Système Distribué - Partie III

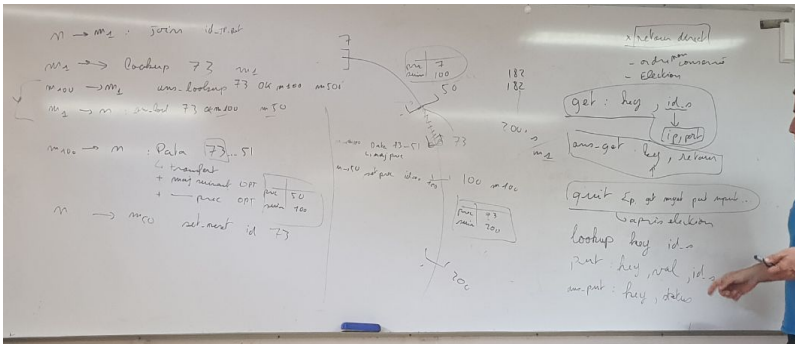
Enseignant: Georges Da-Costa <dacosta@irit.fr>

I. Protocole Chord

A) Misc.

Lien vers le git contenant la bibliotheque commune:
https://github.com/Victor333Huesca/libchord
Pour tout ajout, soit mp Victor -- avec son nom/courriel guiteub -- qui ajoutera les droits, soit fork et faire un pull-request.

B) Types de messages



get : key, id

- id = (ip, port, key)

Format message JSON: {'type':'get', 'key':key, 'id':'id'}

ans_get : key, retour

- retour : val

Format message JSON: {'type':'ans_get', 'key':key, 'value':value}
Avec value la valeur associé à la clé

lookup : key, id_s:

Format message JSON: {'type':'lookup', 'key':key, 'id_s':id_s}
id_s représente l'id de celui qui a fait la demande de lookup

ans_lookup : key, OK/NOK, mbb, mba

-> cf schéma plus bas
-> mbb et mba ne servent à rien si NOK -> pas de traitement

Format message JSON: {‘type’:’ans_lookup’, ‘value’:[OK,NOK], ‘mbb’:id, ‘mba’:id}

put: key, val, id_s
Format message JSON: {‘type’:’put’, ‘key’:key, ‘value’:value, ‘id_s’:id_s}

ans_put : key, status
Format message JSON: {‘type’:’ans_put’, ‘key’:key, ‘status’:[OK, NOK]}

Propriétés :

- retour direct des messages
- ordre non conservé des messages
- quit -> les sommes des stats de chaques machines (après élection)
- stats demandées :
    - nb_msg / get
    - nb_msg / put
    - nb_msg / insert_node
    - nb_msg / remove_node
- Elections

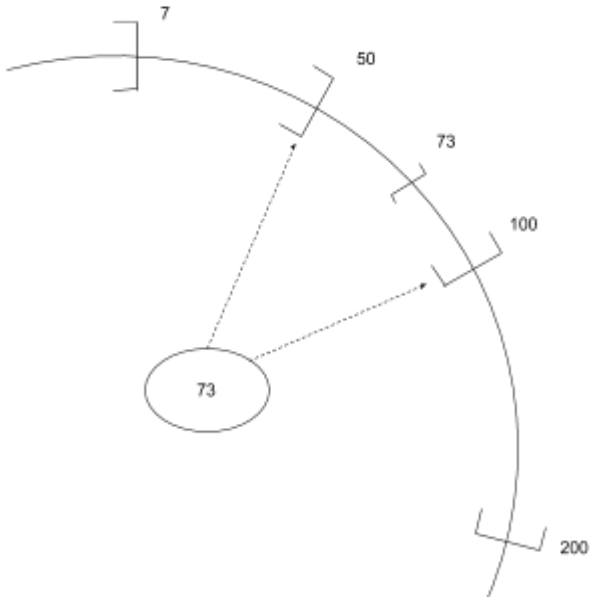
Insertion d’un noeud
m1 est le noeud qui a donner l’key, n est le nouveau noeud, mba et mbb sont les deux nouvelle borne du nouveau noeud
L’IP et le port de m1 sont censés être connus(

- n demande a m1 de lui attribuer une key , et il lui donne son IP et son port (Pour qu’il puisse le joindre après) [join ip port]
- m1 tire au hasard et vérifie que l’key soit disponible [lookup 73]
  m1 -> lookup key [ans_lookup 73 OK mbb mba]
- m1 envoie à n son key si la demande est ok + id(mba) et id(mbb) [ans_lookup 73 OK mbb mba]
- n demande à mbb de lui envoyer ses données (Sinon problème de synchronisation à cause de l’ordre non conservé des messages) [get_data 73]
- mbb envoie à n sa table de routage de n à mba [data n...mba] et se met à jour
- n envoie à mba qu’il est son suivant en précisant son id [set_next mba n]

join : ip, port
Format message JSON: {‘type’:’join’, ‘ip’:ip, ‘port’:port}

ans_join : key, key_mba, key_mbb
Format message JSON: {‘type’:’ans_join’, ‘key’:key, ‘mba’:key_mba, ‘mbb’:key_mbb}

Autres infos
Version 1 du TP : Le lookup va juste se baser sur le suivant() / le prédécesseur()
Version 2 du TP : Le lookup se baser sur la finger table



Ici n=73, mba=50, mbb=100

Les messages sont envoyés au voisin suivant (noeud supérieur % nb noeud)