueur. */
        PLAYER_DISCONNECT,              /* Déconnexion d'un joueur. */
        END_GAME,                       /* Fin de la partie. */
        MAX_SERVER_ACTION_UDP           /* Toujours en dernier. */
};
```

Partie III : Utilisation

- Description et composition du *header* d'un paquet.

```
struct packetH eader
{
        uint32_t        magic;          /* Nombre magique, teste la validité du paquet. */
        uint32_t        checkSum;       /* Autre élément de test de validité du paquet. */
        time_t          timestamp;      /* Date d'envoi du paquet. */
        uint16_t        commandID;      /* Identifiant de la commande. */
        uint16_t        dataSize;       /* Taille de la donnée envoyée. */
        uint32_t        reserved;       /* Espace réservé. */
};
```

- Description et composition des différents paquets.
  (Voir document RFC).