**Protocol R-Type**

Header

|  |  |  |
| --- | --- | --- |
| Header | 1 octet | = 0x7 |
| Command Id | 1 octet | Voir enum |
| Packet Id | 2 octets | Identifiant du packet |
| Len | 2 octets | Longueur des données (sans le header) |
| Checksum header | 2 octets | CRC16 (initialisé à 0) |
| Checksum datas | 2 octets | CRC16 (initialisé à 0) |

Réception d’un paquet :

Chaque paquet commence par 0x7 et sa validité est assurée par les checksum.

Identifiant des paquets et réponses :

Initialisé à 0, et incrémenté de 1 à chaque paquet envoyé. Dans le cas d’une réponse à un paquet reçu, on utilise le même identifiant ce paquet.

Identifiants des commandes :

|  |  |
| --- | --- |
| Commandes TCP | CONNECT, DISCONNECT, CONNECTED, CREATE\_GAME, JOIN\_GAME, REF\_GAME, LIST\_GAME, LEAVE\_GAME, END\_GAME, STATUS |
| Commandes UDP | DEAD, MOVE, POP, PING, PONG |

Tout paquet dont l’id n’est pas reconnu est systématiquement ignoré.

Taille des paquets :

La taille maximale d’un paquet est 512 octets (une taille supérieure n’est pas possible avec les données envoyées.

Status des données reçues :

Valeur de retour de la fonction qui vérifie la validité d’un paquet.

*Enum eStatusReceived : VALID, INVALID, INCOMPLETE*

Identifiants des paquets status :

Valeur contenue dans les datas d’un paquet de type status.

*Enum eStatusId : OK, FAILED, USED, GAME\_FULL, COMPLETE, GAME\_OVER*

Description des paquets :

**CONNECT**

|  |  |  |
| --- | --- | --- |
| Login | 1 à 25 caractères | Login du joueur |

Client vers Serveur

Le Serveur renvoie un paquet CONNECTED.

**DISCONNECT**

|  |  |  |
| --- | --- | --- |
| Login | 1 à 25 caractères | Login du joueur |

Client vers Serveur

Le Serveur ne renvoie rien.

**CONNECTED**

|  |  |  |
| --- | --- | --- |
| Status | 4 octets | Etat de la demande de connexion  (OK, FAILED ou USED) |
| Id | 2 octets | Identifiant unique généré par le serveur pour le client |

Serveur vers Client

Le Client ne renvoie rien.

**CREATE\_GAME**

|  |  |  |
| --- | --- | --- |
| Name | 1 à 25 caractères | Nom de la partie |
| NullByte | 1 octet | = 0x0 |
| Map Id | 1 octet | Identifiant du fichier de description de la map |
| Nb Player | 1 octet | Nombre maximum de joueur acceptés dans la partie (maximum 4) |

Client vers Serveur

Le Serveur renvoie un paquet STATUS avec comme valeur OK, FAILED ou USED.

**JOIN\_GAME**

|  |  |  |
| --- | --- | --- |
| Name | 1 à 25 caractères | Nom de la partie que le joueur souhaite rejoindre |

Client vers Serveur

Le Serveur renvoie un paquet REF\_GAME avec comme valeur OK, FAILED ou GAME\_FULL et informe les autres joueurs de l’arrivée.

**REF\_GAME**

|  |  |  |
| --- | --- | --- |
| Port | 2 octet | Port associé à la connexion UDP de la partie |
| Name | 1 à 25 caractères | Nom de la partie que le joueur souhaite rejoindre |

Serveur vers Client

Le Serveur renvoie un paquet GAME\_REF avec comme valeur OK, FAILED ou GAME\_FULL et informe les autres joueurs de l’arrivée.

**LIST\_GAME**

|  |  |  |
| --- | --- | --- |
| Nb Players | 1 octet | Nombre de joueur actuellement dans la partie |
| Name | 1 à 25 caractères | Nom de la partie que le joueur peut rejoindre |

Serveur vers Client

Lorsqu’un client se connecte au serveur celui-ci lui renvoie un paquet LIST\_GAME par partie en cours sur le serveur.

**LEAVE\_GAME**

|  |  |  |
| --- | --- | --- |
| Name | 1 à 25 caractères | Nom du joueur qui a quitté la partie |

Client vers Serveur

Le Serveur renvoie un paquet STATUS avec comme valeur OK, FAILED ou USED et informe les autres joueurs.

**END\_GAME**

|  |  |  |
| --- | --- | --- |
| Game status | 4 octets | Résultat de la partie : COMPLETE ou GAME\_OVER |
| Game name | 1 à 25 caractères | Nom de la partie |

Serveur vers Clients

Pas de réponse, le paquet est envoyé à tous les clients connectés au serveur pour les informer de la destruction de la partie sur le serveur. Seuls les joueurs présents sur cette partie interprètent le statut.

**STATUS**

|  |  |  |
| --- | --- | --- |
| Status Id | 4 octets | Valeur du statut en réponse à une requête d’un client |

Envoyé en réponse à un autre type de paquet.

**DEAD**

|  |  |  |
| --- | --- | --- |
| Name | 1 à 25 caractères | Nom du joueur qui est mort |

Serveur vers Clients

Pas de réponse

**MOVE**

|  |  |  |
| --- | --- | --- |
| Id | 2 octets | Identifiant unique de l’objet qui bouge |
| Pos X | 2 octets | Position en X du joueur |
| Pos Y | 2 octets | Position en Y du joueur |

Client vers Serveur

Le serveur transmet le paquet aux autres clients

**POP**

|  |  |  |
| --- | --- | --- |
| Id | 2 octets | Identifiant unique de l’objet qui pop |
| Type | 1 octet | Identifiant pour le type visuel de monstre |
| Pos X | 2 octets | Position en X au moment du tir |
| Pos Y | 2 octets | Position en Y au moment du tir |
| Date | 4 octets | Temps entre la date courant et l’instant de pop du monstre |
| Move  (sujet à caution) | 4 octets | Enum sur la fonction de mouvement |

Serveur vers Client

Transmit 5 fois à chaque joueur dans un intervalle de temps précédant l’apparition d’un monstre

**PING, PONG**

|  |  |  |
| --- | --- | --- |
| Nothing | 0 |  |