Section critique dans un système distribué

Modèle pour les systèmes distribués

Composantes :

- Noeud : un ordinateur
- Processus : exécutable qui tourne sur un noeud.
- Liens : canaux de communication entre noeuds.

Hypothèses :

- Il n'existe pas de temps absolu global.
- Les processus d'un noeud se synchronisent localement par les primitives déjà vues.
- Les noeuds ne tombent pas en panne.
- Chaque noeud est connecté à un autre noeud par un canal bi-directionnel.
- Les canaux délivrent les messages sans erreur, mais pas forcement dans l'ordre.
- Le temps de transit des messages est fini mais arbitraire.
- Chaque noeud dispose d'un identificateur unique, `myID`.

Echange de messages :

- Envoi : précise le type du message, son destinataire et les paramètres.
- Réception : precise le type du message et les paramètres. Implicitement, on reçoit que les messages qui nous sont destinés.

```
ch ! MessageType, Destination, Parameters
ch ? eval(MessageType), eval(myID), Parameters
```

Concurrence entre les processus d'un noeud :

- La reception d'un message par un processus est atomique avec le traitement du message.
- Le pre- et post-protocoles pour la section critique sont aussi atomiques.

Pour assurer ces hypothèses, il faut eventuellement utiliser les mécanismes de synchronisation centralisée.

Section critique distribué : algorithme Ricart-Agrawala

Idée : basé sur des numéros d'ordre comme dans l'algorithme du boulanger, le noeud ayant le numéro le plus petit entre dans la section critique.

Problèmes à resoudre :

- Le choix des numéro est (presque) aléatoire.
- La comparaison entre les numéros est faite après l'envoi du numéro par des messages.

Première version (incomplète) :

```
/* Algorithme éeffectu par chaque noeud */
int myNum = 0; /* variable épartage */
set_of_int deferred = empty_set;
active proctype Main () {
  do :: true->
    /* non-critical section */
    myNum = /* choisit un nombre */;
    /* pour tous les autres noeuds N */
      ch ! request, N, myID, myNum
    /* attend le reponse de tous les autres noeuds */
    /* critical section */
    /* pour atous les noeuds deferes N */
      effacer N de deferred
      ch ! reply, N, myID
  od
}
active proctype Receive () {
  int source, recNum;
  do :: true->
    ch ? request, eval(myID), source, recNum
    if :: recNum < myNum -> ch ! reply, source, myID
       :: else -> /* ajouter source a deferred */
```



```
    fi
  od
}
```