Gestion du changement de topologie

Nous cherchons à avoir trois propriétés vraies ci possible à tout instant :

1. Il y a exactement un serveur élu par réseau comportant plus d'un serveur
2. Chaque serveur a connaissance de l'ensemble des clients connectés sur son réseau
3. Chaque serveur a connaissance de l'ensemble des autres serveurs présent sur son réseau

Ce trois propriétés sont mises à mal par les changements de topologie (ajout ou retrait d'un ou plusieurs serveurs).

Par exemple :

- Un ajout de serveur peut entrainer deux réseaux à être relié, et ainsi avoir deux serveurs élus.
- Un ajout de serveur peut entrainer l'ajout de nouveau client / serveurs sur notre réseau.
- Un retrait de serveur peut entrainer la coupure de notre réseau en 2 réseaux. Ainsi un de ces deux réseau se retrouvera sans serveur élu.
- Un retrait de serveur peut entrainer des clients serveurs à qui nous ne pouvons pas parler.
- Perte de messages de diffusion FIFO ou causale.

Notre but est de revenir au plus vite à la normale...

Approche

Pour solutionner ce problème, voici la solution que j'ai choisi de mettre en place :

- En cas de changement de topologie, on lance une élection, et on interdit les diffusions FIFO et causales (messages stockés pour un envoi une fois les propriétés topologiques rétablies).
- En fin d'élection, il appartient au serveur élu de resynchroniser les clients présent ainsi que la liste des serveurs. Pour se faire :
    - Il lance deux echos afn de collecter les données (liste des serveurs et liste des pseudos). Les détails de l'algorithme sont disponibles ici
    - Une fois qu'il a reçu une de ces listes, il lance ensuite un R broadcast pour la communiquer aux noeuds du réseau. Idem à la réception de la deuxième liste.
    - Enfin, chaque noeud qui reçoit cette liste peut à nouveau envoyer des broadcast FIFO et causaux. Il prendront par ailleurs le soin de vider les messages qui attendent.

Problèmes liés à cette approche

Nous bloquons notre émission de messages pendant le processus d'élection, et il y a une perte possible de messages en cas de déconnection d'un serveur.

Par ailleur j'ai bien conscience que **c'est une grosse rustine** ... Le bas blesse pour les opérations effectuées pendant ce méchanisme, qui sont potentiellement perdu, ou diffusé dans le mauvais ordre, etc... J'ai bien conscience de ne pas avoir donné une solution au problème que je me suis posé. Je me rend bien compte que ce problème n'a probablement pas de solutions. En revanche, je ne me voyais pas ne pas implémenter un méchanisme pour essayer de faire quelque chose contre ce problème.

Voici néanmoins l'hypothèse simplificatrice que je fais :

Nous supposerons que :

1. Aucun changement de topologie n'est effectué pendant un changement de topologie
2. Aucun message n'est C broadcasté pendant un changement de topologie
3. Aucun C broadcast n'est en cours au moment du changement de topologie.

A ces conditions, le changement de topologie est correctement effectué sans perte de message, et en respectant l'ordre causal.

Nous ne pouvons que faire des suppositions concernant la première hypothèse.

Concernant la deuxième, nous pouvons choisir de bloquer le C Broadcast manager pendant un changement de topologie, et délivrer les messages après. Aucun message n'est perdu mais on ne respecte plus l'ordre causal.

Concernant la troisième, nous pouvons ici aussi faire que des suppositions.

Mon point de vue, c'est que c'est tout de même mieux que de ne rien faire du tout... Après ce n'est pas non plus miraculeux...