Test sur le cloud d'un protocole distribué de type "blockchain", améliorations possibles

Matthieu Rambaud

Télécom Paris, 27 Septembre 2023

Une blockchain peut se voir comme une base de donnée :

- accessible à tous en lecture et en écriture, organisée sous la forme d'une chaîne ordonnée de "blocks" de taille fixe: $B_0 \leftarrow B_1 \leftarrow \dots B_n$;
- telle que les opérations d'écriture sont seulement de type "append un nouveau block": $B_n \leftarrow B_{n+1}$;
- et qui offre la garantie très importante suivante: si Alice et Bob font chacun une opération de lecture, alors l'output de l'un: $B_0 \leftarrow B_1 \leftarrow \dots \leftarrow B_i$ est nécessairement un préfixe de l'output de l'autre: $B_0 \leftarrow B_1 \leftarrow \dots B_i \leftarrow B_{i+1} \leftarrow \dots \leftarrow B_j$. Autrement dit, il n'y a pas d'historiques conflictuels.

Pour l'implémenter malgré des participants corrompus, on utilise des protocoles distribués appelés "consensus", dont les ingrédients principaux sont des hashs et des signatures. Il est possible d'accélérer les calculs en utilisant des "multi-signatures" basées sur les pairings [BDN18].

L'objectif du projet consiste tout d'abord à reproduire les tests de l'une des implémentations académiques de l'un des deux algorithmes de consensus les plus performants. L'implémentation privilégiée est celle de Jolteon [GKS+22], fournie dans [Son21]. Ce protocole "Jolteon" sera un très bon sujet de présentation d'article. Notamment leur implémentation utilise des outils de production: tokio.rs pour l'exécution asynchrone basée sur l'envoi-réception de messages, rocksDB pour la base de donnée des blocks (et dalek.rs pour la cryptographie), et fournissent des scripts pour exécuter leurs benchmarks avec Fabric.

Une solution alternative plus simple consisterait à utiliser l'implémentation en Python de "speeding Dumbo" [GZT+22], disponible sur demande. Cet algorithme étant plus difficile, l'étude d'article se concentrera sur Jolteon (qu'ils ont également réimplémenté). Par défaut les tests seraient déployés sur AWS, mais il y a de fortes chances que j'aie accès à des machines de test d'algorithmes distribués à EDF labs juste à côté (et beaucoup moins cher).

Les extensions de ce projet, si le temps le permet, seraient de modifier les implémentations actuelles pour implémenter de nouveaux protocoles. Le cas le plus simple serait une simplification de Jolteon, et/ou installer le client (en lecture et/ou écriture) sur une plateforme embarquée, dans le cadre d'un partenariat avec l'aviation militaire. Le cas le plus intéressant serait une implémentation d'améliorations de Jolteon et de speeding Dumbo, dont je peux communiquer le pseudocode de façon confidentielle sur demande. Cette dernière implémentation donnerait donc lieu à une publication d'article.

References

[BDN18] D. Boneh, M. Drijvers, and G. Neven. "Compact Multi-signatures for Smaller Blockchains". In: *ASIACRYPT*. 2018.

[GKS+22] R. Gelashvili, L. Kokoris-Kogias, A. Sonnino, A. Spiegelman, and Z. Xiang. "Jolteon and Ditto: Network-Adaptive Efficient Consensus with Asynchronous Fallback". In: *FC*. we refer to the 18 June 2021 version on arxiv. 2022.

[GZT+22] B. Guo, Y. L. and Zhenliang Lu, Q. Tang, J. Xu, and Z. Zhang. "Speeding Dumbo: Pushing Asynchronous BFT Closer to Practice". In: *NDSS*. 2022.

[Son21] A. Sonnino. *Implementation of Jolteon*. `https://github.com/asonnino/hotstuff/`. 2021.