Systèmes distribués

Acteurs

1/

L'algorithme de Chang et Roberts consiste à élire parmi n candidats le candidat qui possède le numéro le plus grand, le candidat de départ envoie son numéro au prochain candidat et si celui – ci a un numéro moins grand que l'envoyeur alors il récupère le numéro de l'envoyeur et l'envoie au prochain candidat, lorsqu'un candidat reçois le même numéro alors un tour complet a été fait et il est élu Roi.

On a donc qu'un seul type d'acteur qui est le candidat, le candidat possède donc un ID unique qui lui sera attribué à sa création via un message "RecupId" avec en paramètre chacun de leurs ID dans l'ordre à la réception du message chaque acteur range son ID dans une variable private.

De la même manière que le message "RecupId" on crée un message "Suivant" avec lequel on envoie l'actorSlection du candidat voisin, qui sera aussi stocké dans une variable de l'acteur pour pouvoir envoyer un message à son voisin par la suite.

Un message "Debut" vas donc être envoyé à l'acteur de départ qui se trouve avoir le premier ID dans la liste d'ID uniques, une fois le message reçu, l'acteur envoie son ID à l'acteur suivant grâce à un message "msgElection", quand un acteur reçoit ce message il va vérifier si l'ID reçu est supérieur ou bien inférieur, ou bien égal, dans le cas où l'ID reçu est supérieur alors il est envoyé, si il est inférieur alors on envoie l'ID de l'acteur et si il est égal alors on considère qu'on a fait un tour complet, l'envoie des ID se font toujours pas le message "msgElection".

2/

Le code du programme se divise en deux fichiers, le fichier App.java qui permet la configuration du système d'acteurs, et le fichier ChangRoberts.java qui regroupe les fonctionnalités de l'acteur candidat.

App.java :

```java
public class App {
 /**
 * @param args
 */
Run | Debug
public static void main(String[] args) {

 int n = 10;
 int ID[] = new int[n];

ActorSystem actorSystem = ActorSystem.create();

    for (int i=0;i<n;i++){

        int id = (int)(Math.random() * 100) +1;

            for (int j=0 ; j<i ;j++){
                if (id == ID[j]){
                    id = (int)(Math.random() * 100) +1;
                    j=0;
                }
            }

        ID[i]=id;
        ActorRef acteur = actorSystem.actorOf(ChangRoberts.props(), "candidat"+i);
        acteur.tell(new ChangRoberts.RecupId(id), ActorRef.noSender());
      }

    System.out.println(Arrays.toString(ID));
```

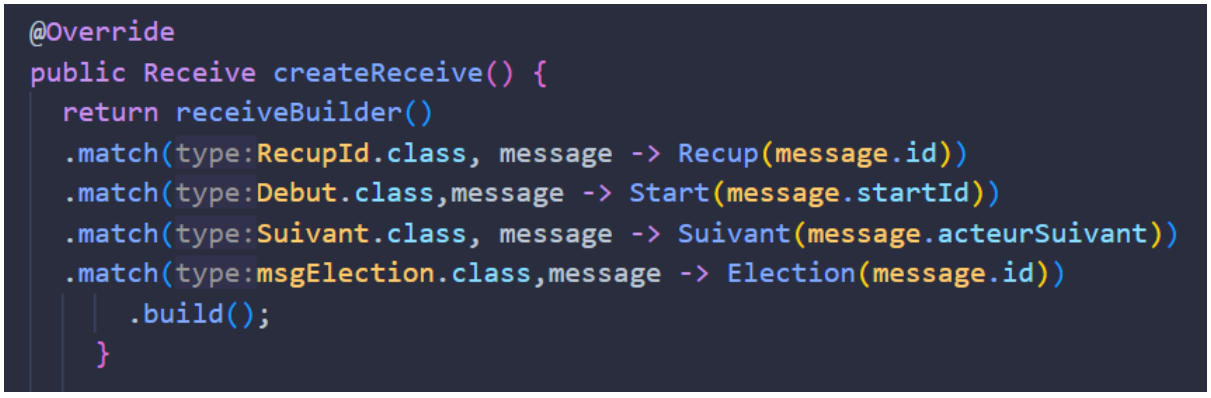
Deux variables sont initialisées, “n” qui définit le nombre d’acteur et un tableau “ID[]” qui va contenir les ID des n acteurs, on crée donc des ID aléatoires entre 1 et 100 qu’on stock dans le tableau en vérifiant bien qu’on ait pas de doublons.

on va ensuite créer les n acteurs en leurs attribuant chacun leurs noms attribués de 0 a 9, et à l’aide d’un “tell” ils récupéreront leurs id envoyé en paramètres.

Une boucle est créée afin de stocker dans un ActorSelection le nom de candidat qui va recevoir le tell “Suivant” avec comme paramètre le nom de l’acteur voisin de celui-ci

On finit par envoyer un tell “Debut” à l’acteur numéro 0 soit notre premier acteur afin de commencer l’élection avec comme paramètre son présumé ID unique qui sert de vérification.

ChangRoberts.java :



```
    private void Recup (final int id) {
      this.id = id;
    }

    private void Start(int startId){
      if (startId == this.id) {
      System.out.println("Acteur " + id + " démarre l'élection");
      acteurSuivant.tell(new msgElection(id), getSelf());
      }
    }

    private void Suivant(ActorSelection acteurSuivant) {
      this.acteurSuivant = acteurSuivant;
  }

  private void Election(int autreId) {
    System.out.println("je suis l'acteur "+id+" et je reçois l'id "+ autreId);
    if (autreId > id) {
     acteurSuivant.tell(new msgElection(autreId), getSelf());
    } else if (autreId <id) {
      acteurSuivant.tell(new msgElection(id), getSelf());
    }
    else if (autreId == id) {
        System.out.println("Roi :" + id);
    }
}
```

La méthode Recup permet à l'acteur de récupérer dans son ID donné en paramètre du message qui déclanche la méthode.

La méthode Start se lance après réception du message "Debut" par le programme initial, elle initie le lancement de l'algorithme en passant par une vérification de l'ID du premier acteur, si la vérification est passé l'acteur envoie son id à l'acteur suivant.

La méthode suivant permet à chaque acteur de stocker l'ActorSelector (le nom de l'acteur) voisin afin de lui envoyer un message.

La méthode Election se lance après réception du message "msgElection" envoyé par l'acteur voisin, s'en suit une comparaison qui résulte à l'envoie le plus grand ID des deux acteurs et une condition de fin de boucle de l'algorithme au cas ou l'id reçu est égal à l'id de l'acteur receveur, cela veut dire que les acteurs on fait un tour complet car l'ID est unique.

Résultats :

```
[46, 45, 25, 40, 90, 18, 10, 66, 70, 28, 94, 77]
Acteur 46 demarre l'election
je suis l'acteur 45 et je recois l'id 46
je suis l'acteur 25 et je recois l'id 46
je suis l'acteur 40 et je recois l'id 46
je suis l'acteur 90 et je recois l'id 46
je suis l'acteur 18 et je recois l'id 90
je suis l'acteur 10 et je recois l'id 90
je suis l'acteur 66 et je recois l'id 90
je suis l'acteur 70 et je recois l'id 90
je suis l'acteur 28 et je recois l'id 90
je suis l'acteur 94 et je recois l'id 90
je suis l'acteur 77 et je recois l'id 94
je suis l'acteur 46 et je recois l'id 94
je suis l'acteur 45 et je recois l'id 94
je suis l'acteur 25 et je recois l'id 94
je suis l'acteur 40 et je recois l'id 94
je suis l'acteur 90 et je recois l'id 94
je suis l'acteur 18 et je recois l'id 94
je suis l'acteur 10 et je recois l'id 94
je suis l'acteur 66 et je recois l'id 94
je suis l'acteur 70 et je recois l'id 94
je suis l'acteur 28 et je recois l'id 94
je suis l'acteur 94 et je recois l'id 94
Roi :94
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.956 s
```

On s'aperçois que l'acteur ayant un id plus petit renvoie l'id reçu comme les 3 premiers acteurs receveurs et on peut voir que l'acteur avec ID 90 qui est supérieur à 46 envoie son id à l'acteur 18 juste après, un second changement avec l'id 94.

D’autres exemples :

```
[99, 75, 17, 49, 98, 12, 52, 94, 10, 90, 68, 85]
Acteur 99 demarre l'election
je suis l'acteur 75 et je recois l'id 99
je suis l'acteur 17 et je recois l'id 99
je suis l'acteur 49 et je recois l'id 99
je suis l'acteur 98 et je recois l'id 99
je suis l'acteur 12 et je recois l'id 99
je suis l'acteur 52 et je recois l'id 99
je suis l'acteur 94 et je recois l'id 99
je suis l'acteur 10 et je recois l'id 99
je suis l'acteur 90 et je recois l'id 99
je suis l'acteur 68 et je recois l'id 99
je suis l'acteur 85 et je recois l'id 99
je suis l'acteur 99 et je recois l'id 99
Roi :99
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

```
[92, 20, 70, 78, 88, 93, 62, 83, 27, 100, 40, 11]
Acteur 92 demarre l'election
je suis l'acteur 20 et je recois l'id 92
je suis l'acteur 70 et je recois l'id 92
je suis l'acteur 78 et je recois l'id 92
je suis l'acteur 88 et je recois l'id 92
je suis l'acteur 93 et je recois l'id 92
je suis l'acteur 62 et je recois l'id 93
je suis l'acteur 83 et je recois l'id 93
je suis l'acteur 27 et je recois l'id 93
je suis l'acteur 100 et je recois l'id 93
je suis l'acteur 40 et je recois l'id 100
je suis l'acteur 11 et je recois l'id 100
je suis l'acteur 92 et je recois l'id 100
je suis l'acteur 20 et je recois l'id 100
je suis l'acteur 70 et je recois l'id 100
je suis l'acteur 78 et je recois l'id 100
je suis l'acteur 88 et je recois l'id 100
je suis l'acteur 93 et je recois l'id 100
je suis l'acteur 62 et je recois l'id 100
je suis l'acteur 83 et je recois l'id 100
je suis l'acteur 27 et je recois l'id 100
je suis l'acteur 100 et je recois l'id 100
Roi :100
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.030 s
```

Remarques personnelles :

Je n'ai pas intégré de tolérance à la panne, si un acteur est hors service alors le programme tourne en boucle, l'idée pour y remédier serait qu'au début de l'élection, le premier acteur envoie un message supplémentaire en parallèle du message "msgElection" (l'envoie d'ID au voisin) au programme principale afin que le programme principale sache à qui est-ce le tour d'envoyer son id, le tout gérer par un système de timeout, si le programme principale ne reçois pas de message toute les dis secondes, il vas modifier la liste pour retirer l'id de l'actor en panne et change le nom du voisin dans l'acteur envoyeur à l'acteur en panne

Ex : acteurs A,B,C

A->B->C

(A)
NomVoisin = "B";

(B)
NomVoisin = "C";

(C)
NomVoisin = "X";

* Modification du nom voisin par le programme principale *

(A)
NomVoisin = "C";

* Debut (C)

Ainsi l'acteur en panne sort de l'élection et l'élection se poursuit en envoyant un message Debut à l'acteur C pour reprendre l'élection.

SOURCES :
Utilisation du actorSelection :
https://stackoverflow.com/questions/17456302/how-to-find-specific-actor-in-akka