Rapport de projet: Blockchain distribuée

Lucas SCHOTT

17 mai 2018

1 Introduction

1.1 Présentation

Le but de ce projet est de réaliser une blockchain distibuée et de pouvoir réaliser une simulation d'utilisation de la blockchain pour pouvoir observer son évolution dans le temps.

Dans cette blockchain distribuée, les noeuds bloc stockent la blockchain, receptionnent des demandes de transactions et mine des block pour valider ces transactions. Les noeuds bloc communiquent entre eux pour s'échanger les nouveaux block, ou envoyer une blockchain entière. Les noeuds participants font des demandes de transactions, et peuvent demander aux noeuds bloc la quantité de points blockchain dont ils disposent.

Ce projet à été réalisé en C++ en utilisant les sockets et la bibliothèque de programamtion distribuée RCF. Il est uniquement utilisable avec des adresse IPv6.

1.2 fonctionnement

Un participant peut contacter un noeud bloc et lui envoyer une demande de transacation d'un compte a un autre, le noeud bloc met alors la transaction en attente pour ensuite la miner dans un block. Le participant peut demander a un noeud bloc quelle est la quantité de points blockchain dont dispose un compte, c'est ce qu'on appelle la "balance", elle peut etre négative si un participant effectue une transaction supérieur à ce dont il dispose.

Les noeud blocs attendent d'avoir suffisement de transactions en attente pour ensuite les miner dans un block et valider les transactions. Il lui faut 5 transactions en attente pour commencer le minage. Une fois le block miné, le noeud bloc va l'envoyer à tous les autres noeuds bloc auquel il est connecté. Un noeud bloc ne peux miner qu'un block a la fois, donc toutes les transactions recu pendant un minage sont misent en attente jusqu'au minage du prochain bloc.

Plusieurs noeuds bloc peuvent se connecter entre eux, au moment de la connection de deux blocs ceux-ci s'échangent les blockchain qu'ils possèdent. Le noeud bloc garde la blockchain valide la plus longue. Ensuite les noeuds bloc s'envoient mutuellement les listes des autres noeuds bloc qu'il connaissent, et les noeuds blocs vont ainsi démarrer de nouvelles procédure de connexion avec les autres noeuds bloc. Tous les noeaud blocs sont connectés entre eux avec une topologie full-mesh. Ainsi quand un noeud bloc mine un nouveau block, il va le transmettre à tous les autres noeuds bloc. Si un noeud bloc recoit 3 block invalides de suite il va contacter un autre noeud bloc pour lui demander sa blockchain, comme ca il pourra avoir à nouveau la bonne blockchain à condition que celle-ci soit plus longue que la sienne.

Les noeuds bloc connectés s'envoient des messages de manière périodique pour maintenir leur connection, si un noeud bloc ne reçoit plus de messages d'un autre noeud bloc il va le conséderer comme mort et les deux noeuds bloc se déconnecteront. Un noeud bloc est considéré comme mort au bout de 2 secondes sans réponse. Le réseau de noeud bloc est résistants aux pannes, aux déconnexions et dysfonctionnements de certains noeuds bloc.

2 Utilisation

Les détails concernant l'intallation du programme sont disponible dans le fichier README.

```
$ ./bloc <compte> <adresse_IPv6|nom_de_domaine> <numero_de_port> [-v]
```

Sert à creer un noeud bloc unique, avec un compte pour récupere les récompenses du à son travail de minage, en lui donnant aussi une adresse IPv6 à utiliser disponible sur la machine courante et un numéro de port à utiliser, ce noeud bloc peut recevoir des demandes de transactions

```
$ ./bloc <compte>  <adresse|nom_de_domaine> <numero_de_port>
<adresse|nom_de_domaine> <numero_de_port> [-v]
```

Permet de creer un noeud bloc comme précédement et de le connecter à un autre noeud bloc déjà existant, en lui donnant en plus de sa propre adresse et numéro de son port, l'adresse (ou le nom de domaine) et le numéro de port du server auquel se connecter. Le nouveau noeud bloc communiquera avec celui déjà en place et intègrera le groupe de noeuds bloc dans lequel le noeud bloc contacté est déjà intégré.

```
$ ./participant <adresse_IPv6|nom_de_domaine> <port>
<comtpe_source> <compte_destination> <montant>[-v]
```

Permet a un participant de contacter un noeud bloc en lui faisant une demande de transaction du compte source au compte destination, du montant donné.

```
$ ./participant  <adresse_IPv6|nom_de_domaine> <port>
balance <compte> [-v]
```

Permet au noeud participant de demander au noeud bloc la balance du compte spécifié(nombre de points blockchain).

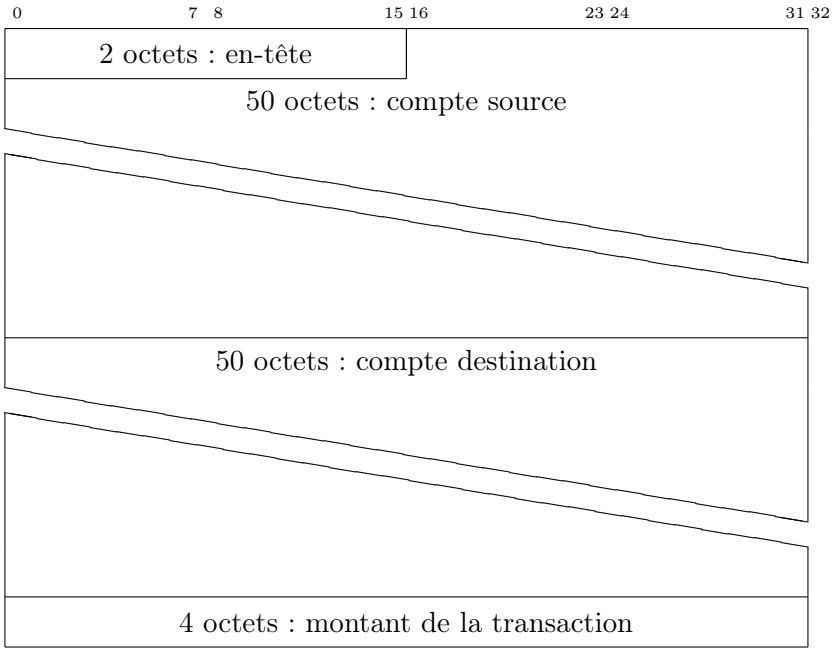
Les deux programmes, *participant* comme *bloc* peuvent prendre l'option *-v* (verbose) en argument. Avec cette option activée le programme affiche explicitement ses différentes actions, et plus particulièrement les détails de synchonisations entre server, et les émissions et réceptions de messages.

Si le prgramma *bloc* est utilisé avec l'option -d (pour display) celui ci se connectera au noeud bloc indiqué et récuperera la blockchain pour l'afficher sur la sortie standard, puis quittera.

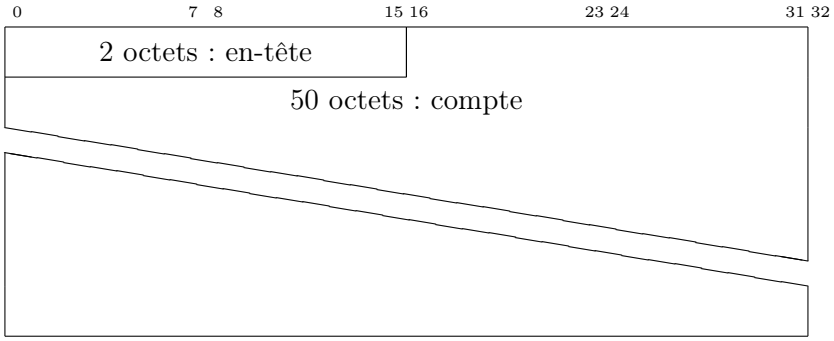
3 Format de messages

Il y a quatre formats de messages envoyés via les sockets :

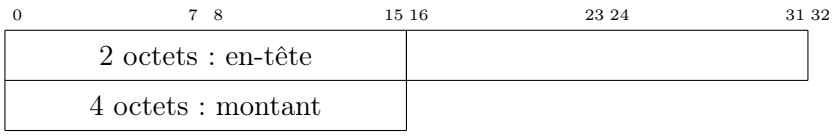
— Le format *transaction_ t* est utilisé quand un participant ou un noeud bloc envoi un transaction à un autre noeud bloc.



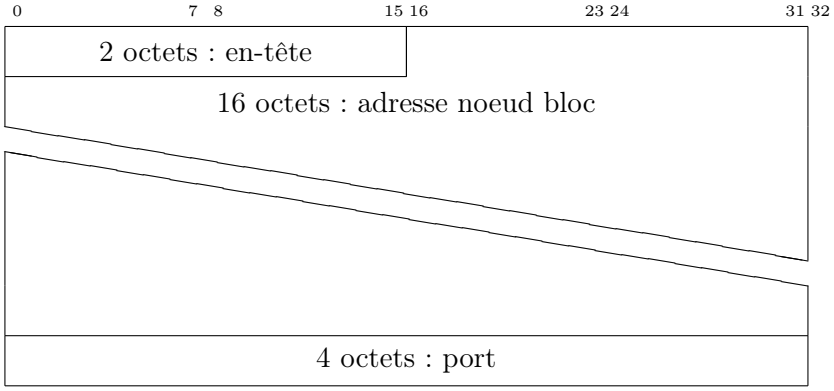
— Le format *ask_ balance_ t* est utilisé quand un participant demande sa balance à un noeud bloc.



— Le format *send_ balance_ t* est utilisé quand un noeud bloc envoi le montant de la balance d'un participant à celui ci.



— Le format *syn_ t* est utilisé quand un noeud bloc envoi des messages de connexion ou de synchronisation avec un autre noeud bloc.



Les autres informations et données échangés : les blockchain et les blocks, le sont via la bibliothèque RCF.

4 Implémentation

4.1 Réseau et Distribution

4.1.1 Transmission des transactions

Les transaction sont envoyés par un noeud participant à un noeud bloc via des sockets. À la réception de cette requete le noeud bloc, va mettre en attente la transaction avant minage du prochain block.

La demande d'un client de la balance de son compte à un noeud bloc se fait via les sockets. À la réception d'une telle requete le noeud bloc, va calculer la balance du compte, et la retourner un noeud participant via des socket.

4.1.2 Échanges de block et de blockchain

Les noeuds blocs s'échangent des blocks via un appelle à une méthode d'une interface partagé par un autre noeud bloc en utilisant la bibliothèque de programmation distribué RCF. Les échange de blockchain se font de la même manière.

4.1.3 Connexion des noeuds bloc

Les noeud blocs se connectent entre eux via des sockets. Ils s'envoient périodiquement des messages keep alive pour maintenir leur connection.

4.2 blobkchain

La Blockchain est un objet contenant un vecteur de Blocks, une récompense de minage, une difficulté de minage et des transaction en attentes qui ne sont pas encore dans la blockchain. Un Block est un objet contenant le hash du block précédent, une date, un nonce, un vecteur de Transactions, et un hash. Une Transaction est un objet contenant un compte source, un comptee destination et un montant.

A partir de 5 transactions en attente le noeud bloc va miner un block pour l'ajouter dans la blockchain. Les transactions qu'il recevera en même temps seront mis en attente pour le prochain bloc. Si un noeud bloc recoit 3 block invalides de suite il va demander une nouvelle blockchain à une autre noeud bloc. Le noeud bloc va comparer ces deux chaine jusqu'au moment ou elle divergent, a partir de la le noeud bloc va mettre toutes les transactions effectués dans les block suivant de la blockchain la plus courte dans les transactions en attente. Il va ensuite pouvoir les miner à la suite de la nouvelle blockchain mise à jour.

5 Conclusion

Le sujet du projet était très intéréssant, Il m'a appris énormément sur le programmation distribué et les nombreux avantages des librairies de programmation distribués contrairement aux socket seuls qui sont bien plus complexes à utiliser. Il a aussi été très intéressant car il m'a permis de comprendre comment fontionne et une blockchain distribué et ca a été une formidable opportinuté d'en programmer une.