Partie 1 : Réalisation d'un système de communication décentralisé

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.

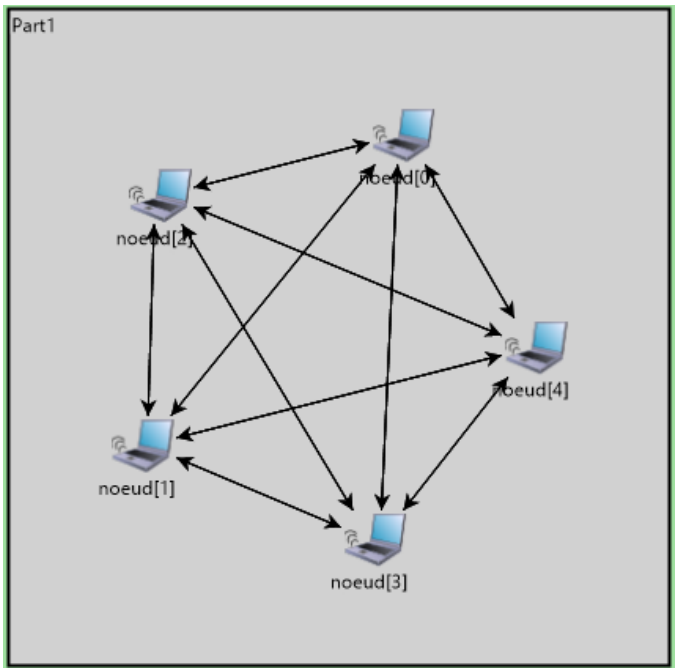

Figure 1 – Exemple de simulation de la topologie peer to peer maillée avec cinq nœuds

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 possède un entier HorlogeG[i] qui est égal au nombre total de diffusions faites au sein du réseau. HorlogeG[i] joue le rôle

d'une horloge global, elle sera incrémentée à chaque événement de diffusion produit par n'importe quel nœud dans le réseau. Quel que soit un nœud i ayant une horlogeG[i], alors pour tout nœud j ayant une HorlogeG[j], on a : HorlogeG[i] = HorlogeG[j].
Chaque nœud i possède un buffer queue[i] lui servant à stocker tous les messages reçus. Cela permet de faire un processus de tri avant la consommation pour assurer un ordre absolu à la consommation.
La solution repose sur un seul type de message nommée Segment, composé principalement des champs Horloge et N°Seq.

```
packet Segment { int num_seq; int horloge;}
```

IV Principe de fonctionnement

Chaque nœud i est identifié par un identificateur : index.
Le nœud ayant un index = 0 est l'initiateur du processus de communication, et va envoyer trois messages de diffusion de type Segment à tous les autres nœuds de du réseau qui sont directement connectée à lui (nœud 0).
Le type Segment contient les champs N°Seq et Horloge.
Le champ N°Seq reflète l'ordre par lequel les trois messages sont envoyés (le premier message aura donc un N°Seq=1, le deuxième message aura un N°Seq=2, et ainsi de suite).
Le champ Horloge contient le numéro de diffusion des trois messages. En effet, à chaque fois qu'un nœud veut envoyer un ou plusieurs messages de diffusion, il incrémente la variable HorlogeG qui est stocké localement et l'affecte au champ Horloge des messages qui a l'intention de diffusion.
La valeur de la variable HorlogeG du nœud 0 est envoyée à tous les autres nœuds à chaque envoi d'un message de diffusion. Chaque nœud j ayant reçu le message de diffusion depuis le nœud 0 récupère la valeur du champ Horloge et l'affecte à HorlogeG. Cela permet à tous les nœuds d'avoir la même valeur de la variable HorlogeG stocké localement dans chacun de ces nœuds.
Le but du champ N°Seq dans un message Segment est d'assurer un ordre absolu à la réception des messages envoyé par le même nœud i.
Le but du champ Horloge dans un message Segment est d'assurer un ordre absolu à la réception des messages envoyé par des nœuds différents.
Pour tout message reçu, chaque nœud j empile le message en question dans son buffer queue[j]. A la fin des réceptions, et avant de commencer la consommation, le nœud j va effectuer une comparaison hiérarchique commençant par les champs Horloge puis les champs N°Seq comme suit.
Le nœud j va comparer non seulement les messages ayant le champ Horloge le plus petit, mais en plus, prendre les messages ayant le N°Seq le plus bas.
Un affichage indiquant l'ordre auquel les messages ont été consommés sera fait pendant le processus du tri et de consommation.

V Performance de la simulation

Aucun message de contrôle n'est utilisé : Trafic du réseau est composé des messages de calcul (user data) seulement.

Les messages échangés sont d'un seul type : Segment. Aucun processus d'identification du type du message reçu par un nœud i n'est nécessaire (traitement monocritère).

VI Critique de la solution proposée

Avantages

1/ Une procédure de tolérance à la panne concernant la mise à jour de HorlogeG[i] d'un nœud i dans le cas où ce dernier a manqué un message de diffusion a été pris en compte. Cette solution consiste à comparer entre HorlogeG[i] local du nœud i et la valeur Horloge d'un message reçu, la valeur la plus élevée sera donc prise en considération.

2/ 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 ne seront transmis en dépit des messages de calcul (data user).

Inconvénients

1/ L'idée d'envoyer toujours les messages en broadcast n'est pas souhaitable dans le cas où on veut envoyer un message unicast ou multicast. Car elle cause une exploitation inutile des nœuds non concernés.

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).