TP5 SG-2: Implementing group communication algorithms

Introduction à la commination de groupe avec Java

La communication multicast permet à un processus émetteur d'envoyer un message à plusieurs récepteurs simultanément, sans établir de connexions individuelles. Java offre cette fonctionnalité via la classe **MulticastSocket**, basée sur le protocole UDP**.**

- Chaque groupe multicast est identifié par une adresse IP spéciale de 224.0.0.0 à 239.255.255.255 (plage réservée au multicast IPv4).
- Un ou plusieurs processus rejoignent ce groupe pour recevoir les messages.
- L'émetteur envoie un paquet UDP à cette adresse — tous les membres du groupe le reçoivent.

Classe/ Fonction	**Rôle**
MulticastSocket	Création du **socket multicast**
InetAddress	Définition de l'adresse et du port du **groupe multicast**
joinGroup()	Le récepteur **rejoint le groupe**
send()	L'émetteur **envoie un message** au groupe
receive()	Le récepteur **reçoit le message**
leaveGroup()	Le récepteur **quitte le groupe**
close()	Les sockets sont **fermés proprement**

Exemples de code

1. Le code sur le line https://moodle.univ-msila.dz/moodle/mod/feedback/view.php?id=75057 est exemple Java complet et clair, intégrant toutes les étapes de la communication multicast : Création du groupe, adhésion, envoi, réception, et quittage du groupe, avec commentaires détaillés à chaque étape.
2. Le code sur le https://moodle.univ-msila.dz/moodle/mod/feedback/view.php?id=48799 est un véritable Chat.

Travail demandé

- Utiliser le *code* Chat sur le lien https://moodle.univ-msila.dz/moodle/mod/feedback/view.php?id=48799 implémenter l'algorithme *TO-multicast(m, g)*.

Help

```
1:  Sender:
2:    procedure TO-multicast(m)
3:      send (m) to sequencer
4:    end

5:  Sequencer:
6:    Initialisation:
7:      seqnum ← 1
8:    when receive (m)
9:      sn(m) ← seqnum
10:     send (m, sn(m)) to all
11:     seqnum ← seqnum + 1
12:   end when

13: Destinations (code of process p_i):
14:   Initialisation:
15:     nextdeliver_{p_i} ← 1
16:     pending_{p_i} ← ∅
17:   when receive (m, seqnum)
18:     pending_{p_i} ← pending_{p_i} ∪ {(m, seqnum)}
19:     while ∃(m', seqnum') ∈ pending_{p_i} : seqnum' = nextdeliver_{p_i} do
20:       deliver (m')
21:       nextdeliver_{p_i} ← nextdeliver_{p_i} + 1
22:     end while
23:   end when
```