Partie 4 : Réalisation d'un système distribués de gestion de plusieurs groupes de communication

La solution proposée est la suivante.

I Conception

La réalisation de ce système consiste à fixer un certain nombre de nœuds hétérogènes en termes de caractéristiques hardware mais ayant un même profil unifié pendant tout le processus de communication.
La communication entre ces nœuds se réalise par échanges de messages d'une manière 1/ distribuée 2/ décentralisée et 3/ en Peer to Peer tout en assurant 4/ un ordre absolu à la réception.

II Architecture

On a pensé à implémenter la solution sur un réseau sans fil en mode sans infrastructures (communément connu par : Réseau Ad Hoc).
La topologie construite est une topologie maillée (en mesh) référenciée par la norme IEEE 802.11s. Cette dernière est générée automatiquement après avoir précisé, comme paramètre de simulation, le nombre de nœuds à utiliser.

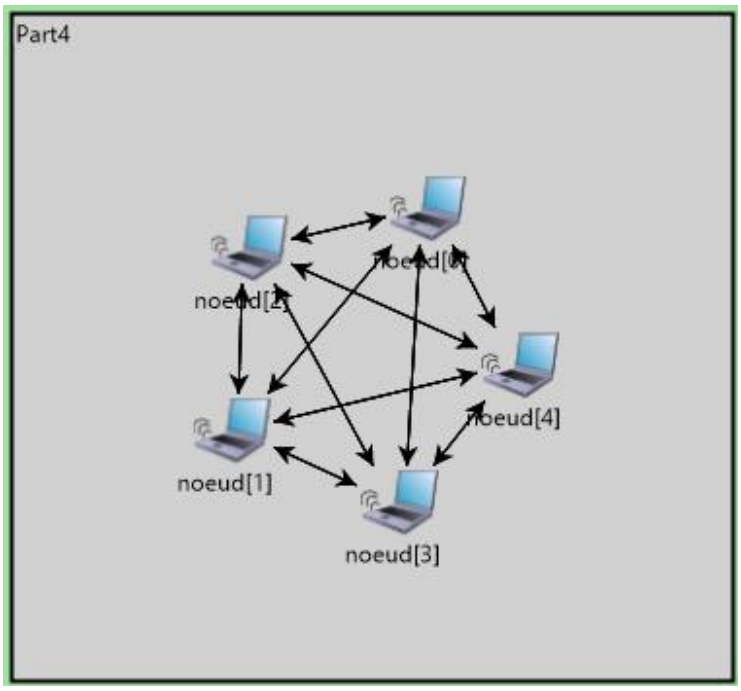

Figure 4 – Exemple de simulation de la topologie peer to peer maillée avec cinq nœuds

Chaque nœud, possède deux types d'interfaces. Chaque type assure une direction de communication (émission ou réception).

```
input inport[];
output outport[];
```

Le nombre d'interface de chaque type est égal au nombre des nœuds du réseau - 1.
Pour réaliser les liaisons entre ces nœuds, on a utilisé deux liaisons en Half-Duplex pour chaque paire de nœuds. Ceci est pour assurer la bidirectionnalité des communications comme suit.

```
noeud[i].outport++ --> {  delay = 500 ms; } --> noeud[j].inport++;
```

```
noeud[i].inport++ <-- {  delay = 500 ms; } <-- noeud[j].outport++;
```

III Structure de données et message utilisés

Chaque nœud i est identifié par une adresse IPv4 représentée par un vecteur adress_c[i][4], ce dernier est composé de 4 cases qui contiennent 1/l'ID du réseau (10.0.0.0/16) 2/l'ID du sous réseau(10.0.x.0/24) 3/ et l'ID de la machine respectivement.
Chaque nœud i possède un buffer queue[i] pour stocker les messages reçus.
Chaque nœud i possède une horloge locale horlogeL[i] initialisé à 1. Cette horloge est orientée évènement (incrémenté par réception de message).

La solution repose sur un seul type de message nommée Segment, composé de 4 champs comme suit :

```
packet Segment {
    int grp_destination;
    int estampille;
    int horloge_temporaire; //NULL par défault
    int horloge_definitve;  //NULL par défault
}
```

IV Principe de fonctionnement

Tout d'abord, chaque nœud i s'auto-attribue une adresse IPv4 ayant l'ID réseau de 10.0.0.0/16. L'ID du sous réseau (qui reflète le groupe auquel le nœud i appartient) est choisi aléatoirement entre 0 et 2. Pour éviter la duplication des adresse IP, L'ID machine prend la valeur de l'index i (sauf pour le nœud 0 où l'ID machine est exceptionnellement 254 car l'adresse 0 n'est pas attribuable).
Le nœud ayant l'id i = 0 va envoyer 2 messages de diffusions à tous les autres nœuds dans le but d'être consommés par ordre consistant à l'aide d'une implémentation du protocole ABCast.
Chaque nœud envoyé est accompagné par un numéro d'estampille.
A la réception, chaque nœud j va comparer son horlogeL[j] avec l'estampille du message reçu, et renvoi le même msg avec une horloge temporaire = max[horlogeL[j], estampille recu]+1 en diffusion (il n'y aura que le nœud i=0 qui va traiter ce message de diffusion, tous les autres nœuds vont l'ignorer).
A chaque réception d'un message avec une horloge temporaire, le nœud source (i=0) va mettre à jour son horlogeL[i], puis va diffuser le même message avec une horloge définitive = max[horloge[i], horloge temporaire]+1.
A chaque réception d'un message avec une horloge définitive, chaque nœud j va stocker le message dans son buffer queue[j].
A la fin des émission/réception, chaque nœud j va consommer, pour chaque message ayant une estampille e, le message stocké dans le buffer ayant la plus grande horloge définitive.

V Performance de la simulation

Pour deux messages envoyé par le nœud i=0 aux nœuds j (j !=i), calculons le nombre de messages échangés dans le processus de communication proposé :
Les deux messages sont envoyés en diffusions tous les nœuds : 2*(n-1) messages.

Chaque nœud j répond par un message de diffusion avec une horloge temporaire : 2*n(-1)*(n-1).
Le nœud i, à chaque réception d'un message ayant une horloge temporaire, il renvoi un message de diffusion avec une horloge définitive : 2(n-1)*(n-1).
Total de message échangés=2*(n-1)+ 2*n(-1)*(n-1)+ 2*n(-1)*(n-1).
Total de message échangés=$4n^2-6n+2$ où n=nombre de nœuds.

VI Critique de la solution proposée

Avantages

1/ Absence de point de défaillance.
2/ Efficacité lors d'envoi des messages (control ou calcul).
3/ Vu l'utilisant d'un processus de diffusion dans une topologie complètement maillée, aucun protocole de routage ne sera nécessaire dans le réseau, et donc aucune structure de donnée supplémentaire ni aucun message de control lié au protocole de routage ne seront transmis en dépit des messages de calcul (data user).

Inconvénients

1/ Cout élevé des messages de diffusion et des messages de contrôles relatifs au protocole ABcast, cela qui peut provoquer une saturation en termes de bande passante et donc des disfonctionnements des liens de transmission (overhead)
2/ Si le nombre de nœuds est élevé, les messages de diffusions échangés causeront une dégradation des performances des nœuds et pourrons même provoquer une saturation du réseau à cause des messages d'inondation (flooding message).
3/ Vu l'utilisation d'un réseau Ad Hoc qui est caractérisé par une limite des ressources des nœuds, les processus de diffusion peuvent être la cause d'une non évolutivité du réseau, c'est-à-dire que la solution ne passe pas à l'échelle (non scalable).
4/ Absence d'un protocole pour assurer la fiabilité de transmission des messages. Tout message perdu pendant le processus de transmission ne sera pas renvoyé.