Projet : Programmation parallèle et distribuée

A Agents Mobiles

Le but de ce projet est de faire jouer des agents mobiles au football. Un agent mobile est une sorte de robot informatique. Notre robot va pouvoir se déplacer dans différentes machines pour chercher le ballon et le mettre dans les buts de l'adversaire. Le ballon sera un objet `Ballon`. Il ne devra exister qu'une fois sur tout le système.

A.1 Terrain

Dans cette partie nous allons décrire le terrain. Le terrain sera composé de plusieurs serveurs contenant une parcelle. Chaque parcelle est reliée aux parcelles voisines où un joueur peut se déplacer en utilisant la fonction `goto`. Un joueur peut envoyer des messages aux autres joueurs. Lorsqu'un joueur entre dans une parcelle avec la fonction `incoming`, la parcelle crée un thread exécutant la fonction `void run()` du joueur. Le terrain peut savoir à tout instant qui a le ballon et lui reprendre à tout instant. Un joueur peut tenter de prendre le ballon avec la méthode `getBallon` ; Il est informé de sa réussite lorsqu'il n'a pas `null` comme réponse.

Il existe deux sortes de parcelles : les parcelles du milieu du terrain et les buts. Un but va compter le nombre de fois ou un joueur a tiré dans les cages et a réussi. Pour réussir un but le joueur doit tirer dans la bonne direction et avoir un peu de chance.

Un joueur ne peut pas sortir du terrain. En cas de tentative la fonction `goto` renvoie false.

Touches : Un ballon sortant du terrain va être remis à l'équipe adverse dans la même parcelle. Un message `Touche` sera envoyé à tout le monde.

Afin de rendre croustillant les matchs, toutes actions (tir, prise de ballon, but, ...) ont des probabilités de réussite dépendant du nombre de personnes de chaque équipe sur la parcelle. Pour la parcelle but, le gardien de but apporte plus de chance d'éviter un but qu'un autre joueur.

L'interface de `Terrain` sera visible de tout le monde et `TerrainImplInterface` ne sera visible que des joueurs dans la parcelle de terrain.

```
interface Terrain extends Remote{
  public void incoming(Agent agent); /* Agent : Ballon ou Joueur*/
}

Interface TerrainImplInterface {
  public static final int NORTH=0;
  public static final int SOUTH=1;
  public static final int EAST=2;
  public static final int WEST=3;

  public int[] getIdTerrain();/* donne la position x,y */
  public void sendMessage(Message mess);
  public boolean goto(int direction, Agent agent);
  public Ballon getBallon();
  public boolean lauch(Ball ball, int direction,int team,int vel);
}
```

A.2 Ballon

L'objet ballon possède une vélocité donnée par le joueur qui le tire et diminue en fonction du nombre de joueurs sur la parcelle. Si le ballon n'est pas récupéré par un joueur, il continuera sa course jusqu'à avoir une vélocité nulle. Chaque fois qu'un ballon traverse une parcelle contenant des joueurs, le ballon a une probabilité non nulle de se faire prendre par un des joueurs.

A.3 Joueur

Le joueur possède au moins deux fonctions : `Ballon getBallon()` et `boolean haveBallon()`. La seconde renvoie `True` si le joueur a le ballon (on suppose qu'il ne peut pas mentir)

Les joueurs peuvent utiliser les messages pour se coordonner.

A.4 Match

A.4.1 Début de match

- Tous les joueurs sont placés sur une parcelle centrale contenant également le ballon.
- Les joueurs vont se placer en deux équipes de façon équitable (autant de joueurs dans chaque équipe)
- Chaque équipe organise une élection du gardien de but (Vous pouvez également décider d'attribuer d'autres rôles plus précis aux joueurs)
- Une fois que tout est prêt, le match commence ! Un message est envoyé à tout le monde signifiant que le match à commencé.`getBallon()` pour tout le monde.

A.4.2 Déroulement du Match

Pendant le match chaque joueur se déplace et essaye d'avoir le ballon. Une fois celui-ci acquis, le joueur se déplace en direction des buts afin de marquer. Chaque fois qu'une parcelle voit un joueur avec le ballon entrer, il le signale à tout le monde en envoyant un message.

ATTENTION : Il ne devra pas y avoir de duplication de Ballon ou de Joueurs.

A.4.3 Fin de match

A partir d'un temps donné ou par une méthode manuelle, un signal signalant la fin du match est envoyé. Les joueurs doivent alors sortir de leur fonction `run()` et les scores sont envoyés par message.

B Travail à effectuer

B.1 Prétravail

Pour améliorer le confort de test, il est conseillé de faire un script de déploiement. Ce script va se connecter sur le nombre d'ordinateurs désirés, va lancer `rmiregistry` (à moins que votre programme java le lance lui-même) et lancer les serveurs avec les bonnes adresses.

Pour cela il est conseillé d'utiliser `ssh` avec certificat rsa.

```
$ ssh-keygen -t rsa
$ cat .ssh/id_rsa.pub >> .ssh/authorized_keys
```

La première ligne génère une clef rsa pour ssh. La deuxième autorise les possesseurs de la clef privée à accéder. On fait cela sur la même machine, car tous les ordinateurs partagent votre $HOME$ avec NFS ce qui vous permet d'accéder à vos documents et *.ssh* sur tout les postes.

B.2 Modélisation du Terrain

B.1 Une fois votre script prêt, Modélisez une première version des parcelles.

B.2 Adaptez votre script pour connecter les parcelles entre elles.

B.3 Implémentez la façon dont un joueur entre dans un terrain (création de thread).

B.4 Faites une gestion de tir et de sortie de ballon.

B.5 Implémentez les messages de début et de fin du match.

B.3 Stratégie du joueur

B.6 Implémentez le choix d'équipe.

B.7 Mettez en place l'élection du goal.

B.8 Ecrivez une stratégie.

B.4 Version finale

Le rendu final devra comporter :
- Un script de compilation ;
- Un script de déploiement ;
- Un README (comment compiler, déployer et lancer) ;
- Un rapport plus détaillé que le fichier README (divers choix dans la modélisation) en PDF ;
- Les sources.