Université Abdelhamid Mehri Constantine 2- Algérie
Faculté des Nouvelles Technologies de l'Information et de la Communication
Département d'Informatique Fondamentale et ses Applications

1ère Année Master Réseaux et Systèmes Distribués

TP ALGORITHMES DISTRIBUÉS (ALDI)

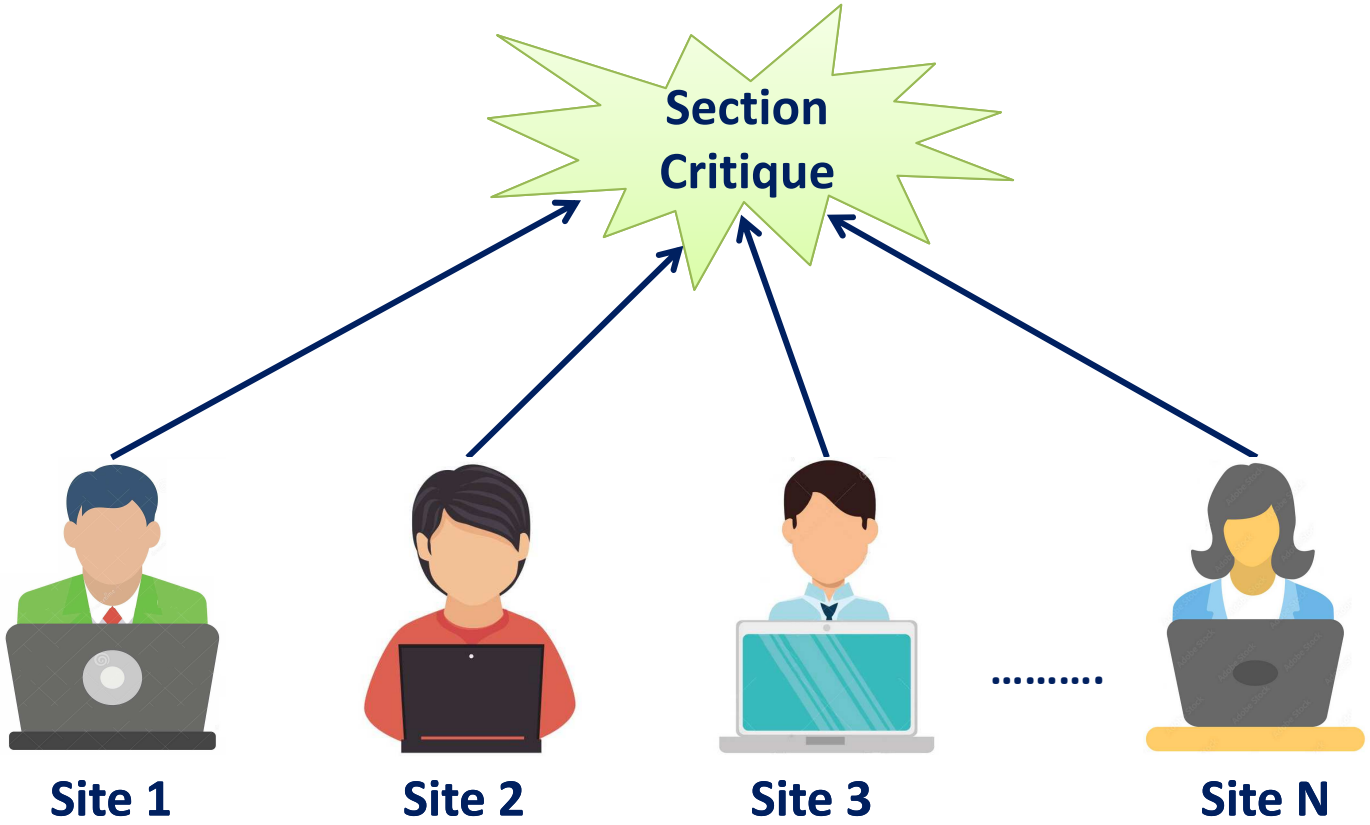
TP N°6 : Implémentation de l'algorithme de Ricart et Agrawala

Année universitaire : 2021/2022

Objectif du TP

Implémenter l'algorithme de Ricart et Agrawala en utilisant la plateforme Jade.

Rappel sur l'algorithme de Ricart et Agrawala



Rappel sur l'algorithme de Ricart et Agrawala

Variables locales pour un processus Pi :

- **Ri = {1, 2, …, N}-{i}** : ensemble contenant les identités des sites auxquels le site **'i'** doit demander l'autorisation d'accès à la section critique;
- **étati = {dehors, demandeur, dedans}** initialisé à **dehors**;
- **hi, lasti** : entier croissant initialisé **0**;
- **prioritéi** : booléen;
- **attendui, différéi** : ensemble de sites initialisé à **Ø**;

Rappel sur l'algorithme de Ricart et Agrawala

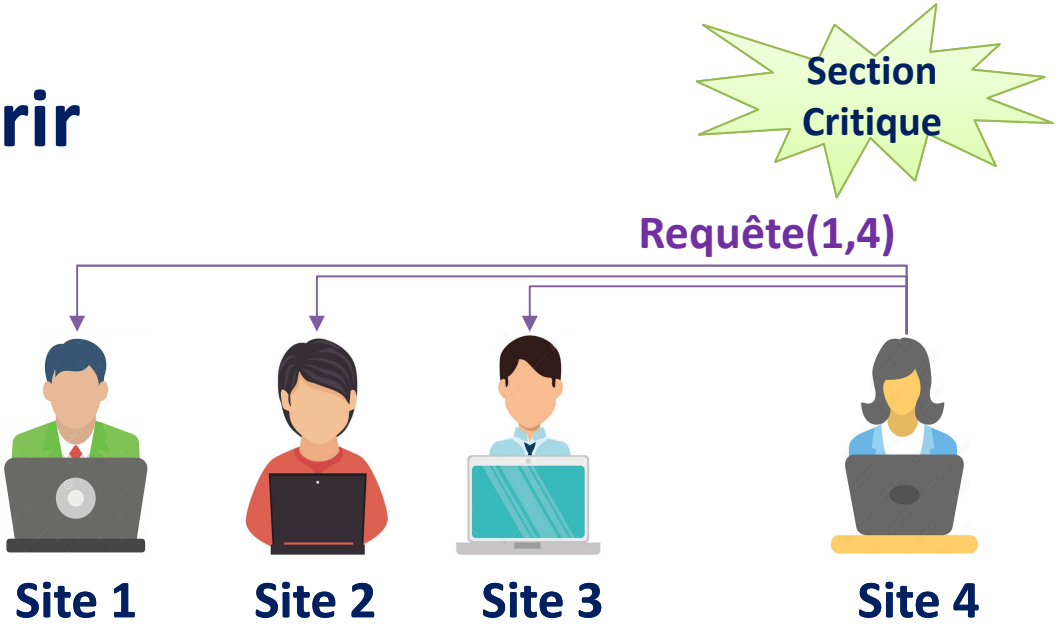
Lors d'un appel à acquérir

étati = demandeur;

hi = hi + 1;

lasti = hi;

attendui = Ri;



$\forall$ j $\in$ Ri : envoyer (requête(lasti, i)) à j;

Attendre (attendui == $\varnothing$);

étati = dedans;

Rappel sur l'algorithme de Ricart et Agrawala

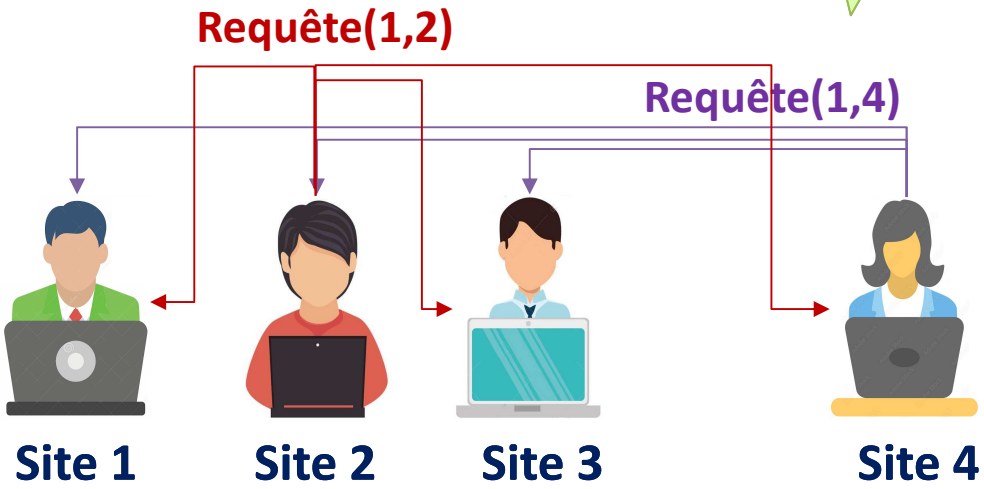
Lors d'un appel à acquérir

étati = demandeur;

hi = hi + 1;

lasti = hi;

attendui = Ri;



$\forall j \in Ri$: envoyer (requête(lasti, i)) à j;

Attendre (attendui == ∅);

étati = dedans;

Rappel sur l'algorithme de Ricart et Agrawala

Lors d'un appel à libérer

étati = dehors;

$\forall$ j $\in$ différéi : envoyer (permission(i)) à j;

différéi = $\emptyset$;

Rappel sur l'algorithme de Ricart et Agrawala

Lors de la réception de requête (k, j)

hi = max(hi, k);

prioritéi = (étati = dedans) ou ((étati = demandeur) et (lasti,i)<(k,j));

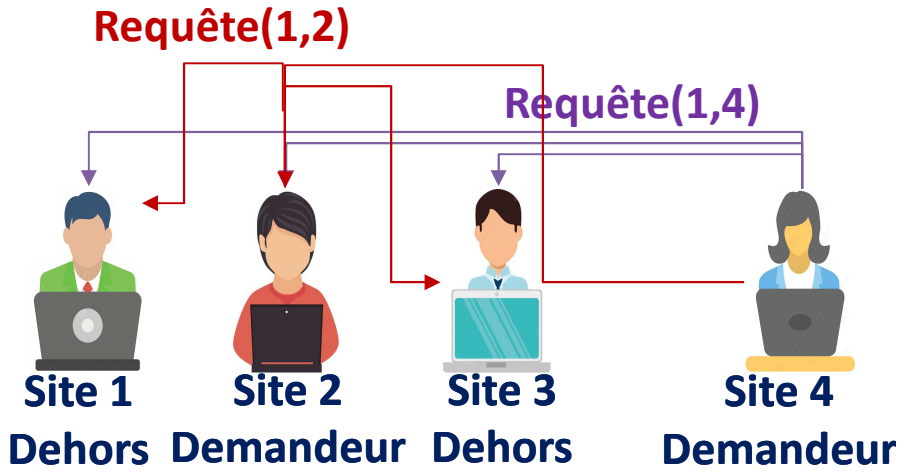
Si prioritéi **alors**

différéi = différéi ∪ {j};

Sinon

Envoyer (permission(i)) à j;

Fin Si



Rappel sur l'algorithme de Ricart et Agrawala

Lors de la réception de requête (k, j)

hi = max(hi, k);

prioritéi = (étati = dedans) ou ((étati = demandeur) et (lasti,i)<(k,j));

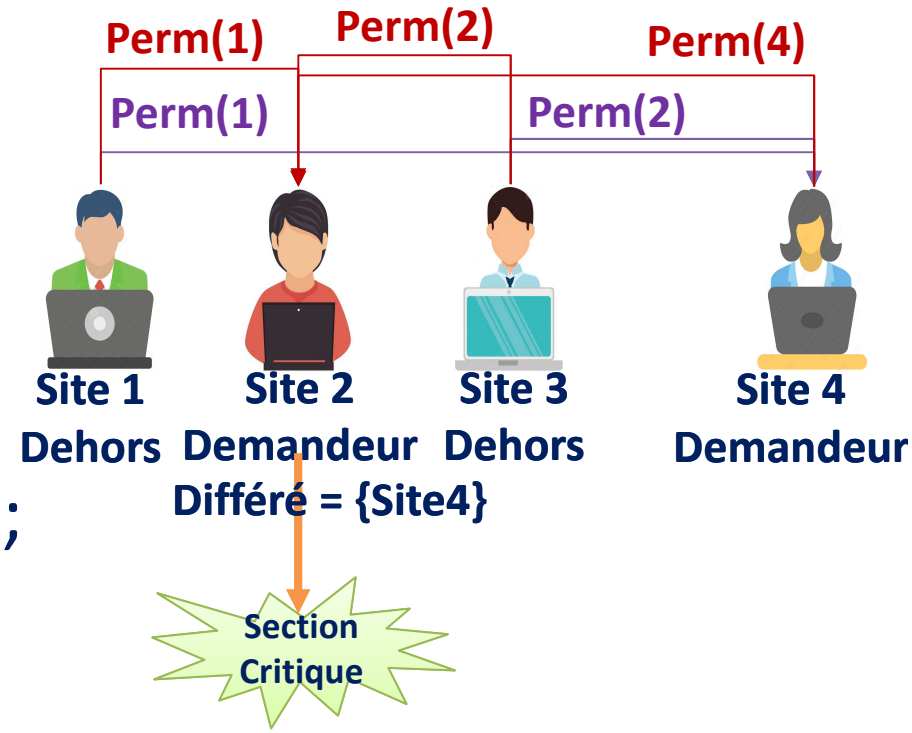
Si prioritéi **alors**

différéi = différéi ∪ {j};

Sinon

Envoyer (permission(i)) à j;

Fin Si



Rappel sur l'algorithme de Ricart et Agrawala

Lors de la réception de permission (j)

attendui = attendui - {j};

Démarche d'implémentation d'un algorithme distribué sous la plateforme JADE

1. Identifier les types d'agents qui seront utilisés dans l'algorithme distribué.
2. Identifier les comportements de chaque agent ainsi que leur type puis tracer un graphe qui représente le lien entre les différents comportements.
3. Identifier les différents messages échangés entre les agents et donner leur contenu.
4. Identifier les arguments qui seront passés à chaque agent.
5. Implémenter la classe de chaque agent (la méthode setup et les différents comportements).
6. Tester les classes implémentées.
7. Analyser le contenu de la console après exécution, s'il y a des anomalies alors il faut revoir l'étape 5.

Implémentation de l'algorithme

1. **Identifier les types d'agents qui seront utilisés dans l'algorithme distribué :**

   Dans cet algorithme, nous avons un seul type d'agents :

   - Un agent (site) qui désire accéder à une section critique.

Implémentation de l'algorithme

2. **Identifier les comportements de chaque agent ainsi que leur type puis tracer un graphe qui représente le lien entre les différents comportements :**

Comportement de l'Agent (site)	Type de Comportements
Dehors	GenericBehaviour
DevenirDemandeur	OneShotBehaviour
AttendreVecteurAttenduEnsembleVide	GenericBehaviour
EnSc	OneShotBehaviour
LibererSC	OneShotBehaviour

Implémentation de l'algorithme

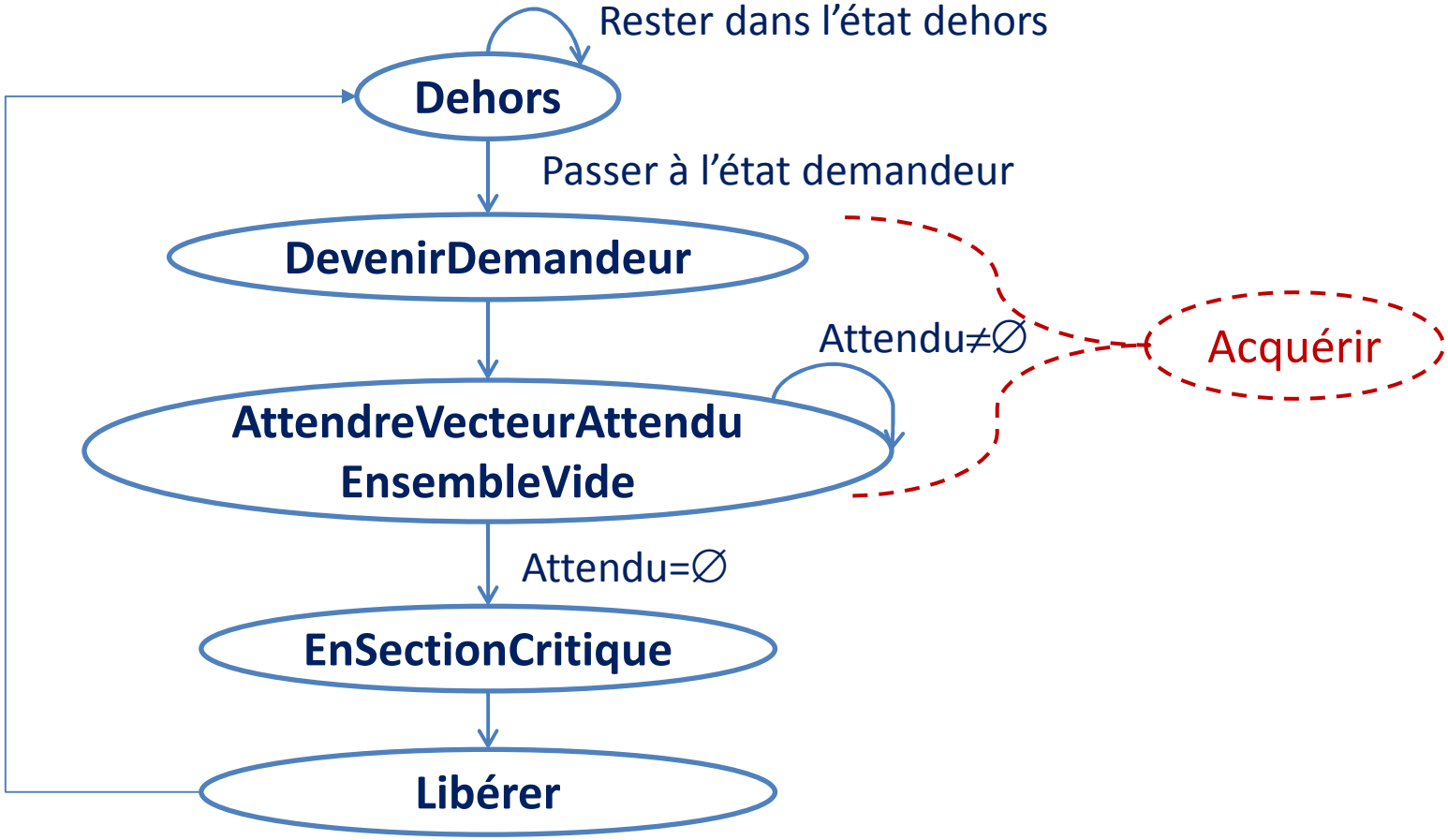
2. **Identifier les comportements de chaque agent ainsi que leur type puis tracer un graphe qui représente le lien entre les différents comportements :**

Comportement de l'Agent (site)	Type de Comportements
Dehors	GenericBehaviour
DevenirDemandeur	OneShotBehaviour
AttendreVecteurAttenduEnsembleVide	GenericBehaviour
EnSc	OneShotBehaviour
LibererSC	OneShotBehaviour

Remarque : la consultation de la boite aux lettres se fera dans une méthode qui sera appelée au niveau de chaque comportement

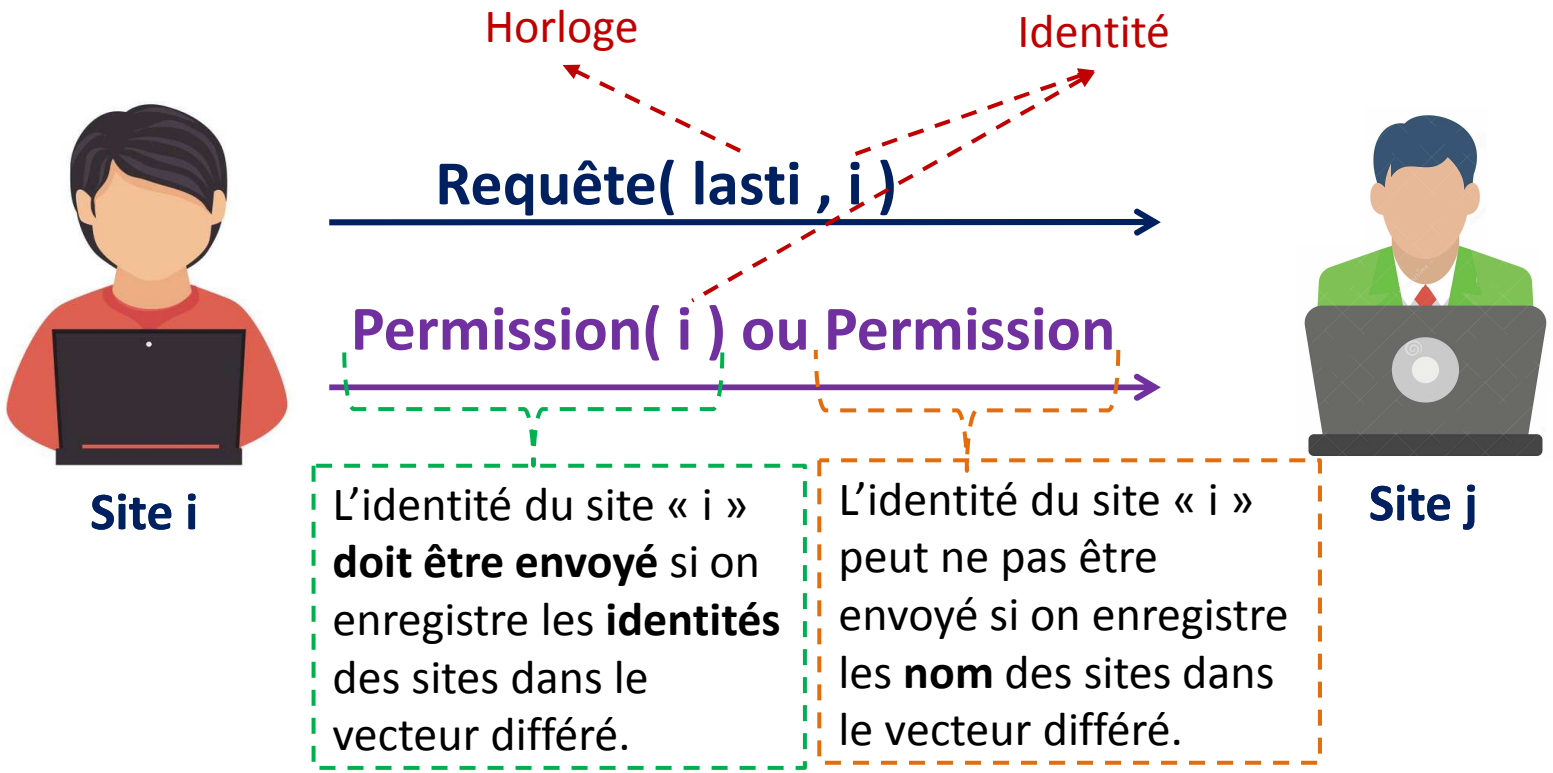
Implémentation de l'algorithme

2. Identifier les comportements de chaque agent ainsi que leur type puis tracer un graphe qui représente le lien entre les différents comportements :



Implémentation de l'algorithme

3. Identifier les différents messages échangés entre les agents et donner leur contenu :



Implémentation de l'algorithme

4. Identifier les arguments qui seront passés à chaque agents

- Chaque agent (site) doit connaitre :
  - ✓Son identité
  - ✓Les noms des sites à qui il vais envoyer la requête
  - ⇒La liste des arguments sera la forme suivante :

    (id, nomSite1, nomSite2, …, nomSiteN)

Exemple :

a:site(1,b,c)

b:site(2,a,c)

c:site(3,a,b)

Implémentation de l'algorithme

5. Implémenter la classe de chaque agent (la méthode setup et les différents comportements)

- Nous allons implémenter 1 seule classe :

  Classe Site

Implémentation de l'algorithme

5. Implémenter la classe de chaque agent

Algorithme exécuté par chaque site i :

Variables du site i :

Ri = {1, 2, …, N}-{i} :

étati = {dehors, demandeur, dedans} initialisé à **dehors**;

hi, lasti : entier croissant initialisé **0**;

prioritéi : booléen;

attendui, différéi : ensemble de sites initialisé à **Ø**;

```
public class site extends Agent{
  ArrayList  Ri = new ArrayList(),
             attendu = new ArrayList(),
             differe = new ArrayList();
  String etat;
  int h, last, id;
  boolean priorité;
```

Implémentation de l'algorithme

5. Implémenter la classe de chaque agent

Méthode setup()

```
public void setup(){
 System.out.println("Agent "+ this.getLocalName());
 Object [] args = this.getArguments();
 if (args != null){
  id = Integer.parseInt(args[0].toString());
  for( int i = 1; i < args.length; i++)
   Ri.add(args[i].toString());
   //System.out.println("Agent " + this.getLocalName()+ "
   id "+ id + " Ri "+ Ri.toString());
 }
 this.addBehaviour(new Dehors());
}//setup
```

Implémentation de l'algorithme

5. Implémenter la classe de chaque agent

Les comportements

```
public class Dehors extends Behaviour{
  public void action(){
    etat = "dehors";
    System.out.println("Agent "+getLocalName()+" état dehors");
    block((int) (Math.random() * 1000));
  }
  public boolean done(){
    int val = (int) (Math.random() * 2);
    if (val == 0) return false; //je reste dans l'état dehors
    else {//(val == 1) //je vais devenir demandeur
      addBehaviour(new DevenirDemandeur());
      return true;
    }
  }
}
```

Implémentation de l'algorithme

5. Implémenter la classe de chaque agent

Les comportements

```
public class DevenirDemandeur extends OneShot
  public void action(){
    etat = "demandeur";
    h++;
    last = h;
    for( int i = 0; i < Ri.size(); i++){
      attendu.add(Ri.get(i));
      ACLMessage msgEnvoi  = new ACLMessage (ACLMessage.INFORM);
      msgEnvoi.addReceiver(new AID(Ri.get(i).toString(),AID.ISLOCALNAME));
      msgEnvoi.setContent("requete"+"$"+last+"$"+id);
      send(msgEnvoi);
    }
    addBehaviour(new AttendreVecteurAttenduDevientVide());
  }
}
```

Lors d'un appel à acquérir
étati = demandeur;
hi = hi + 1;
lasti = hi;
attendui = Ri;
∀ j ∈ Ri : envoyer (requête(lasti, i)) à j;
Attendre (attendui == ∅);
étati = dedans;

Implémentation de l'algorithme

6. Tester les classes implémentées (faire une exécution avec 3 sites).
7. Analyser le contenu de la console après exécution, s'il y a des anomalies alors il faut revoir l'étape 5.