Le serveur utilisé dans ce projet est un serveur fonctionnant avec le protocole UDP. Son rôle ici est de centraliser les messages des différents robots présent sur le circuit afin de déterminer un ordre de passage dans l’intersection, c’est-à-dire en donnant ou en ne donnant pas l’autorisation de franchir l’intersection.

Le serveur fonctionne en 3 parties majeures. La première partie concerne la réception et l’enregistrement des données envoyées par les robots. La seconde partie se chargera de construire la séquence de passage en fonction des requêtes reçues et en suivant la politique batch expliqué plus haut. Pour finir, le serveur envoi cette séquence de passage à l’ensemble des robots qui n’auront qu’à vérifier s’ils sont présents dans la liste. Si oui alors ils peuvent avancer, si non, ils attendent les prochaines instructions.

L’implémentation finale utilisée fonctionne sur un ordinateur, même si elle peut être portée sur un robot. Du point de vue du code elle fonctionne de la manière suivante :

On crée un listener qui va se déclencher à chaque réception d’un message sur le réseau via le protocole UDP. A partir de se listener, les données vont être converties de Bytes à String afin de pouvoir traiter les données plus facilement. Les données sont ensuite traitées comme suit :

Voici un exemple de trame de donnée provenant d’un robot :

« 0\n0\n255\n0 »

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | \n | 0 | \n | 255 | \n | 0 |
| ID du robot | Délimiteur | Position du robot | Délimiteur | Vitesse du robot | Délimiteur | Sens de rotation |

Une fois la trame de données reçue, elle est découpée suivant les délimiteurs afin de former un couple de 4 variables rassemblant ainsi, la position, la vitesse et le sens de rotation d’un robot définit par son ID. A cela, on ajoute le temps système du serveur afin de gérer la dimension temporelle des demandes. On prend ici le temps serveur, car c’est une base commune à tous les robots.

Une fois tout ceci effectué, on passe à la partie mise à jour de la séquence de requête. En effet, chaque robot effectuant une requête ne se retrouve pas directement dans la séquence de passage, c’est pourquoi il faut une séquence « tampon » afin de conserver les demandes. Il se présente alors 2 cas :

* Le robot est déjà présent dans la liste
  + On met à jour sa position, sa vitesse ainsi que son sens de rotation
* Le robot n’a pas été trouvé dans la liste
  + On l’ajoute à la liste

Quand la séquence de requête est à jour, on passe à la création de la séquence de passage. Celle-ci est créée conformément à la politique batch. C’est-à-dire :

* Si la séquence de requête ne comporte qu’un robot et que la séquence de passage est vide
  + Ajouter le robot dans la séquence de passage
* Sinon on parcourt la séquence de requête
* Si un robot de la séquence de requête est présent dans la séquence de passage
  + Si le robot a quitté l’intersection
    - On le retire de la séquence de requête et de la séquence de passage
  + Sinon
    - On met à jour sa position, sa vitesse ainsi que son sens de rotation
* Sinon, si le robot n’a pas été trouvé dans la séquence de passage
  + Si le dernier robot présent dans la séquence de passage à le même sens de rotation que le robot actuel
    - Ajouter le robot à la séquence de passage
  + Sinon, si le délai entre la requête du dernier robot et du robot actuel excède un certain seuil (deltaTime)
    - Ajouter le robot à la séquence de passage

Pour finir, le serveur se charge de créer une string représentant l’ensemble de la séquence de passage en y ajoutant un identifiant en début (« server ») et fin (« end ») de chaîne afin de permettre aux robots de différencier les messages provenant du serveur des messages provenant des autres robots.

La séquence de passage est donc composée de ces deux identifiants, ainsi que des ID, position et vitesses des robots autorisés à franchir l’intersection. Une fois créée, cette séquence ressemble à celle-ci :

« server\n0\n0\n255\n[…]\nend »

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| server | \n | 0 | \n | 0 | \n | 255 | \n | … | \n | end |
| Début | Délimiteur | ID | Délimiteur | Pos. | Délimiteur | Vitesse | Délimiteur |  | Délimiteur | Fin |