University of M'sila, Mathematical and Computer Science Faculty CS department
Module: Distributed Systems Year: 2025-2026

Master 1 RTIC

TP3: *Synchronization*

Exercice 1

Nous avons vu en cours l'algorithme centralisé d'exclusion mutuelle.

1. Proposer une solution permettant de garantir que les requêtes d'entrée en section critique sont traitées dans l'ordre de leur émission, et non dans l'ordre de leur réception.

2. Concevoir et implémenter, en Java, l'algorithme ainsi modifié.

Exercice 2

La description ci-dessous représente l'algorithme distribué d'exclusion mutuelle de **Leslie Lamport**.

- Structure de données sur chaque site : tableau de N messages, 1 par site.
- Initialement sur Si : pour tout j $Mij := (lib, 0, Sj)$
- Demande d'entrée d'un site Si en section critique : diffusion de $(req, H(req), i)$ à tous les sites, Si compris.
- Réception de $M = (req, H(req), j)$ ou $(lib, H(lib), Sj)$ => $Mij := M$
- Réception de $M = (acq, H(acq), Sj)$ => si $Mij \neq (req, H(req), j)$ alors $Mij := M$
- Si entre en section critique quand sa demande $Mii << Mij$ pour tout j

1. Vérifier que l'algorithme est correct ?

2. Concevoir, et Implémenter avec Java cet algorithme.

Exercice 3

Simuler l'interblocage :

1. Avec les threads.
2. Avec les sockets TCP de Java.

Note

Several java codes of distributed algorithms on web site of **Vijay K. Garg**, Professor at University of Texas at Austin, http://users.ece.utexas.edu/~garg/jbk.html