1. Introduction RFC Ly-Ieng-Debats

Ce document spécifie un protocole de réseau pour la distribution de taches parmi un groupe d'applications.

Notre protocole permet à des applications liées entre eux avec une structure d'arbre d'appliquer des conjectures et les renvoyer à l'utilisateur dans un fichier donné en paramètre par l'utilisateur.

Chaque application dans l'arbre fonctionne de tel sorte qu'il applique une fonction donnée à une gamme de valeurs qui lui sera transmise.

Une application est lancée avec un fichier Jar qui lui est remis par l'utilisateur et les valeurs sur lesquelles l'application doit faire la conjecture, cette application partage son travail avec les applications qui sont dites libre dans le réseau (c'est à dire qui ne sont pas en train de faire une tâche). Cela nous permet de faire des calculs en parallèles pour accélérer la sortie des conjectures.

2. Aperçu du protocole

Il est donné une gamme de valeurs et une fonction via un jar fournis par l'utilisateur.

Cette gamme de valeurs sera divisé en sous gamme par le nombre d'application libre et donné à ces applications du protocole.

Le nombre d'application libre sera retrouvé par un paquet ping que l'application qui a initié la tâche enverra au réseau.

Chaque application appliquera alors la fonction donnée sur la sous gamme qui lui est donnée et renvoi les résultats dans un fichier texte dont le chemin a été transmis.

Quand l'application se déconnecte, elle transmet ses taches à ses applications enfant qui les stockerons dans un buffer séparé et les traiterons après avoir fini leurs taches actuelles.

Pendant la déconnexion l'application envoi aussi une trame, signalant qu'elle va se déconnecter et devra attendre que les applications connexes lui retransmettent qu'elles vont bien se déconnecter de cette application.

Les applications enfant auront donc le buffer sur lequel ils sont actuellement en train de traiter et un buffer contenant les éléments à traiter de leurs parent déconnecté (dans un buffer dit de stockage). Ce buffer de stockage sera traité après le traitement des données principales. S'il y a eu plusieurs déconnexions parentes à une application, les données stockées seront séparées par un long de 0

Quand la root se déconnectera ce signifiera l'arrêt de chaque application.

La root n'est dé-connectable que si toutes les applications sont déconnectées.

3. Détails d'implémentations

3.1 Format de communication

Au début de chaque tâche, l'application qui l'initie envoie à ses applications connectées des trames ping d'envoie. Toutes les applications du réseau doivent recevoir ce ping et doivent répondre par un ping réponse contentant un byte disant si l'application est dite libre ou non.

Chaque application recevant une trame ping d'envoi le retransmettra à toutes ses autres connexions sauf celle d'où elle a reçu la trame et renverra un ping réponse.

Si une application reçoit une trame ping d'envoi est une feuille, elle ne fait juste que répondre.

L'application retransmettra le paquet ping de réponse à ses connexions dont il ne l'a pas reçu.

Seul l'application qui a ping pourra lire les paquet réponses.

Les urls/Chemin de jar, les noms de classes, les nom des fichiers seront encodé en ASCII.

Aussi Chaque application possèdera une table de routage donnant par quel application connexe doit passer l'information pour atteindre la destination.

La table de routage est mise à jour à chaque fois qu'une nouvelle application se connecte.

On changera la table de routage lors de la déconnexion des applications on supprimera l'application des autres tables de routage des applications de tout le réseau

3.2 Transfert de données

Le partage des données entre les applications et le protocole seront fait en paquets TCP

3.2.1 Cas d'une Tâche :

On considère qu'une application par laquelle on fait passer la tâche est l'application source de la tâche.

Quand une application demande une tâche au réseau, elle va envoyer un paquet ping à chacune de ses connexions et récupèrera quels sont les applications disponibles.

Elle enverra donc (la gamme de valeurs à traiter / le nombre d'applications disponible) paquets à ses connexions, en précisant quel paquet va vers quelle application directement dans le paquet.

Quand une application a fini de traiter les données, elle sera libre et sera en attente d'un ping d'envoi

Exemple:

Fonction X
range [1...20]
ID Source L (application actuel)
Les applications qui sont libre sont A, B, C
On enverra ces buffers vers les applications de ce nom

Opcode	Adresse A	Données...

Opcode	Adresse B	Données...

Opcode	Adresse C	Données...

Forme des données

Taille URL (long)	URL (Ascii)	Taille Nom Classe	Nom Classe	Val range min (int)	Val range max(int)	Taille Nom fichier	Nom de fichier

3.2.2 Cas d'une application en attente

Lorsqu'une application vient de se connecter ou qu'elle a fini sa conjecture et n'a pas de tâche attribuée dans le buffer de stockage, elle est en attente.

Si cette application vient de finir une tâche qui lui a été fournis elle vérifie d'abord si son buffer de stockage ne contiendrait pas des éléments à traiter si oui elle traitera ces éléments sinon elle se mettra en attente d'une nouvelle conjecture.

3.2.3 Table de routage

Lorsqu'une nouvelle application se connecte, elle envoie une trame First Leaf, qui remontra jusqu'a la Root en ajoutant son adresse.

Lors de la remonté, les applications par lesquelles la trame First Leaf passera ajouterons si ce n'est pas déja le cas dans leurs tables de routage les adresses qui sont dans la trame sachant que la dernière adresse doit être la route par laquelle on doit accéder pour aller aux autres applications.

Dès que la root reçoit une trame First Leaf, met à jours sa table de routage et elle broadcast une trame Full Tree contenant toutes les adresses clé de la table de routage et son adresse en tant que dernière adresse.

On obtient le même principe que pour la trame First Leaf cependant on le fait dans le sens inverse, vers les feuilles du réseau.

Exemple de table de routage :

On a ce réseau constitué de A B C

```
  A
 / \
B   C
```

Table de routage A	Table de routage B	Table de routage C
B -> B	A -> A	A -> A
C -> C	C -> A	B -> A

l'application G viens se connecter a C

```
  A
 / \
B   C
/
D
```

Table de routage A	Table de routage B	Table de routage C	Table de routage D	trame FirstLeaf partant de D
B -> B	A -> A	A -> A		opcode 1 D
C -> C	C -> A	B -> A		

Ajout à la table de B et envoi de la trame

Table de routage A	Table de routage B	Table de routage C	Table de routage D	trame FirstLeaf partant de B
B -> B	A -> A	A -> A		opcode 2 D B
C -> C	C -> A	B -> A		
	D -> D			

Ajout a la table de A et envoi de la trame Full Tree

Table de routage A	Table de routage B	Table de routage C	Table de routage D	trame FullTree partant de A
B -> B	A -> A	A -> A		opcode 4 B C D A
C -> C	C -> A	B -> A		
D -> B	D -> D			

Réception par B et C et envoi de la trame full Tree

Table de routage A	Table de routage B	Table de routage C	Table de routage D	trame FullTree partant de B	trame FullTree partant de C
B -> B	A -> A	A -> A		opcode 4 C D A B	opcode 4 B D A C
C -> C	C -> A	B -> A			
D -> B	D -> D	D -> A			

Réception par D de la trame Full Tree

Table de routage A	Table de routage B	Table de routage C	Table de routage D
B -> B	A -> A	A -> A	B -> B
C -> C	C -> A	B -> A	A -> B
D -> B	D -> D	D -> A	C -> B

3.3 Connexion et déconnexion d'une application de l'arbre

3.3.1 Connexion

Le réseau acceptera toujours des connexions d'applications, seulement si une conjecture est en cours les nouvelles applications ne participeront pas à cette dernière mais attendrons de recevoir/démarrer une nouvelle conjecture.

Il est à préciser que chaque application se connectera sur un port unique et une adresse unique, cependant il est possible que sa propre adresse soit partagée par plusieurs applications.

3.3.2 Déconnexion

La déconnexion d'une application se fera normalement et non brutalement.

Lors de la déconnexion de l'une des applications, cette dernière signalera sa déconnexion aux applications connexes en envoyant un paquet Annonce d'intention de déconnexion et elle transmettra les données qu'elle n'a pas encore traitée aux applications connexes avec des paquets Envoi de données à traiter aux applications en attente.

Elle séparera les données en divisant la gamme de valeur par le nombre d'applications connexes. Si son buffer de stockage contient des données à traiter d'autre application qui se sont déconnectées, elle divisera et transmettra aussi ces données en plus de l'url du jar.

Quand une application connexe à celle qui s'est déconnectée, elle reçoit les données de cette dernière, elle les stocke dans son buffer de stockage et les traiteras quand elle aura fini sa tâche principale.

Si son buffer de stockage est plein elle transmettra, à ses fils.

Pour ce qui est du stockage s'il y a plusieurs types de tâches dans le buffer de stockage, on les traitera un par un et lorsque l'application aura fini elle sera considérée comme libre.

Lors de la tentative de déconnexion, après avoir envoyé les données qu'elle devait traiter, l'application enverra un paquet Annonce d'intention de déconnexion aux application connexes signifiant qu'elle attend ses fils de se connecter à son père. Durant la procédure, et tant que tous ses fils ne lui ont pas confirmé qu'ils s'étaient connectés à son père en lui envoyant un paquet Ping de confirmation de changement de connexion, l'application agira comme si elle n'était pas libre.

Une fois que chacun des anciens fils de l'application se déconnectant sera connectés à son père, elle pourra envoyer à son père une trame suppression d'application à son père et se déconnectera du réseau.

Le père renverra à toute ses connexions cette même trame pour que chaque application mette à jour leur table de routage et supprime l'application qui s'est déconnectée.

Toutefois, si l'application est une feuille, ses données seront transmises à son père et l'application n'aura besoin de seulement la confirmation de son père pour se déconnecter.

Exemple :

Application Z en cours de déconnexion à deux applications (A,B) connexe et contient lui-même un buffer contenant des éléments d'applications qui se sont déconnecté.

A étant le père de Z et B le fils de Z

Application initiant la tâche : K

Application initiant une tache ayant une application déjà déconnecté : L,M

URL de l'application sera l'URL Z

Dès que les éléments seront traités, ils seront stockés dans un fichier réponses instantanément

Buffer Contenant les élément non traités (que l'application Z va séparer et envoyé à A et B)

Taille URL Z (long)	URL Z	taille nom	fichier Z	4	10

Buffer de stockage d'élément d'application déjà déconnecté (que l'application Z va séparer et envoyé à A et B)

Taille URL L (long)	URL L	taille nom	fichier L	5	10	0000 0000	Taille URL M	URL M	taille nom	fichier M	5	20

Lors de la déconnexion

Données envoyées a A

| OPCODE | Taille URL Z (long) | URL Z | taille nom | fichier Z | 4 | 7 | 0000 0000 | Taille URL L (long) | URL L | taille nom | fichier L | 5 | 7 | 0000 0000 | Taille URL | URL M | taille nom | fichier M | 5 | 12 |

Données envoyés a B

| OPCODE | Taille URL Z (long) | URL Z | taille nom | fichier Z | 8 | 10 | 0000 0000 | Taille URL L (long) | URL L | taille nom | fichier L |8 | 10 | 0000 0000 | Taille URL | M | URL M | taille nom | fichier M | 12 | 20 |

Intention de déconnexion envoyé à A et B

| opcode | Adresse application déco | Adresse application père|

A et B vont le recevoir, A est le père donc va l'ignorer, B est le fils donc va envoyer une demande de connexion au père

Demande de reconnexion de B vers A

| Opcode | Adresse du demandeur | op code : 2

A va accepter la connexion et envoyer à B l'acceptation de connexion (en passant par Z qui est toujours le fils de A et le père de B)

| Opcode | Op Code : 1

B va initier la connexion à A, puis envoyer à Z la confirmation de reconnexion

| Opcode | Adresse Application source | Adresse Application déco |

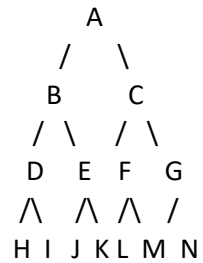
Z n'a donc plus de fils encore connecté à lui ni de données à traiter, il peut donc envoyer une trame suppression d'application a son père, signifiant qu'il ne fera plus partie réseau et qu'il faut donc le supprimer de la table de routage. Son père le transmettra à tout le réseau.

Opcode	Adresse Application déco

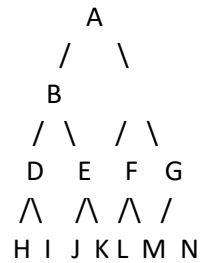
Les applications qui recevront les données, traiterons ces conjectures après avoir traité leurs propres conjectures.

Pour cela chaque application a un buffer sur lequel il traite les données et un autre buffer contenant les données de déconnexion.

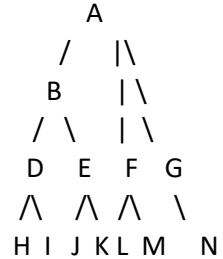
4. Structure de l'arbre



On déconnecte C



On lie F et G au père de C qui est donc A



5. Définition des trames transmis

(Comment définir un type, structure de données, opcode pour voir si c'est une réception ou un acquittement etc...)

Définition des données

Taille URL (long)	URL (Ascii)	taille nom fichier (int)	fichier réponse	Val range min (int)	Val range max(int)

Annonce d'intention de déconnexion de l'application (donc de reconnexion des fils au père)

opcode	Adresse application déco	Adresse application pere	Op code : 3

Ping de confirmation de changement de connexion

Opcode	Adresse Application source	Adresse Application déco	Op code : 4

Trame suppression d'application

Opcode	Adresse Application déco	Op code : 5

Trame First LEAF (La première trame First LEAF aura un int égal à 1 et une seule adresse, celle de la feuille qui l'a envoyé)

Opcode	Nombre Applications (int)	Adresse Application	...	Op code : 7

Trame FULL TREE

Opcode	Nombre Applications (int)	Adresse Application	...	Op code : 8

Trame ping d'envoie

Opcode	Adresse Application source qui ping	Op code : 10

Trame ping réponse

Opcode	Adresse Application source de la réponse	Adresse Application source qui ping	Byte true/false	Op code : 11

Envoi de données à traiter aux applications en attente

| Opcode | Adresse destinataire | Données | Op code : 12

Envoi de données de déconnexion à traiter aux applications connexes
---
| Opcode | Adresse source | Données | Op code : 13

Opcode sera un int
Adresse sera un inetSocket
Données sera les différentes données (taille fonctions (long) fonctions ranges) le tout séparé par un long égal à 0
Il n'y aura aucune série de 8 octet nulle autre que les long servant à délimiter les données

6. Actions des trames

-Si je reçois une trame Annonce d'intention de déconnexion de l'application (op : 3), Si je suis l'application père j'envoie une trame de suppression dans la table de routage vers la Root (op : 5) et j'envoie une trame Ping de confirmation de changement de connexion (op : 4) à l'application qui veut se déconnecter pour qu'elle se déconnecte.
-Si je reçois une trame Annonce d'intention de déconnexion de l'application (op : 3), Si je suis une application fils je me connecte à mon "Application Grand Père"

-Si je reçois une trame Ping de confirmation de changement de connexion ET que je ne suis pas l'application "Adresse Application déco", c'est que je suis le nouveau père d'un fils. J'envoie donc cette même trame à l'adresse du fils "Adresse Application déco". Si je suis l'application "Adresse Application déco", c'est que c'est moi qui me déconnecte. Si l'ancien fils "Adresse Application source" n'est pas le dernier fils à m'avoir confirmé qu'il est bien reconnecté, j'attends jusqu'à avoir la réponse de tous les fils. Si c'est le dernier fils, j'envoie la trame suppression d'application (op : 5) à mon père, et je peux me déconnecter.

-Si je suis la Root et je reçois une trame suppression d'application (op : 5), je supprime l'application "Adresse Application déco" de ma table de routage, et je renvoi une trame Full Tree (op : 8)

-Si je reçois la trame First LEAF (op : 7), si je ne suis pas root, j'incrémente "Nombre Applications" et j'envoie une nouvelle First LEAF (op : 7) en ajoutant mon adresse à la fin de la trame.
Je mets à jour la table de routage avec chaque application contenue dans la trame que j'ai reçu ayant pour chemin le fils qui me l'a envoyé
Si je suis root et que j'ai plus d'un fils, je mets à jour ma table de routage et j'envoie à chacun de mes fils une trame FULL TREE (op : 8) qui contiendra toutes les applications du réseau.

-Si je reçois la trame FULL TREE (op : 8), je mets à jour ma table de routage j'ajoute mon adresse a la trame et je l'envoi à mes fils.

-Si je reçois la trame ping d'envoie (op : 10), j'envoie cette même trame aux applications connexes hormis celle qui me l'a envoyé. Si je suis une feuille je ne le fais pas.

Si je suis libre, (qu'importe si je suis feuille) j'envoie une trame ping réponse (op : 11) avec un byte a 1 à la fin à l'application "Adresse Application source" qui a initié la trame ping d'envoie (op : 10).

Je ne serai plus libre pour les autres applications et j'attendrai les données à traiter de cette application (avec un timeout qu'on modifiera à l'implémentation)

Sinon j'envoie une trame ping réponse avec un byte a 0 a la fin

Dans la trame ping réponse (op : 11) je serai "Adresse Application source" et l'adresse qui sera dans "Adresse Application qui a ping" sera celle qui a initié la demande (Pour plus de clarté)

-Si je reçois la trame ping réponse (op : 11), si je ne suis pas l'application qui a envoyé un ping d'envoie (op : 10), je retransmets cette trame à l'application "Adresse Application qui a ping"

Si je suis cette application, je noterai que l'application "Adresse Application source" est libre, je pourrai lui envoyer des données à traiter.

-Si je reçois la trame Envoi de données à traiter aux applications en attente (op : 12), si je suis l'application "Adresse Destinataire", je traiterai les données et à la fin, j'enverrai la trame Envoi de données traitées (op : 14) vers l'application qui m'avait demandé le travail avec sa trame Trame ping d'envoie (op : 10) et pour laquelle je n'étais plus libre, s'il reste des données à traiter dans mes buffers de stockage, je les traiterai après. Si je ne suis pas cette application, je renvoie la trame vers l'application "Adresse Destinataire".

-Si je reçois la trame Envoi de données de déconnexion à traiter aux applications connexes (op : 13), si je suis libre, je ne deviens plus libre et je traite les données.

Si je n'étais pas libre, s'il reste de la place dans mes buffers de stockage, je sépare les différentes données à traiter avec des long égaux à 0 puis je stocke les données de la trame reçu dans mon buffer. Je le traiterai après la tache sur laquelle je travaille.

Si je n'ai plus de place dans mes buffers de stockage (cas rare car range de données divisée), je le transmet à mon père.

Si je suis la root et que mes buffers de stockage sont remplis, je vais envoyer une trame Trame ping d'envoie (op : 10) pour que les données soient traitées par des applications libres