Partie 3 : Réalisation d'un système de serveurs distribués pour la gestion de plusieurs groupes de communication

I Conception

La réalisation de ce système se base sur deux types de nœuds Client dont le nombre est à fixer et Server instancié par 3 entités.
Les nœuds clients peuvent être hétérogènes en termes de caractéristiques hardware mais ayant un même profil unifié pendant tout le processus de communication. Cela s'applique aussi pour les servers.
La communication se fait seulement par échange de message entre les clients et les servers ou entre les servers eux même.
La communication entre les servers se réalise par échanges de messages d'une manière 1/ distribuée et 2/ en mode peer to peer.
La communication entre les clients et le server se réalise par échanges de messages d'une manière 1/ distribuée 2/ en mode Client/Server tout en assurant 3/ un ordre consistant à la réception.

II Architecture

On a pensé à implémenter la solution sur un réseau filaire en mode infrastructures dans une architecture hybride.
La topologie est générée automatiquement après avoir précisé, comme paramètre de simulation, le nombre de nœuds clients à utiliser.

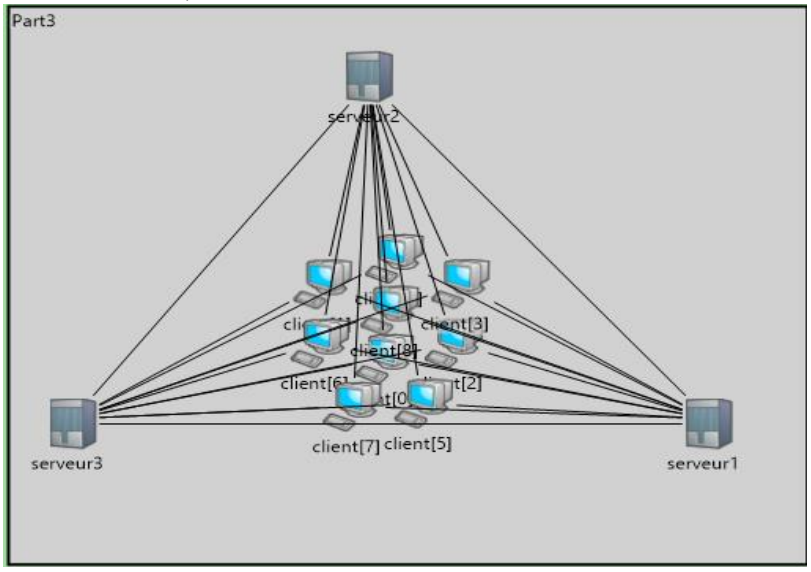

Figure 3 – Exemple de simulation de la topologie hybride avec 9 nœuds client

Chaque nœud client, possède une 1 interface de type inout pour chacun des 3 servers. Ces interfaces assurent des communications bidirectionnelles avec les servers.

```
inout inoutport1;
inout inoutport2;
inout inoutport3;
```

Chaque nœud server possède 1/ un groupe d'interfaces de types inout. Ces dernières assurent des communications bidirectionnelles avec les clients auxquelles elles sont liées au server 2/ deux interface pour le lier avec les deux autres servers.

Le nombre total d'interfaces d'un server est égal au nombre des nœuds clients + 2.
Pour réaliser les liaisons entre le server et les clients, on a utilisé des liaisons en Full-Duplex comme suit.

```
serveur1.serv1 <--> {  delay = 500 ms; } <--> serveur2.serv1;
serveur2.serv2 <--> {  delay = 500 ms; } <--> serveur3.serv1;
serveur3.serv2 <--> {  delay = 500 ms; } <--> serveur1.serv2;
client[i].inoutport1 <--> {delay = 500 ms;} <--> serveur1.inoutport++
client[i].inoutport2 <--> {delay = 500 ms;} <--> serveur2.inoutport++
client[i].inoutport3 <--> {delay =500 ms;} <--> serveur3.inoutport++;
```

III Structure de données et message utilisés

Chaque nœud client i possède 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 server contient une matrice adress_s[253][3],le nombre de ligne total = 253 = nombre maximum de nœud dans la topologie.
Chaque ligne contient 3 cases qui reflète 1/L'ID du nœud 2/l'ID du sous réseau auquel le nœud appartient 3/L'ID machine de l''adresse IPv4 de ce même nœud.
Chaque nœud i possède un buffer queue[i] lui servant à stocker tous les messages reçus. Cela permet de faire un processus de tri basé sur des numéros de séquence avant la consommation pour assurer un ordre consistant à la consommation.
La solution repose sur un seul type de message nommée Segment, composé de 5 champs comme suit :

```
packet Segment {
int source;//id du nœud source
int destination;//id du groupe de destination
int num_seq;
int msg_dhcp;//si = 1 donc c'est un msg d'attribution d'addresse
int address[4];// ladresse que le server va attribuer au client
int MAJ_DHCP;//booleen si = 1 donc cest un msg de maj de la table d'adressage envoyé par un server à un autre
int adress_maj;//id du noeud auquel son adresse va etre maj dans un autre server
int ss_reseau;//valeur du troisieme octet de ladresse (idf groupe)
int idf_mac;//valeur du 4eme octet de ladresse (idf machine)
int reply;}
```

IV Principe de fonctionnement

Tout d'abord, le server1 attribue des adresses IPv4 à ses clients, ces adresses IPv4 attribuées appartiennent toutes à la même adresses 10.0.0.0/16, mais elles se différent dans l'ID du sous réseau x 10.0.x.0/24. L'ID du sous réseau est entre 0 et 2 (donc il y'aura 3 groupes), et le server1 choisit cette variable aléatoirement. Donc on aura 3 sous réseau qui regroupe l'ensemble des nœuds. A chaque attribution d'adresse IP à un client, le server1 va envoyer un message de MAJ aux autres servers pour qu'ils mettent à jours, à leurs tours, leurs tables d'adressage.

Le nœud ayant l'ID = 0 envoie au server1 3 messages au 1er groupe (le groupe ayant l'ID du sous réseau = 0).
A la réception de chaque message, le server affecte un numéro de séquence selon l'ordre de réception, puis le diffuse à tous les nœuds membres du groupe indiqué par le client source. A la fin de la réception des 3 messages, le server1 envoi au nœud 0 un message avec le champ REPLY=1 pour indiquer qu'il est bien fonctionnel puis supprime le nœud 0 de sa liste d'adresses.
Si le nœud 0 n'a pas reçu le message REPLY, il renvoi les 3 messages en choisissant, aléatoirement, l'un des server 2 ou 3.
Une fois le message REPLY est bien reçu, le nœud 0 désactive sa pile protocolaire.
Les clients membres du groupe indiqué par le client source reçoivent les 3 messages, et les consommes selon leurs numéros de séquence.

V Performance de la simulation

Les messages de control utilisé sont : 1/ les messages d'attribution d'adresses IPv4 par le serveur à ses clients 2/ les messages de MAJ des tables de routage entre les servers et 3/ les messages REPLY pour indiquer la disponibilité du server.
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, la fonction des messages est identifiée par les valeurs des champs inclus.
La totalité des messages circulant dans le réseau est = n (où n = nombre total de nœud dans la topologie = message d'attribution d'adresse IP)+ n*3 (message des MAJ des tables d'adressage) + 1 (message de disponibilité REPLY)+ Message de calcul (user data).

VI Critique de la solution proposée

Avantages

1/ Disponibilité du server améliorée.
2/ Tolérance aux pannes qui peuvent affecter le server.
3/ Passage à l'échelle : Amélioration de l'évolutivité de la topologie (solution plus ou moins scalable mais dépend toujours des capacités des servers).
4/ Répartition de charge après défaillance (quand le server1 n'est plus disponible, les requêtes seront transmises au server2 ou bien au server3).
5/ Vu l'utilisation du message "REPLY", la fiabilité, par rapport à l'envoi et au traitement des messages de calcul, est assurée.
6/ Solution plus ouverte en termes d'extensibilité (openess).

Inconvénients

1/ Existence de plusieurs points de défaillance.
2/ Absence d'un processus de gestion de coherence entre les tables d'adressage stoquées dans chacun des trois servers.
3/ Taille plus ou moins grande du message Segment (vu qu'il sert à assurer les messages des clients et les messages de control (DHCP + MAJ_DHCP + REPLY) en même temps, ça sera plus léger d'utiliser quatre types de messages pour chaque type de donnée.