Blockchains basées sur du Consensus Répété

Yackolley Amoussou-Guenou, Antonella del Pozzo, Maria Potop-Butucaru, Sara Tucci-Piergiovanni

Blockchains basées sur du Consensus Répété[†]

Yackolley Amoussou-Guenou[1,2] et Antonella Del Pozzo[1] et Maria Potop-Butucaru[2] et Sara Tucci-Piergiovanni[1]

[1]*Institut LIST, CEA, Université Paris-Saclay, F-91120, Palaiseau, France*
[2]*Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, F-75005 Paris, France*

Les blockchains basées sur le consensus sont considérées aujourd'hui comme étant parmi les alternatives les plus viables aux blockchains utilisant un mécanisme de Proof-of-work (Bitcoin, Ethereum, ...), ces dernières étant très énergivores et ne garantissent pas une cohérence forte. Elles ont pour but d'offrir des garanties de cohérence forte (pas de fourches) dans un système ouvert grâce à : (i) un ensemble de validateurs qui produit un bloc via une variante du protocole de consensus Practical Byzantine Fault Tolerant (PBFT), et (ii) un mécanisme de sélection qui choisit dynamiquement les nœuds qui seront validateurs pour le bloc suivant. Dans cet article, nous caractérisons précisément le problème que tentent de résoudre ces protocoles de blockchains. Nous étudions Tendermint. Nos contributions sont les suivantes : nous formalisons pour la première fois le protocole Tendermint, puis nous présentons le modèle et les hypothèses précis sous lesquels il atteint son objectif. Nous prouvons que dans un système ultimement synchrone et avec une hypothèse supplémentaire, une légère modification du protocole résout une variante (i) du consensus pour la production d'un bloc, et une variante (ii) du consensus répété pour construire la chaîne de bloc ; cela si strictement moins d'un tiers des validateurs est atteint de fautes Byzantines. Nous nous sommes ensuite intéressés à l'étude de l'équité du mécanisme de récompense dans ces blockchains. Cette étude préliminaire permet d'établir que garantir (ultimement) l'équité de la récompense requiert une communication (ultimement) synchrone.

Mots-clefs : Blockchain, Consensus, Tendermint, Équité

1 Introduction

Depuis la mise en ligne du livre blanc de Bitcoin [Nak08] en 2008, les Blockchains font partie des technologies les plus attrayantes, autant dans le domaine scientifique, que pour le grand public. Bien que la plupart des Blockchains aujourd'hui aient comme application principale la finance, elles peuvent être appliquées dans d'autres domaines (ex : médecine, notariat, ...). En effet, une Blockchain est une chaîne de blocs, cette chaîne est ordonnée grâce à un mécanisme cryptographique. Chaque bloc est lié au précédent en contenant son haché, de plus, chaque bloc contient une liste de transactions. Les blocs ne peuvent qu'être ajoutés, et pas retirés.

Une Blockchain étant une structure de données distribuée, i.e. maintenue par de multiple nœuds, mettre à jour une Blockchain en conservant sa forme de chaîne peut être problématique. Si deux nœuds essaient en même temps d'ajouter un bloc, alors il y a ce que nous appelons une *fourche*, et certains nœuds pourraient profiter de cet état d'incohérence. La *Preuve de travail* qui consiste à accorder la possibilité d'ajouter son bloc à la chaîne au nœud ayant résolu en premier un calcul cryptographique gourmand en puissance. Une fois le bloc ajouté, le nœud l'ayant fait est récompensé. Bien que la *Preuve de travail* soit considérée comme équitable, et assez efficace, elle a quelques défauts. En effet, les fourches sont toujours possible car plusieurs nœuds peuvent résoudre le problème, la communication pouvant-être non synchrone, ou certains nœuds peuvent présenter des comportements malicieux.

Des techniques ont été proposées pour éviter l'apparition de fourches dans les Blockchains. Les Blockchains utilisant un consensus (Tolérant aux fautes Byzantines – BFT) font partie des alternatives les plus intéressantes car elles garantissent la non-existence de fourche. Tendermint [Kwo14] est la première Blockchain proposant un tel mécanisme. Tendermint fonctionne dans un système ouvert et utilise un mécanisme

[†]Une première version de cet article [ADPT18] a été acceptée et présentée à OPODIS 2018.

déterministe de sélection pour sélectionner un sous-ensemble de nœuds, ce sous-ensemble utilise une variante de l'algorithme PBFT [CL02] pour décider du prochain bloc à introduire dans la chaîne, puis un nouveau sous-ensemble est sélectionné, etc. Dans cette article, pour la première fois, nous caractérisons le problème des blockchains basées sur le consensus, ensuite nous formalisons Tendermint et nous étudions sa correction. Enfin, nous étudions l'équité du mécanisme de récompense de ces Blockchains.

État de l'art Il n'y a que très récemment que le monde académique s'intéresse aux fondements de la technologie Blockchain. La plupart des études est faite sur les Blockchains qui comme Bitcoin utilisent la *Preuve de travail*. [ES18], entre d'autres, montre que pour garantir la sécurité de Bitcoin, au moins 75% des nœuds doivent être correct, et pas uniquement la majorité contrairement à la croyance populaire.

Red Belly [CGLR17] est une Blockchain fermée, les nœuds pouvant ajouter et valider des transactions sont connus à l'avance. Il existe cependant une variante de RedBelly, ComChain [VG18], où les nœuds ne sont pas connu à l'avance. [CGLR17] utilise le *consensus binaire* pour construire sa Blockchain. Plusieurs autres propositions, utilisant des résultats classiques des systèmes distribués ont été faites, mais la plupart ne sont pas encore implémentées. Aucun travail académique n'a cependant fait le lien entre les travaux académiques des spécifications de consensus répétés, et le mécanisme dans les Blockchains, où chaque bloc nécessite un certain "accord", et cela de manière répétée pour chaque bloc de la chaîne.

Tous les travaux concernant l'équité dans les Blockchains, pour la répartition des récompenses, ont été menés sur les Blockchains à la *Preuve de travail* (ex : [ES18]), et aucun n'a été mené sur les blockchains basées sur un consensus répété, ou avec un comité.

2 Modèle du Système et Définition du Problème

Le système est composé d'un ensemble infini Π de processus séquentiels et asynchrones. Les processus communiquent en s'échangeant des messages par un réseau ultimement synchrone.

Nous faisons l'hypothèse d'un modèle à *arrivé fini*, i.e., il y a un nombre infini de processus, mais chaque exécution ne concerne qu'un nombre fini de processus noté Π_ρ. Les processus participants à l'exécution ne sont pas connu d'avance, et la taille de l'ensemble des processus participants n'est pas connu. Nous considérons un sous-ensemble $V \subseteq \Pi_\rho$ des processus participants, appelés *validateurs*, qui peut changer durant l'exécution, mais dont la taille $|V| = n$ est connue et reste fixe. Un processus intègre l'ensemble des validateurs par rapport à un paramètre de mérite.

Un processus suivant le protocole défini est qualifié de *correct*. Nous ne restreignons pas le nombre de processus Byzantins (comportement arbitraire) dans le système, mais à chaque moment de l'exécution, nous supposons qu'au plus $f < n/3$ validateurs peuvent être Byzantins.

Nous supposons aussi la présence d'une primitive de diffusion sur Π_ρ. Cette primitive est au meilleur effort, i.e., si un processus correct diffuse un message, ultimement tous les processus corrects le recevront. Un processus reçoit un message en exécutant la primitive delivery(). Les messages sont signés, et les signatures ne peuvent être falsifiées.

Problème. Dans ce papier, nous étudions la correction du protocole Tendermint par rapport aux deux abstractions en système distribué suivant : le *consensus (unique)*, et le *consensus répété*. Pour ce faire, nous utilisons le concept de *validité externe* qui a été initialement défini par [CKPS01] pour le problème d'accord multi-valué, et que [CGLR17] a adapté pour les Blockchains.

Définition 2.1 (Consensus Unique) *Un algorithme implémente le consensus unique ssi il satisfait les propriétés suivantes : **Terminaison.** Tous les processus corrects décident ultimement une valeur. **Intégrité.** Aucun processus correct ne décide plus d'une fois. **Accord.** Si un processus correct décide une valeur B, ultimement, tous les processus corrects décideront B. **Validité.** Une valeur décidée est valide, elle satisfait un prédicat prédéfini.*

Delporte-Gallet *et al.* [DDF+08] définissent le consensus répété comme une suite infinie d'occurrences d'un consensus unique, où entre chaque occurrences les paramètres peuvent être différents, mais tous les processus corrects doivent avoir la même suite infinie de décisions. Nous considérons le consensus répété [DDF+08] étendu aux fautes Byzantines. Chaque processus correct produit une suite de décisions, sa *sortie*.

Définition 2.2 (Consensus Répété) *Un algorithme implémente le consensus répété ssi il satisfait les propriétés suivantes :* ***Terminaison.*** *Tous les processus corrects ont une sortie infinie.* ***Accord.*** *Si la $k^{ème}$ valeur de la sortie d'un processus correct est B, alors B est la $k^{ème}$ valeur de la sortie de tout processus correct.* ***Validité.*** *Toute valeur dans la sortie d'un processus correct est valide, elle satisfait un prédicat prédéfini.*

3 Description des Protocoles et Résultats

3.1 Algorithme de Consensus Unique de Tendermint

L'algorithme de consensus unique a pour objectif de permettre aux validateurs, de produire un bloc. Les validateurs sont sélectionnés de manière déterministe par rapport à l'histoire de la Blockchain. Pour une hauteur donnée, l'algorithme pour le consensus unique se déroule en *tour*s, et chaque tour a un proposeur choisi par round-robin. Un tour se décompose en 3 phases : PROPOSE, PREVOTE, et PRECOMMIT. Chaque phase à une durée maximale qui peut être augmentée dans certaines cas, cela pour garantir la propriété de terminaison. Quand un processus ne reçoit pas *suffisamment* de messages lors d'une phase (la proposition dans la phase PROPOSE, $2n/3+1$ de prévotes dans la phase PREVOTE, $2n/3+1$ de précommits dans la phase PRECOMMIT) et que le chronomètre correspondant à la phase expire, le processus augmente la durée de ce chronomètre pour le tour suivant.

Quand un validateur reçoit au moins $2n/3+1$ prévotes pour un bloc B pour le tour r, nous disons que ce validateur a un *polc* (pour *Proof-Of-LoCk*). Quand un validateur reçoit plus de $2n/3+1$ prévotes pour un bloc B durant un tour r, il se *verrouille* sur B au tour de *verrou* r : cela signifie qu'il y a au moins $n/3+1$ processus corrects qui ont accepté B. Un processus verrouillé sur un bloc B pendant un tour r, peut se déverrouiller s'il a un *polc* pour un bloc $B' \neq B$ pour un tour $r' > r$. Lorsqu'un processus reçoit un nouveau message, il le diffuse aux autres. Quand un validateur reçoit au moins $2n/3+1$ précommits de différents processus pour la même valeur B pour un même tour, ce processus **décide** B et termine.

Notre analyse nous a permis de mettre en évidence des bugs dans la spécification de l'algorithme, ensuite nous avons proposer des corrections dont fournissons la preuve de correction. Les bugs que nous avons identifiés (mauvaise gestion du chronomètre, mauvaises conditions de verrouillage) entraînent une violation de la propriété d'*accord* du consensus. De plus une hypothèse supplémentaire est nécessaire pour garantir la *terminaison*.

1. Phase PROPOSE : Un validateur est le proposeur du tour. Le proposeur du tour propose son bloc selon son état. S'il est verrouillé sur une valeur B, alors il propose et diffuse B et le tour de verrou.
2. Phase PREVOTE : Le validateur vérifie la validité de la proposition reçue. Si (i) le validateur n'est pas verrouillé et que la proposition est valide, ou (ii) le validateur reçoit comme proposition le bloc sur lequel il est verrouillé, ou (iii) la proposition contient un bloc qui a reçu un *polc* plus récemment que le tour de verrou du processus (déverrouillage), alors il diffuse son prévote pour la proposition, sinon il diffuse un prévote pour la valeur par défaut *nil*. Puis attend l'expiration du chronomètre ou la réception d'au moins $2n/3+1$ prevotes de différents validateurs, puis passe à la phase PRECOMMIT.
3. Phase PRECOMMIT : Si le processus a reçu un *polc* pour un bloc B, alors il se verrouille sur B, met à jour le jour le tour de verrou, et diffuse son précommit pour B. Sinon il envoie un précommit pour la valeur *nil* et attend de recevoir au moins $2n/3$ précommit, ou l'expiration du chronomètre, puis passe au tour suivant.

3.2 Algorithme de Consensus Répété de Tendermint

Rappelons que l'ensemble de validateurs V_h est fixe pour une hauteur h, mais qu'elle peut changer d'une hauteur à l'autre tout en conservant toujours une taille fixe, $\forall h \geq 1, |V_h| = n$, et que $f < n/3$.

L'algorithme pour le consensus répété fait une succession de consensus unique. Par contre, comme tous les processus ne participent pas à chaque instance de consensus unique, il faut que les processus qui ne sont pas validateur pour une hauteur mette la valeur décidé/produite par les validateurs dans leur *sortie*. Soit p un processus. Pour une hauteur h, si $p \in V_h$, alors il exécute l'algorithme de consensus unique, et diffuse sa décision, qui est le bloc à la position h dans sa *sortie*. Si $p \notin V_h$, alors il attend de recevoir le même bloc par au moins $f+1$ validateurs différents dans V_h, et ce bloc sera le bloc à la position h dans sa *sortie*.

3.3 Résultats

Correction de Tendermint Nous avons prouvé que : *Dans un système ultimement synchrone, et en présence de moins de $n/3$ Byzantins par comité de n processus,*

- *Une version légèrement modifiée de Tendermint implémente le consensus unique si pendant la période de synchronie il y a une proposition valide qui sera accepté par au moins 2n/3 membres du comité ;*
- *Avec le consensus unique, l'algorithme répété de Tendermint implémente le consensus Répété.*

Équité Nous proposons dans cet article une définition de l'équité dans les Blockchains utilisant un comité.

- Le *mécanisme de sélection* qui est en charge de sélectionner les processus dans les comités. Il est équitable si les processus sont équitablement sélectionnés selon un certain paramètre de mérite ; et
- Le *mécanisme de récompense* qui récompense les membres du comité une fois un bloc produit. Il est (ultimement) équitable ssi les processus corrects sont (ultimement) exactement ceux récompensés.

Nous disons qu'un protocole de Blockchain basé sur des comités est (ultimement) équitable ssi le mécanisme de sélection est équitable, et que le mécanisme de récompense est (ultimement) équitable.

Notre étude préliminaire sur le mécanisme de récompense nous permet d'établir la condition nécessaire suivante : *Un système (ultimement) synchrone est nécessaire pour avoir un mécanisme de récompense (ultimement) équitable, même sans présence de fautes*.

4 Conclusions et Perspectives

Cet article décrit la spécification du consensus que tente d'implémenter les Blockchains. Cette formalisation, nous a permis de reporter quelques bugs qui ont permis à la fondation Tendermint de proposer un nouvel algorithme [BKM18], qui tente de résoudre le consensus unique sans hypothèse supplémentaire. La correction de cette nouvelle version du protocole est toujours sujette à discussions.

Nous nous sommes ensuite intéressé au problème de la récompense dans les blockchains utilisant des comités pour la production de blocs, et en particulier la question de l'équité dans la distribution de la récompense. Notre étude préliminaire nous permet d'exhiber qu'un système (ultimement) synchrone est une condition nécessaire, pour avoir un mécanisme de récompense (ultimement) équitable.

Références

[ADPT18] Yackolley Amoussou-Guenou, Antonella Del Pozzo, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Correctness of Tendermint-Core Blockchains. In *OPODIS 2018, December 17-19, 2018, Hong Kong, China*, pages 16 :1–16 :16, 2018.

[BKM18] E. Buchman, J. Kwon, and Z. Milosevic. The latest gossip on BFT consensus. *CoRR*, abs/1807.04938v1, jul 2018.

[CGLR17] Tyler Crain, Vincent Gramoli, Mikel Larrea, and Michel Raynal. (Leader / Randomization / Signature)-free Byzantine Consensus for Consortium Blockchains, 2017.

[CKPS01] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and efficient asynchronous broadcast protocols (extended abstract. In *in Advances in Cryptology : CRYPTO 2001*, pages 524–541. Springer, 2001.

[CL02] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4) :398–461, November 2002.

[DDF+08] Carole Delporte-Gallet, Stéphane Devismes, Hugues Fauconnier, Franck Petit, and Sam Toueg. With finite memory consensus is easier than reliable broadcast. In *Principles of Distributed Systems, 12th International Conference, OPODIS 2008, Luxor, Egypt, December 15-18, 2008. Proceedings*, pages 41–57, 2008.

[ES18] Ittay Eyal and Emin Gün Sirer. Majority is not enough : bitcoin mining is vulnerable. *Commun. ACM*, 61(7) :95–102, 2018.

[Kwo14] Jae Kwon. Tendermint : Consensus without mining. Technical report, Tendermint, 2014.

[Nak08] S. Nakamoto. Bitcoin : A Peer-to-Peer Electronic Cash System. `https://bitcoin.org/bitcoin.pdf` (visited on 2018-05-22), 2008.

[VG18] Guillaume Vizier and Vincent Gramoli. ComChain : Bridging the Gap Between Public and Consortium Blockchains. In *Proceedings of the IEEE International Conference on Blockchain (Blockchain'18)*, 2018.