

Performance et sécurité d'une Blockchain auto-adaptative et innovante

Guilain Leduc

▶ **To cite this version:**

Guilain Leduc. Performance et sécurité d'une Blockchain auto-adaptative et innovante. Automatique / Robotique. Université de Lorraine, 2022. Français. NNT : 2022LORR0220 . tel-04026958

HAL Id: tel-04026958
https://hal.univ-lorraine.fr/tel-04026958

Submitted on 13 Mar 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE LORRAINE | BIBLIOTHÈQUES UNIVERSITAIRES

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact bibliothèque : ddoc-theses-contact@univ-lorraine.fr
(Cette adresse ne permet pas de contacter les auteurs)

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Performance et sécurité d'une Blockchain auto-adaptative et innovante

THÈSE

présentée et soutenue publiquement le 14 décembre 2022

pour l'obtention du

Doctorat de l'Université de Lorraine

(en Sciences, spécialité Automatique, Traitement du Signal et Génie Informatique)

par

Guilain Leduc

Composition du jury

Président :	STATE Radu	Professeur, Université du Luxembourg
Rapporteurs :	LACAN Jérôme	Professeur, ISAE-SUPAERO
	MELLOUK Abdelhamid	Professeur, Université Paris-Est Créteil
Examinateurs :	CHINNICI Marta	Chercheuse senior, ENEA C.R. Casaccia, Italie
	GEORGES Jean-Philippe	Professeur, Université de Lorraine (Directeur de thèse)
	KUBLER Sylvain	Maître de conférences, Université de Lorraine (Co-encadrant)

CRAN — UMR 7039

CRAN

Mis en page avec la classe thesul.

Remerciements

Cette thèse est l'achèvement d'un long travail de trois années éprouvantes, mais riches en expériences et en rencontres. Je n'aurais pas pu terminer cette étape sans la contribution des personnes qui m'ont entouré et aidé.

Je remercie d'abord mes deux encadrants de thèse. D'abord le Pr. Jean Philippe Georges, qui m'a conseillé au fur et à mesure de cette thèse au prix de nombreuses prises de bec et discussions enflammées. J'ai beaucoup appris sur la recherche, ses habitudes et la beauté de la science.

Je remercie également également le Dr. Sylvain Kubler. Il a été ma porte d'entrée dans la recherche à l'occasion d'un stage cherché. Cela a été le début d'une belle aventure qui continue à l'Université du Luxembourg.

L'ambition de devenir docteur date depuis que je suis gamin. Je remercie ainsi tous les enseignants et professeurs qui ont l'occasion de me conseiller au long de mes études, de ma vieille école de mon village natal à mes enseignants de classe préparatoire, de mon école d'ingénieur Télécom Nancy et ceux de mon master de l'Université de Lorraine. Je remercie ainsi Pr. Didier Galmiche, qui a été l'un des derniers de cette liste avant d'attaquer le doctorat.

La thèse est aussi une aventure humaine. J'ai été heureux de la réaliser avec mes deux comparses du laboratoire Loïc et Paul. Nous l'avons fait. Nous sommes arrivés ensemble et désormais docteurs ensemble. Mais d'autres papiers à signer et verres à boire nous attendent dans le futur.

Je remercie mes divers amis qui m'ont accompagné durant ces épreuves, dont ceux de la Paintine.

Merci également à ma compagne Malvina, qui m'a accompagné et plutôt supporté durant cette épreuve. Ça n'a pas été facile tous les jours, mais cette épreuve est désormais finie.

Merci aussi à mes parents. Merci déjà de m'avoir supporté déblatérer des termes informatiques pendant autant d'années, mais aussi de m'avoir permis de suivre mes études jusqu'au point où j'en suis. Vous avez toujours été là pour moi.

Merci aussi à Christine, dont la présence m'a été indispensable à mes côtés. Merci également à Leslie Lamport, dont les travaux ont été fondamentaux pour le cadre de cette thèse et pour avoir construit ce formidable outil qu'est LaTeX.

Cette liste est sans fin, mais je suis heureux d'avoir achevé cette étape dans mon plan de domination du monde. Ce n'est que le début de la prochaine étape.

Je dédie cette thèse
à ma machine.
Oui, à Christine,
qui fut la première de toutes.

Table des matières

Introduction générale **1**

Chapitre 1
Des systèmes distribués à la blockchain **5**

1.1 Notions de cryptographie 5
1.1.1 Fonction de hachage 6
1.1.2 Chiffrement 7
1.1.3 Authentification 7
1.2 Notions de systèmes distribués 8
1.2.1 Modèle de Duplication de Machines à États 9
1.2.2 Cohérence 9
1.2.3 Exactitude 10
1.2.4 Disponibilité 11
1.2.5 Le problème du consensus 11
1.2.6 Principaux théorèmes 13
1.3 De la nécessité d'utiliser une blockchain 14
1.4 La blockchain dans le détail 17
1.4.1 Couche donnée 18
1.4.2 Couche réseau 19
1.4.3 Couche contrôle (d'accès) 22
1.4.4 Couche consensus 24
1.4.5 Couche contrat 31
1.4.6 Couche application 32
1.5 Conclusion 33

Chapitre 2
Problématiques de performance et de sécurité : état de l'art **35**

2.1 Mesures des performances de blockchain 36
2.1.1 Comparaison générale des consensus 36

2.1.2 Insuffisance de cette approche 37
2.1.3 Mesure des performances d'une blockchain et comparaison 37
2.1.4 Bilan des métriques mesurées 40
2.2 Solutions d'optimisation des performances de blockchain 42
2.2.1 Optimisation d'architectures 43
2.2.2 *Sharding* 43
2.2.3 Cas des chaînes BFT : optimisation par sélection des validateurs 46
2.3 Positionnement 50
2.4 Conclusion 50

Chapitre 3
Une première proposition autour de l'évaluation des *Smart contracts* 53

3.1 Identifications des liens entre les éléments 53
3.2 Contexte IoT et *Smart Farming* 56
3.2.1 État des lieux des initiatives de *Smart Farming* 57
3.2.2 Vue d'ensemble de FarMarket 57
3.3 Mesure de la *capacité* et résultats 62
3.3.1 Définition d'un plan d'expérience 62
3.3.2 Analyse de la capacité de la FarMarketChain 65
3.3.3 Discussions 67
3.3.4 Limites de l'approche 68
3.4 Une adaptation : la difficulté 69
3.4.1 Définition de la difficulté 69
3.4.2 Relation entre la difficulté et le temps de minage 70
3.4.3 Validité du modèle 71
3.4.4 Résolution du calcul de la difficulté 71
3.5 Conclusion 74

Chapitre 4
Adaptation en ligne de la taille du pool de validateurs 77

4.1 Description des bancs d'expérimentations 78
4.1.1 Structure de la chaîne 78
4.1.2 Description des plateformes 81
4.2 Objectif de la démarche 83
4.2.1 Impact du nombre de validateurs 83
4.2.2 Commande en boucle fermée 88
4.3 Fonctionnement de Sabine 90

4.3.1 Phase d'entraînement . . . 90
4.3.2 Phase opératoire . . . 92
4.4 Experimentation . . . 94
4.4.1 Modèle . . . 94
4.4.2 Consigne dynamique . . . 97
4.4.3 Adaptation à un délai . . . 97
4.4.4 Consigne et délais variables . . . 98
4.5 Conclusion . . . 98

Chapitre 5
Solutions pour la sélection des validateurs 101

5.1 Présentation des problématiques associées à la sélection . . . 102
5.1.1 Sécurité . . . 102
5.1.2 Efficacité . . . 102
5.1.3 Dynamisme . . . 102
5.1.4 Décentralisation . . . 102
5.1.5 Efficience . . . 103
5.2 Système par réputation . . . 104
5.2.1 Calcul d'une réputation locale . . . 104
5.2.2 Calcul de la réputation globale . . . 105
5.2.3 Validation . . . 105
5.2.4 Discussions . . . 107
5.3 Exploitation de la proximité . . . 108
5.3.1 Sous-graphe optimal . . . 108
5.3.2 Sous-graphe approché . . . 109
5.4 Sélection centrique . . . 111
5.4.1 Distance compensée . . . 111
5.4.2 Fonctionnement du protocole . . . 112
5.4.3 Validation . . . 113
5.5 Perspectives . . . 116
5.5.1 Dynamique de la sélection . . . 116
5.5.2 Interactions avec Sabine . . . 117
5.6 Conclusion . . . 118

Conclusions et perspectives 119

Publications 123

Annexes

Annexe A
Annexes

A.1 Autres protocoles basés sur des votes 125
A.1.1 Algorithme CFT : Paxos 125
A.1.2 Consensus de Hyperledger Fabric 128
A.2 État de l'Art des solutions agricoles basées sur la blockchain 131

Références

Table des figures

1.1 Plan du chapitre 6
1.2 Exemple de désynchronisation dans deux réplicas sans consensus 12
1.3 Exemple de consensus générique entre trois processus 12
1.4 Organigramme du *DHS Science & Technology Directorate* pour la proposition d'alternative à la blockchain (Yaga et al., 2018) 16
1.5 Pile d'architecture de blockchain et questions de recherche associées (Yuan & Wang, 2018) 18
1.6 Schéma d'une blockchain 19
1.7 Schéma d'une topologie complète composé de 5 nœuds 21
1.8 Représentation d'un hypercube de dimension 4 21
1.9 Sélection d'un modèle de blockchain (d'après Fernández-Caramés et Fraga-Lamas (2018)) 24
1.10 Diagramme représentant le flux de message dans un cas général qui a réussi (Castro & Liskov, 1999). 29

2.1 Modèle de Blockchain en 5 couches (Dinh et al., 2017 ; Dong et al., 2019 ; Fan et al., 2020) 38
2.2 Exemple de partitionnement en validateurs et sous-validateurs. Les nœuds de couleur bleue sont non-validateurs et ceux de couleurs sont des validateurs. 46

3.1 Modèle d'interaction des performances en ce qui concerne les paramètres liés à la blockchain et à l'infrastructure (notez qu'il n'y a aucune différence entre les flèches pleines et les flèches en pointillés, elles ne sont utilisées que pour la clarté de la figure). 54
3.2 Valeurs principales sur la chaîne principale d'Ethereum, d'après Etherscan.io (Etherscan.io, 2022) 55
3.3 Vue d'ensemble de l'écosystème FarMarket et des interactions associées entre les parties prenantes et le marché de soutien. 57
3.4 Protocole de messagerie soutenant l'écosystème FarMarket 59
3.5 Difficulté (de minage) par horodatage des blocs 63
3.6 Méthodologie de détermination des plans d'expériences 63
3.7 Évolution des métriques QoS pour différents nombres d'expériences (n) et dans le temps (τ) 64
3.8 Analyse expérimentale de la *capacité* de la blockchain 66
3.9 Comparaison de l'évolution de la *capacité* pour deux contrats intelligents 67
3.10 Valeurs principales sur la chaîne principale Bitcoin, d'après Bitcoinity (Bitcoinity.org, 2022) 69

3.11 Évolution du temps de minage moyen mesuré en fonction du nombre de *threads* sur une machine . . . 72
3.12 Évolution de l'inverse du temps de minage moyen mesuré en fonction du nombre de *threads* sur une machine . . . 73
3.13 Solution approchée du délai de minage τ de l'équation (3.11) en fonction d'un délai de propagation δ_t . . . 74

4.1 Divers protocoles utilisés par la chaîne . . . 80
4.2 Diagramme d'état de la version modifiée du consensus IBFT . . . 80
4.3 Photo du banc d'expérience composé de Raspberry PI . . . 82
4.4 Comparaison entre le nombre de messages avec et sans agrégat des messages intermédiaires, en fonction du nombre de validateurs pour 200 nœuds . . . 84
4.5 Expérience menée pendant 6 heures sur le Grid5000 avec 200 nœuds, sans délai avec un débit fixe . . . 87
4.6 Temps nécessaire pour accomplir chaque étape pour un nœud dans une chaîne soumise à une demande fixe supérieure à sa capacité pendant 4000 secondes . . . 89
4.7 Phases de Sabine . . . 90
4.8 Paramètres du modèle avec Sklearn . . . 91
4.9 Modèles estimés par *Machine Learning* . . . 92
4.10 Calcul de Sabine expliqué avec un diagramme, chaque bloc représente une fonction détaillée dans la section 4.3.2 . . . 93
4.11 Comparaison du débit ajouté pour une demande commune pour les chaînes avec et sans Sabine (PI et G5K) . . . 95
4.12 Réponses de Sabine aux variations de délai . . . 96

5.1 Mesures moyennes diverses sur 400 itérations en fonction du nombre de validateurs sur une chaîne composé de 200 nœuds . . . 106
5.2 Algorithme de recherche pour un sous-graphe . . . 110
5.3 Délai de la meilleure et de la pire approximation de la clique optimale par algorithme génétique, associé avec une estimation du délai pour une sélection aléatoire 111
5.4 Représentation de la sélection des validateurs, décrits dans 5.4.2 . . . 113
5.5 Exemple de 6 itérations de la sélection centrique, avec 200 nœuds . . . 115
5.6 Mesures moyennes diverses sur 2000 itérations en fonction du nombre de validateurs sur une chaîne composé de 200 nœuds . . . 116

A.1 Diagramme représentant le flux de message dans un cas général qui a réussi (Lamport, 2001) avec $v' = Max_{propo}(\{(v_i, n_i)\}_{i\in\{1,2,3\}}, v)$. . . . 127
A.2 Diagramme représentant le flux de message dans un cas général qui a réussi (Androulaki et al., 2018). . . . 129
A.3 Chaîne de transport de produits agricoles traditionnelle . . . 131

Introduction générale

Au commencement était le *Web 1.0*. Dans les années 90, les usagers pouvaient consulter des pages et s'envoyer des fichiers. Ces pages étaient principalement statiques et peu interactives. Ces pages, principalement statiques et peu interactives, reposaient sur un modèle requérant que les créateurs de sites devaient héberger eux-mêmes leur site, le flux d'informations étant unidirectionnel (du site vers le lecteur). Bien que ce modèle permettait à ses créateurs de conserver la main sur leurs données, ceux-ci ne disposaient que d'une diffusion restreinte, tandis que l'infrastructure et les compétences nécessaires pour la mise en place d'un site limitaient l'émergence de contenu.

Des solutions clefs en main sont donc apparues en réponse avec l'émergence de plateformes commerciales, lesquelles ont rendue possible le stockage et la mise en relation des utilisateurs au travers de ces plateformes. Les usagers sont devenus au fur et à mesure des avancées et des fonctionnalités, bien plus acteurs dans les échanges et la création de contenu, menant à une nouvelle version de l'Internet, nommée *Web 2.0*. Ce concept, introduit par Tim O'Reilly (2005), décrit ce nouveau Web comme participatif, social et collectif. Cette époque a connu l'avènement des sites marchands, des hébergeurs de contenus et des réseaux sociaux, où chaque utilisateur pouvait échanger au travers de la plateforme, devenant à la fois créateur et consommateur de contenus. Néanmoins, ces services et plateformes sont la propriété d'une poignée d'entités centralisées, formant ce que l'on appelle des tiers de confiance, qui, en monopolisant les interactions, rendent les utilisateurs dépendant de leurs décisions. Par exemple, un achat d'un bien sur Internet réunit une longue succession d'intermédiaires ayant chacun un droit de regard et pouvant influencer la transaction. Des problématiques de monopole, de confidentialité et d'indépendance sont donc apparues vis-à-vis de ces plateformes émergentes.

En réponse à cette centralisation (qui implique une perte de contrôle des données par les utilisateurs), de nombreux groupes font la promotion du *Web 3.0* (Cao, 2022). Beaucoup de définitions gravitent autour de ce terme, mais la version choisie ici est celle de Gavin Wood et prône un partage d'information et de contenu par un écosystème ouvert et décentralisé (Wood, 2018). Le Web 3.0 vise à supprimer ces tiers de confiance, favorisant le paradigme de tiers de confiance et amenant donc plus de transparence et d'implications sur les décisions prises, selon le paradigme "*Less trust, more truth*" (Edelman, 2022). Ce nouveau Web se base sur des protocoles standardisés, afin de faciliter l'interopérabilité entre applications, et exploiter également les dernières innovations des technologies pair-à-pair. L'objectif est d'abandonner le modèle client-serveur au profit de solutions décentralisées comme IPFS (*InterPlanetary File System*) pour le stockage (Daniel & Tschorsch, 2022) ou BitTorrent pour l'échange de fichiers (Cohen, 2003).

Ce cadre de pensée libertarien a été soutenu dans la fin des années 1990 par les milieux "cypherpunk" (Ramiro & de Queiroz, 2022), pour qui la suppression des institutions financières et autres autorités centrales apparaissent comme un objectif important pour la société. Des propositions d'alternatives ont donc été apportées sans grand succès par ces groupes, comme DigiCash ou HashCash de Adam Back (Back, 2002), jusqu'à l'introduction de la cryptomonnaie

Bitcoin par Nakamoto (2008) et la technologie sous-jacente appelée "blockchain", permettant de répondre aux problématiques de transmissions d'informations tout en sécurisant les transactions. La blockchain apparaît aujourd'hui comme une pierre angulaire du Web 3.0 et des applications concrètes apparaissent de jour en jour, comme les cryptomonnaies avec Bitcoin, ou les jetons non fongibles (*NFT*) révolutionnant le concept de propriétés (Q. Wang et al., 2021). La popularité de cette technologie n'a eu de cesse d'augmenter, et les propositions de blockchains n'ont eu de cesse de croître. Plusieurs milliers de propositions de références sont ainsi rapportés dans les différents marchés de ces cryptomonnaies (*Toutes les Crypto-monnaies*, 2022).

L'engouement pour les blockchains leur ont permis de dépasser le simple cadre financier et de nombreux projets de recherche ont pour but d'exploiter les propriétés de la blockchain dans d'autres secteurs d'applications (Di Francesco Maesa & Mori, 2020). Blockchain signifie littéralement "chaîne de bloc" en français. Par commodité, nous utiliserons le terme anglais. Il s'agit techniquement d'un registre distribué, où chaque utilisateur possède une copie des données. La nature du contenu des transactions, des échanges d'actifs dans le cadre de la cryptomonnaie, est indépendante de la technologie. Le nom provient de la structure de données qui empaquette en bloc les transactions puis les enchaîne avec des liens cryptographiques pour assurer leurs intégrités. En rendant l'intégralité de l'historique des transactions publique et diffusée sur tous les nœuds du réseau. Il devient difficile, voire impossible, de manipuler les données.

La blockchain apporte donc intégrité, transparence et vérifiabilité entre acteurs dont la confiance n'est pas assurée, et sans devoir recourir à un tiers de confiance coûteux dont l'avis peut être biaisé. Grâce à toutes ces propriétés, elle va ainsi trouver des applications dans l'industrie pour réaliser du contrôle d'identité (Esposito et al., 2021) ou améliorer la gestion des chaînes d'approvisionnement (Wüst & Gervais, 2017). Des blockchains spécialisées ont donc vu le jour, avec des systèmes d'authentification plus adaptés et des protocoles prenant en compte un nombre d'intervenants plus réduit. L'exemple le plus manifeste est le projet Hyperledger qui a pour but de développer des *frameworks* et des blockchains à destination des entreprises et qui a reçu la contribution de IBM, Intel et la fondation Linux (Androulaki et al., 2018).

De plus, les dernières innovations permettent d'aller au-delà du stockage de données et donnent lieu à considérer la blockchain comme un véritable ordinateur distribué grâce à la définition des *smart contracts*. Avec cet outil, des extraits de code informatique peuvent être écrits dans la chaîne pour être exécutés et vérifiés par chaque membre du réseau lors de l'ajout du bloc. En utilisant un langage informatique, des concepts logiques plus avancés que de simples échanges peuvent être définis, comme des clauses contractuelles. L'exécution d'un contrat ne génère donc plus de confusions et d'ambiguïtés, car la confiance est ainsi assurée par l'informatique. Dès lors, la blockchain offre l'opportunité de bâtir des organisations décentralisées complexes pouvant remplacer des organisations plus avancées comme les banques, assurances (Kar & Navin, 2021) et autres plateformes d'échanges (Subramanian, 2017) dans de petites et grandes structures.

Cependant, la difficulté de conception de la blockchain correspond à sa capacité à définir l'unicité du registre de données sur un réseau de taille conséquente qui est assuré par l'innovation majeure et centrale de la blockchain : le consensus. Celui-ci a un fort impact sur la rapidité d'exécution des transactions et représente souvent le goulot d'étranglement de la blockchain. Des collections entières de protocoles existent pour mettre d'accord les différents membres du réseau de la blockchain sur quel bloc sera ajouté à la chaîne. Chaque protocole possède ses propres spécificités, mais ils peuvent être classés selon deux familles (basées sur une preuve ou des votes), chacune possédant des avantages et problématiques miroirs l'une de l'autre. Des résultats théoriques ont prouvé l'impossibilité de trouver un consensus unique répondant à toutes les problématiques sans inconvénient (Angelis et al., 2018). Le consensus, parmi d'autres briques

logiciel, doit être en conséquence choisi pertinemment par rapport à l'environnement d'exécution afin de satisfaire les performances attendues lors de la conception. Il convient par conséquent de connaître précisément les liens entre les différents paramètres de la chaîne, leurs impacts sur la performance du système et de l'application, et ce, afin d'adapter les paramètres en vue d'adapter le consensus à l'environnement (Schäffer et al., 2019).

Cependant, l'environnement d'exécution peut être amené à évoluer au cours du temps. Par conséquent, des blockchains réglées pour fonctionner dans un environnement donné de manière optimale peuvent voir leurs performances diminuer significativement en cas de modifications notables de l'environnement. C'est le cas lors de l'arrivée soudaine de nœuds dans le réseau, venant perturber le consensus, voir menacer sa sécurité si ces nœuds sont associés à un unique malveillant. La chaîne risque dans ce dernier cas une attaque Sybil (Douceur, 2002). Fort heureusement, des boucles de régulations existent dans certains de ces consensus pour faire face à ces situations, comme la difficulté dans les chaînes sur la preuve de travail (Fullmer & Morse, 2018). Toutefois, ces boucles concernent essentiellement la sécurité et rarement les performances. L'essentiel des adaptations de blockchain pour améliorer le débit et la latence concernent l'environnement dans lequel s'inscrit la blockchain ou le routage entre les nœuds, peu d'adaptations sont réalisées directement sur le consensus (Miglani & Kumar, 2021).

Cette thèse s'inscrit dans l'établissement de nouvelles boucles de régulations pour améliorer les performances des blockchains. Ces dernières introduisent une latence non négligeable ($> 100ms$) qui peut s'avérer critique en cas de perturbations réseau notamment dans le cadre d'application de type Internet des Objets (ou *IoT*, *Internet of Things* en langue anglophone) (Schulz et al., 2017). Notre constat est que les systèmes de régulation des blockchains ne prennent que peu en compte les attentes sur les performances du système lors des décisions. Si elles sont prises en compte, ce n'est uniquement que durant les choix de conceptions, qui ne sont pas amenés à évoluer alors que l'environnement d'exécution peut être amené à diverger. Nous estimons donc que ces décisions doivent être dynamiques et s'adapter en temps réel aux attentes des utilisateurs pour lui fournir la meilleure expérience possible.

L'étude des performances en ligne de la chaîne est donc un des points de départ de ce travail de recherche. Il s'agit de distinguer l'impact du consensus sur les performances des autres briques constitutives de la chaîne. Dès lors, des modifications aux protocoles BFT seront proposées afin de faire varier les validateurs à la fois en nombre, mais également en qualités en temps réel, pour atteindre de meilleures performances, tout en satisfaisant les demandes de l'utilisateur. L'aspect sécurité n'est en revanche pas à alléger et ne doit pas être inférieur aux attentes minimales de l'utilisateur. Elle doit pouvoir au contraire être renforcée si le système n'est pas sous tension durant les phases avec une faible demande. D'autres impacts doivent également être pris en compte, par exemple, si les modifications du consensus réduisent la décentralisation des blockchains.

Cette thèse se décompose en cinq chapitres principaux :

— La blockchain est une nouvelle technologie de stockage issue de la réunion des domaines cryptographiques avec les systèmes distribués. Le chapitre 1 revient sur les principes théoriques généraux de ces deux domaines et comment leurs unions a défini la technologie blockchain. Des cas d'utilisations justifiées de la blockchain sont introduits. Cette structure de données est ensuite décomposée en plusieurs briques fonctionnelles qui sont détaillées.

— Le chapitre 2 revient sur les mesures de performances des blockchains. Un état de l'art sur les méthodes de mesure de performances des blockchains sur divers consensus est réalisé afin de déterminer les règles génériques de la mise en place d'un banc d'expérience ciblant

efficacement les composants de la chaîne. Par la suite, les différentes propositions existantes d'améliorations de performances des blockchains présentés dans la littérature sont évoquées.

— À partir de l'analyse des mesures de performances des blockchains, une méthodologie d'évaluation d'un environnement est réalisée dans le chapitre 3, prenant en compte les différents liens liant les choix de conception et l'environnement d'exécution. La notion de *capacité* d'une chaîne est alors introduite, correspondant au débit maximum atteignable par celle-ci. Enfin, ce chapitre est conclu par l'étude d'une boucle de régulation particulière en preuve de travail permettant de s'abstraire de l'environnement d'exécution.

— Le chapitre 4 introduit la contribution majeure de cette thèse : l'algorithme Sabine qui permet d'adapter le nombre de validateurs composant le réseau blockchain (BFT) en fonction de la demande en termes de transactions. Les différentes étapes de Sabine sont décrites, de la construction de son modèle par *Machine Learning*, à sa méthode d'action pour conserver la cohérence du pool de validateur au sein du réseau.

— Après avoir déterminé le nombre idéal de validateurs avec Sabine, le chapitre 5 introduit deux méthodes de sélection du pool de validateurs dans le but d'améliorer le débit de génération de bloc d'une blockchain BFT. L'évaluation de ces sélections se fait en prenant en compte l'impact du calcul sur le consensus et sur la décentralisation de la blockchain.

Enfin, cette thèse se conclura sur les différentes perspectives des protocoles développés et sur la prise en compte des performances lors des consensus.

1

Des systèmes distribués à la blockchain

Cette thèse se place dans le contexte des blockchains. Les blockchains sont une technologie de stockage et de partage de données sans autorité de contrôle autre que les membres du réseau associé. Afin de pouvoir mesurer et améliorer les performances de cette technologie, il convient tout d'abord de comprendre les notions sur lesquelles se repose cette technologie. Ce premier chapitre détaille ainsi les notions élémentaires qui seront exploitées par la suite dans les différents chapitres de cette thèse.

Dans un premier temps, sécurité et intégrité des données sont des propriétés au cœur de la blockchain, imposées dès la conception d'une blockchain. Ainsi, la blockchain est une technologie dite *secure by design*. À ce titre, elle fait intervenir des éléments cryptographiques qui seront détaillés dans la section 1.1.

Dans un second temps, les blockchains appartiennent à la famille des registres distribués. Il s'agit d'une famille de systèmes permettant de partager et de synchroniser des informations sur plusieurs hôtes géographiquement distants. Les blockchains s'intègrent au sein de cette famille en recopiant l'entièreté du registre sur tous les utilisateurs du réseau. Des problématiques de synchronisation et de cohérence typiques des systèmes distribués interviennent donc. Ces problématiques seront exposées dans la section 1.2.

La réunion de ces deux disciplines permet de composer un ensemble de bases de données sécurisées dont fait partie la blockchain. Il existe donc un certain nombre d'alternative à la blockchain, parfois plus judicieuse en fonction des situations. Ces alternatives sont listées en section 1.3.

Une fois ces prérequis assimilés, nous allons pouvoir présenter en section 1.4 l'articulation d'une blockchain, de sa structure de données aux différents niveaux d'abstractions que l'ont peut définir.

Le schéma 1.1 résume le sommaire de cette partie, la cryptographie, traitée en section 1.1, et les systèmes distribués, traités en section1.2, concourent conjointement aux problématiques des bases de données sécurisées, traitées notamment en section 1.3. Un des sous-domaines est la blockchain, traitée en section 1.4.

1.1 Notions de cryptographie

La cryptographie est une discipline très ancienne visant à protéger des messages. Initialement, cette discipline consistait simplement à dissimuler l'information. Le mot *cryptographie* vient ainsi du grec $\kappa\rho\upsilon\pi\tau o'\varsigma$, *caché*, et $\gamma\rho\alpha'\varphi\epsilon\iota\upsilon$, *écrire*. Les premiers systèmes cryptographiques cachaient

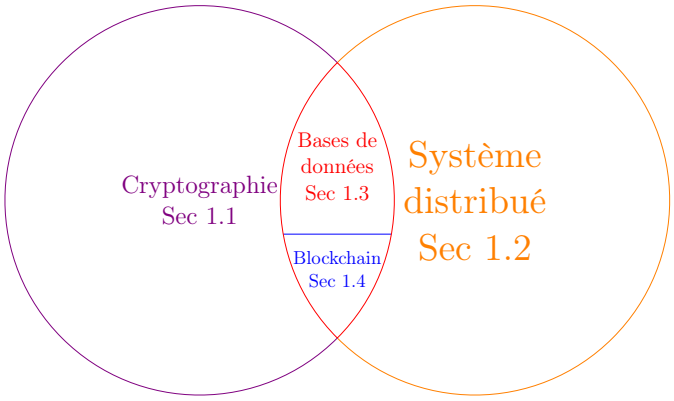


FIGURE 1.1 – Plan du chapitre

les informations par substitution (en changeant un caractère pour un autre) ou par transposition (en changeant l'ordre des caractères dans un message). Cette science, et son pendant opposé, la cryptanalyse, se sont étoffés aux fils des siècles en exploitant les découvertes mathématiques et technologiques pour permettre d'assurer plusieurs autres propriétés. Nous nous focaliserons ici sur la présentation des différentes propriétés cryptographiques exploitées au sein des blockchains à savoir l'intégrité, la confidentialité et l'authentification des messages, ainsi que de certains outils informatiques permettant d'assurer ces propriétés.

1.1.1 Fonction de hachage

Les fonctions de hachage (*hash functions* en anglais) forment une des briques élémentaires de la cryptographie. Ces fonctions permettent d'authentifier rapidement et avec peu d'espace mémoire, une donnée de taille arbitraire en lui associant une image de taille fixe, plus petite et pseudoaléatoire appelée *haché*. La fonction est déterministe. Ainsi, une même donnée donnera avec la même fonction de hachage, toujours la même image. Cependant, si la donnée initiale est modifiée alors l'image résultante est différente.

Une fonction de hachage H doit disposer des propriétés suivantes d'après Rogaway et Shrimpton (2004) :

1. *Résistance à la préimage :* une fonction de hachage est à sens unique. Quand une donnée est hachée, il devrait être difficile de retrouver la donnée originale à partir du haché. Pour une valeur de hachage h, il doit être *calculatoirement impossible* de trouver un message m tel que $h = H(m)$
2. *Résistance à la seconde préimage :* il doit être très difficile de trouver un équivalent d'une donnée donnant un même haché. Ainsi, pour un message m_1 donnée, il doit être *calculatoirement impossible* de trouver un message m_2 tel que $m_1 \neq m_2$ et $H(m_1) = H(m_2)$.
3. *Résistant aux collisions :* il est *calculatoirement impossible* de trouver une collision, c'est-à-dire de trouver deux messages différents m_1 et m_2 donnant le même haché, soit $m_1 \neq m_2$ et $H(m_1) = H(m_2)$.

Une fonction de hachage étant une fonction injective, il est impossible d'assurer ces propriétés. On parle alors de *calculatoirement impossible* dans le sens où briser une de ces propriétés nécessite du temps et des ressources considérables. Ainsi, l'algorithme SHA-256 (Dang, 2015) nécessite de réaliser 2^{128} hachés de données différentes pour trouver une collision (attaque des

anniversaires (Patarin & Montreuil, 2006)). Dans le cadre de cette thèse, on se place dans le cadre du *modèle de l'oracle aléatoire*(Bellare & Rogaway, 1993). Dans ce modèle, tous les participants, dont les attaques, ont accès à un oracle qui :

— pour une requête jamais formulée, l'oracle renvoie une chaîne de bits d'une taille donnée choisie uniformément au hasard ;

— pour une requête déjà reçue, l'oracle renvoie la chaîne de bits précédemment associée.

Ce modèle implique donc que les fonctions de hachages sont supposées idéales, sans prendre en compte leurs implémentations. Les propriétés de résistances à la première et seconde préimages sont par conséquent admises.

Les fonctions de hachages sont donc destructives. Il est très difficile de retrouver la donnée source d'un haché. Pour des systèmes permettant de transformer une donnée en un message brouillé, mais dont on peut retrouver la source, il faut se tourner vers le chiffrement.

1.1.2 Chiffrement

Le chiffrement est la transformation d'un message par cryptographie rendant incompréhensible la lecture du message pour toute personne ne disposant pas de la clef. Deux grandes familles de chiffrements coexistent.

Chiffrement symétrique

Le chiffrement symétrique est la forme la plus ancienne de chiffrement. Le message est chiffré et déchiffré avec une même clef. Le message chiffré peut donc être transmis sans crainte d'être lu. Cependant, la sécurité de la communication repose sur la clef de chiffrement, qui doit être transmis aux agents par un canal sécurisé afin que cette clef ne soit pas récupérée par un attaquant.

Chiffrement asymétrique

Contrairement au système précédent, le chiffrement asymétrique se réalise avec une paire de clefs :

— Une clef dite *publique* est utilisée pour chiffrer le message. Cette clef est généralement rendue publique.

— Une clef dite *privée* pour déchiffrer le message. Cette clef est conservée secrète.

La cryptographie asymétrique supprime le problème de transmission sécurisée de la clef, permettant de définir des protocoles de communication n'utilisant pas de canal sécurisé. Cependant, comme le temps de calcul du chiffrement asymétrique est plus long (S. Singh et al., 2013), il est parfois utilisé comme canal de communication sécurisé pour transmettre une clef symétrique lors d'une communication chiffré par chiffrement symétrique.

1.1.3 Authentification

L'authentification est un processus informatique permettant d'assurer qu'un message lu est bien envoyé par l'émetteur du message. Une preuve d'authentification générée par l'émetteur est envoyée en parallèle du message et une fois reçue par le récepteur, celui-ci utilise un algorithme généralement cryptographique pour s'assurer que le message a bien été envoyé par l'émetteur. Pour économiser du temps de calcul et de l'espace, souvent seul le haché du message est authentifié,

garantissant ainsi l'intégrité du message. On peut retrouver les deux méthodes d'authentification détaillées ci-après.

Code d'authentification de message

Dans le cadre de l'utilisation d'un MAC (*Message Authentication Code*) (Boneh & Shoup, 2020), une clef privée est partagée par l'émetteur et le récepteur. L'émetteur génère à partir de son message et de cette clef privée une étiquette appelée MAC grâce à un algorithme de signature. Il envoie ensuite ce message accompagné de cette étiquette au récepteur. Une fois reçu, le récepteur utilise le message, l'étiquette et la clef privée pour s'assurer de l'authentification de l'émetteur grâce à un algorithme de vérification. Ce dernier algorithme peut-être le même algorithme de signature. Le récepteur essaye alors de régénérer une étiquette avec le message et la clef privée, et si le résultat correspond à l'étiquette reçue, alors l'authenticité du récepteur est prouvée. À l'instar du chiffrement symétrique, le MAC se base sur une clef secrète partagée entre deux utilisateurs par un canal sécurisé.

Signature numérique

La signature (Boneh & Shoup, 2020) est un procédé similaire ne requérant pas de canal sécurisé. Elle s'appuie sur un chiffrement asymétrique à la différence que l'on inverse les notions publique et privée. Un émetteur dispose d'une paire de clefs privées/public et diffuse sa clef publique. Lors de la diffusion d'un message, l'émetteur utilise sa clef privée pour chiffrer/signer le haché du message et cette signature obtenue est associée au message. Lorsque le récepteur reçoit le message, il déchiffre la signature avec la clef publique de l'émetteur et vérifie que le résultat correspond au haché du message.

1.2 Notions de systèmes distribués

Andrew Tanenbaum définit comme suit les systèmes distribués dans son livre *Distributed systems : principles and paradigms* (Tanenbaum & Steen, 2007) :

> *A distributed system is a collection of independent computers that appears to its users as a single coherent system.*

Un système distribué consiste ainsi en un ensemble de processus distincts, séparés dans l'espace, qui communiquent entre eux en échangeant des messages afin de poursuivre un but commun. Chacun des processus dans un système distribué est autonome et considérable comme un système unique. À cette autonomie s'ajoute également la potentielle hétérogénéité des processus, puisqu'un système distribué peut faire intervenir des processus avec des ressources et des capacités différentes. La force du système distribué ne vient pas des capacités matérielles, mais de la performance du système et de son adaptabilité sur différents supports. Par la suite, la séparation spatiale évoquée précédemment reste relative, car deux processus sur un même ordinateur peuvent former un système distribué, tant qu'ils sont isolés et ne partagent pas de mémoire commune. Ainsi, avec toutes ces contraintes, le cœur du système distribué est la collaboration entre plusieurs systèmes différents, afin d'apparaître comme un système unique pour tout utilisateur de ce système.

Trois axes motivent la distribution d'un système : le gain de puissance de calcul (*Distributed Computing Systems*), l'amélioration du partage de mémoire (*Distributed Information Systems*) ou l'amélioration de la disponibilité de l'information (*Distributed Pervasive Systems*). La blockchain s'intègre ainsi dans ce deuxième axe en répliquant les données de la chaîne dans chacun des

membres du réseau. Ce processus de réplication, qui est la copie d'une information vers plusieurs hôtes, est un processus courant des systèmes distribués et est associé à diverses propriétés permettant de garantir à des degrés différents la fiabilité, la disponibilité et la tolérance aux pannes des systèmes répliqués. Diverses de ces propriétés sont évoqués dans les sections 1.2.2, 1.2.3 et 1.2.4. Également, des théorèmes nécessaires pour aborder la blockchain seront évoqués en section 1.2.6. Auparavant, en section 1.2.1, nous détaillerons le modèle SMR, utilisé pour modéliser le comportement de blockchain.

1.2.1 Modèle de Duplication de Machines à États

La modélisation la plus courante d'un problème de réplication d'un système distribué est la *Duplication de Machines à États* (*SMR*) (*State Machine Replication*) (Schneider, 1990). Dans ce modèle, chaque nœud est représenté par un automate qui possède un *état*, composé de différentes variables représentant entre autres la base de données. Cet état subit, à la suite de divers évènements, une *transition* vers un autre état. Dans notre approche, les machines à états sont déterministes, impliquant qu'un même évènement donne une transition identique vers un même état, et les transitions sont dites atomiques, c'est-à-dire qu'une transition est soit appliquée complètement, soit pas du tout.

Les évènements à l'origine des transitions sont des *requêtes* initiées par les clients, c'est-à-dire des transactions venant modifier les variables d'états. Dans ce contexte, les clients sont ainsi les utilisateurs, mais également les autres machines à état du système, afin de propager les modifications des variables d'états pour assurer la cohérence (voir section 1.2.2) du système.

Le modèle SMR est particulièrement utile pour masquer l'apparition de fautes puisque si un nœud est considéré en panne, un utilisateur peut trouver un réplica alternatif ayant un état identique. En effet, les nœuds ayant exécuté les mêmes transitions, l'état final de chaque nœud est supposé être identique. Cependant, cet ensemble de machines à état doit être associé à un protocole de réplication assurant que toutes les machines correctes exécutent les transitions dans un ordre similaire et soient dans le même état. Cette notion est la *cohérence*.

1.2.2 Cohérence

La cohérence (*consistency* en anglais) est un terme pouvant avoir plusieurs sens suivant le contexte et le domaine dans lequel il est employé. Ainsi dans le cadre des bases de données, la cohérence est définie dans le cadre des propriétés ACID (*Atomicité, Cohérence, Isolation, Durabilité*) comme la conformation d'une transaction aux règles de la base de données. On parle dès lors de *database consistency*. Ainsi, une transaction ne violant aucun des invariants d'une base est dite cohérente (Haerder & Reuter, 1983). Comme la blockchain s'apparente à une base de données, les transactions soumises doivent alors respecter les propriétés ACID, et donc la cohérence de la base de données. Ainsi, dans l'exemple d'une chaîne avec une cryptomonnaie comme Bitcoin, une transaction menant à un wallet avec un déficit de cryptomonnaie n'est pas valide. Les blockchains font aussi appel à une autre définition, associée cette fois-ci au système distribué. C'est cette définition qui sera détaillée par la suite et qui concerne singulièrement les systèmes distribués.

Dans le cas de systèmes répliqués, chaque nœud représente un état de la base de données. Ces états sont mis à jour à l'aide de transactions vers un nouvel état. Comme il s'agit de réplication, les nœuds tendent à converger, en l'absence de nouvelle transaction, vers un état unique. La cohérence est donc la capacité du système d'avoir rapidement cet état unique en réponse à toute modification afin que les données soient identiques en tout point du réseau. D'un point de vue

extérieur, un système cohérent est assimilable à un système unique. On parle alors de *Replica Consistency*. Suivant les besoins, un système incohérent est source d'erreur et peut mener à des conflits.

La définition de la cohérence est complexe et est dépendante de l'application, il n'en existe donc pas une définition unique. Par conséquent, on parlera plutôt de *modèles de cohérence*. Ces différents modèles proposent des contraintes plus ou moins restrictives et assurant une cohérence plus ou moins forte. On dit ainsi qu'un système est en cohérence **forte** si à un instant donné, la lecture d'une donnée est la même sur toutes les copies de cette donnée. À l'inverse, un système en cohérence **faible** nécessite de devoir attendre un délai pour qu'une modification de la donnée soit effective sur toutes les copies de cette donnée.

Le modèle le plus strict est la *cohérence strict* (*Linearizability*) (Herlihy & Wing, 1990). Les transactions ont lieu de manière atomique (la transaction s'effectue complètement ou pas du tout) et un ordre total est défini entre les transactions afin de les exécuter séquentiellement (les transactions ne peuvent pas donc s'exécuter en parallèle). Comme les transactions sont déterministes, l'état final de tout réplica exécutant cette même séquence de transactions est identique.

Ces différents modèles utilisent comme hypothèse initiale l'utilisation de processus corrects, c'est-à-dire des processus répondant toujours et exécutant correctement les transactions. Par conséquent, en cas de panne de l'un des processus, ces modèles sont fortement remis en question, même si diverses implémentations de protocoles, comme le *two-phase commit protocol*, tolèrent des processus défaillants. Cependant, cette hypothèse ne prend pas en compte le cas où il y a la présence de processus malveillant renvoyant volontairement des réponses incohérentes. Ce cas de figure doit être compris dans l'implémentation de ces modèles.

1.2.3 Exactitude

Afin d'assurer la pérennité du système et son exactitude au long de son exécution, le système doit satisfaire un certain nombre de propriétés d'abstractions dans tous les cas possibles. Ces propriétés peuvent en général être classées en deux catégories (Lamport, 1977) : la vivacité et la sûreté.

Sûreté

Une propriété de sûreté (*Safety*) assure que "rien de mal n'arrive". Elle permet ainsi d'assurer que le système ne fait pas d'erreur en toute étape d'exécution. Plus formellement, une propriété de sûreté P_s vérifie qu'à tout instant t, P_s est vérifié. Ainsi, si une erreur est détectée à tout instant, la propriété de sûreté est fausse. Dans le cadre des algorithmes répartis, ce type de propriété assure que chaque réplica répondra correctement comme le ferait un système unique tolérant aux fautes. Un exemple de propriété de sûreté est l'impossibilité d'accéder à un état bloquant, en menant à un état exempt de transitions pour le mener à un autre état.

Les propriétés de sûreté ne sont pas suffisantes pour assurer l'exactitude d'un algorithme, car elles ne prennent pas en compte l'évolution des états au cours du temps. Par exemple, pour satisfaire toute propriété de sûreté, il suffit de bloquer le système dans un état satisfaisant ces propriétés et d'empêcher son évolution dans un autre état. Pour prendre en compte cette évolution, des propriétés de vivacité sont définies.

Vivacité

À l'inverse de la catégorie précédente, une propriété de vivacité (*Liveness*) assure que "quelque chose de bien va éventuellement arriver". Elle assure que pendant une durée (non finie), un évènement arrivera, permettant au système de progresser vers un nouvel état, assurant donc que le système n'est pas bloqué faute d'évènements. Plus formellement, une propriété de vivacité P_v vérifie qu'à tout instant t, la propriété P_v sera éventuellement vérifiée à un temps $t' > t$ (Cachin et al., 2011). Un exemple de propriété de vivacité est "Si une transaction est soumise, alors elle sera éventuellement prise en compte par un automate".

1.2.4 Disponibilité

La disponibilité (*availability*) est la capacité pour un système de répondre à une requête reçue par un nœud non défaillant avec une réponse correcte, sans cependant la garantie que cette réponse corresponde à l'état le plus récent. Cette absence de garantie stricte signifie que si la requête est la lecture d'une variable, la réponse obtenue ne reflète pas l'état le plus récent de cette variable.

Par exemple, un client réalise une requête de lecture d'une variable stockée dans un registre hébergé par un système répliqué. La disponibilité implique que ce client trouve toujours un nœud lui donnant une réponse à sa requête. Cependant, ce client n'a pas la garantie que cette réponse est à jour. Le nœud qui lui a répondu n'a pas forcément reçu les transitions menant à l'état final associé à cette variable . Aucune limite de temps n'est spécifiée entre la requête et la réponse. Dans certaines définitions, ce temps peut être considéré comme sans limite. Cette capacité est assimilable à la terminaison de l'algorithme utilisé par le service distribué (Gilbert & Lynch, 2002).

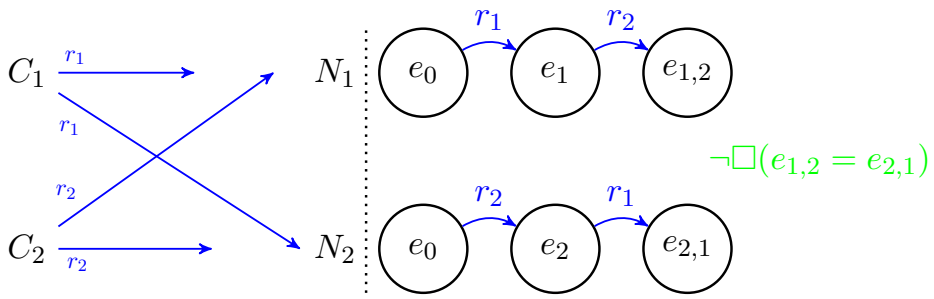
1.2.5 Le problème du consensus

Dans le cas d'un système cohérent où tous les nœuds sont dans le même état, plusieurs transitions sont possibles. Ces nœuds sont sollicités par différents clients à l'aide de requêtes correspondantes chacune à un évènement menant à différentes transitions possibles. Le réseau imposant des délais différents entre chaque nœud et client, les requêtes ne sont pas toujours reçues dans l'ordre par chaque nœud, si par ailleurs elles ont été adressées à chaque nœud. Dans le cas où le système est décentralisé et où les nœuds agiraient indépendamment de leurs réplicas, le système pourrait être composé d'une multitude de nœuds dans des états différents, comme dans l'exemple de la figure 1.2.

Afin de garder le système cohérent, les nœuds doivent ainsi se coordonner pour sélectionner un état unique à répliquer sur l'ensemble du système. C'est ce que l'on réfère comme étant le "problème du consensus". Par ailleurs, si le système applique un modèle de cohérence stricte, ce problème de consensus revient à définir un ordre d'exécution des requêtes correspondant à l'ordre de franchissement des transitions.

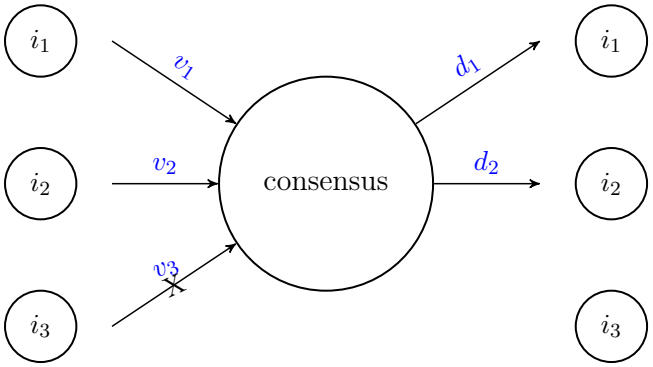
Une version simplifiée de ce problème, tel que défini dans Coulouris et al. (2011), est de supposer un ensemble de n processus qui doivent décider d'une même valeur. La figure 1.3 représente ce modèle. Chaque processus i possède une valeur v_i appartenant à un ensemble D. L'objectif est de trouver un algorithme où les différents processus se mettent d'accord sur une valeur commune en communiquant entre eux. À la fin de l'exécution de l'algorithme, chaque processus p_i possède une valeur de décision d_i résultante de cet algorithme.

Un tel algorithme doit répondre à un certain nombre d'exigences :



Les nœuds N_1 et N_2 commencent ensemble dans le même état e_0. Les clients C_1 et C_2 envoient chacun une requête r_1 et r_2 au même moment. Dues aux différences de temps de trajets, elles ne sont pas reçues dans le même ordre. Les nœuds exécutent ensuite les transactions dans l'ordre de réception, menant à deux états finaux $e_{1,2}$ et $e_{2,1}$ non forcément identiques.

FIGURE 1.2 – Exemple de désynchronisation dans deux réplicas sans consensus



Les processus i_1, i_2 et i_3 essayent de prendre la décision sur une valeur. Chacun possède une valeur $v \in D$ et cherche à déterminer une valeur $d \in D$ identique aux autres processus grâce à un consensus. Dans l'exemple, le processus i_3 est incorrect et n'a pas envoyé de message.

FIGURE 1.3 – Exemple de consensus générique entre trois processus

— *Terminaison* : Tous les processus corrects doivent éventuellement avoir une valeur de décision ;

— *Accord* : La valeur de décision des processus correcte est la même. Pour tous les processus différents et corrects i et j, leurs valeurs de décision sont identiques : $d_i = d_j$.

— *Intégrité* : Si les processus corrects proposent tous une même valeur v, alors tous les processus corrects, à la suite de l'algorithme, doivent avoir comment valeur de décision $d_i = v$.

L'exigence d'intégrité peut varier en fonction de l'application. Les exigences d'accord et d'intégrité correspondent à des propriétés de sûreté, celle de terminaison à une propriété de vivacité.

Les algorithmes répondant aux problèmes du consensus sont ainsi des systèmes permettant d'assurer la cohérence au sein d'un système répliqué. Ce sont des réponses plus résilientes à des processus ne supposant que des processus corrects comme les protocoles de *atomic commit* (ajout atomique) de transaction, tel le *two-phase commit protocol* (Coulouris et al., 2011), pour assurer la cohérence au sein d'un système. En effet, les exigences d'accord et d'intégrité des algorithmes de consensus tolèrent la présence de nœuds défaillants au sein du réseau, ce qui permet de parvenir à un accord bien qu'un certain nombre de nœuds puissent ne pas participer au consensus du fait d'un isolement par un partitionnement du réseau ou d'une panne. On parle

alors de consensus CFT (*Crash Fault Tolerant*). Cette tolérance aux nœuds défaillants peut être étendue jusqu'à englober des nœuds malhonnêtes venant perturber le consensus. Ces nœuds sont appelés byzantin et seront abordés plus en détail dans la section 1.2.6. Dans ce cas, l'algorithme est appelé BFT pour *Byzantine Fault Tolerant*.

1.2.6 Principaux théorèmes

La mise en place d'une base de données répartie répondant aux diverses définitions détaillées précédemment est limitée par des contraintes formalisées par un ensemble de théorèmes. La compréhension de ces théorèmes est donc nécessaire pour pouvoir définir une blockchain puisqu'elle sera également limitée par ces théorèmes.

Un premier théorème fondamental est le théorème CAP comme décrit dans la section 1.2.6. Il distingue les systèmes répliqués disponibles et cohérents. Ce théorème possède de forts impacts sur le choix des stratégies des consensus qui seront exploités par la suite. En effet, l'un des objectifs d'un consensus imposant une cohérence forte est de pouvoir fonctionner tout en tolérant parmi ses membres une fraction d'utilisateurs malveillants. Le problème des généraux byzantins, décrit en section 1.2.6, permet de donner une fraction maximale d'utilisateurs malveillants.

Théorème de Brewer (CAP)

Le théorème CAP ou théorème de Brewer est une conjecture énoncée par Éric Brewer (Brewer, 2000) puis prouvée par la suite par Seth Gilbert et Nancy Lynch (Gilbert & Lynch, 2002). Ce théorème défini qu'il est impossible pour un système distribué de garantir en même temps les trois contraintes suivantes, associées aux lettres "CAP" :

— **Cohérence** (***C****onsistency*) : Chaque demande de lecture retourne la dernière écriture la plus récente. Il s'agit de la propriété décrite en section 1.2.2.

— **Disponibilité** (***A****vailability*) : Le système répond aux requêtes qui lui sont soumises. Il s'agit de la propriété décrite en section 1.2.4.

— **Tolérance au partitionnement** (***P****artition Tolerance*) : Le système peut continuer à fonctionner bien qu'un nombre arbitraire de nœuds soient perdus ou retardés.

D'après ce théorème, on ne peut donc disposer que de deux contraintes. Dans le cas d'un système réparti CFT (*Craft Fault Tolerant*), donc tolérant aux pannes de certains nœuds, le système garantit la propriété P, c'est-à-dire la tolérance au partitionnement. Dans ce cas, le théorème impose donc que l'on choisisse entre un système AP (Disponibilité - Tolérance au partitionnement) et un système CP (Cohérence - Tolérance au partitionnement). Les consensus de blockchain se divisent par conséquent en deux catégories, suivant que le système privilégie la réponse aux requêtes ou à une base de données cohérente (Angelis et al., 2018).

Théorème des généraux byzantins

Le problème des généraux byzantin est une métaphore énoncée par Lamport et al. (1982). Elle traite de la difficulté de définir un système informatique fiable dans le cas où les différentes parties du système distribué recevraient des informations conflictuelles. Il s'agit donc d'un problème de consensus. À travers cette métaphore, des conditions sur le nombre maximal de nœuds malveillants tolérables sont définies pour établir un algorithme BFT (*Byzantine Fault Tolerant*).

Le problème Cette métaphore fait écho à une ville assiégée par un ensemble de généraux de l'armée byzantine. Les généraux ne communiquent l'un à l'autre que par messager et doivent parvenir ensemble à un accord sur le plan de bataille. Cependant, un ou plusieurs traîtres sont présents parmi les généraux et cherchent à perturber le plan de bataille. Pour cela, les généraux doivent avoir un algorithme garantissant :

1. Tous les généraux loyaux décident du même plan d'action ; c'est une exigence d'accord.
2. Un sous-ensemble de traîtres ne peut forcer les généraux loyaux d'adopter un mauvais plan. Cette contrainte spécifie que le consensus doit tolérer des malveillants.

Le rôle des traîtres n'est pas obligatoirement d'imposer un mauvais plan de bataille. Ce cas de figure nécessite qu'ils soient bien plus nombreux que les généraux loyaux. Leur but n'est que d'enfreindre au moins une des garanties précédentes. La métaphore distingue également si les messages envoyés sont oraux ou écrits. Dans les deux cas, le modèle implique que les malveillants ne peuvent interférer avec les communications entre processus, en subtilisant ou corrompant les messages. La différence repose sur la possibilité d'authentifier le message. Dans la version orale, seul le destinataire peut authentifier un message alors que dans la version écrite, les messages sont signés et authentifiable par tous les généraux. Cette principale différence implique que dans la version orale, un général malveillant pourra donner des ordres contradictoires, chose qu'il ne peut faire avec des messages écrits, car les messages écrits le confondraient de par l'authentification.

Solution Il est finalement démontré que dans le cas où les messages seraient oraux, un accord peut être trouvé en tolérant f traîtres si le nombre de généraux m satisfait la relation $m > 3f$ (Lamport et al., 1982).

Dans le cas où les messages seraient écrits, la tolérance aux traîtres est plus renforcée, et le nombre de généraux total doit satisfaire la relation $m > f + 2$.

Cette métaphore est donc analogue au problème de consensus. Les généraux remplacent alors les processus et les traîtres sont des processus dits byzantins. Ces processus byzantins sont des nœuds qui peuvent être défaillants (ils ne répondent plus et s'arrêtent) ou retournent des résultats arbitraires et malveillants. Dans ce dernier cas, les réponses peuvent être incorrectes, différentes en fonction des processus ou délibérément trompeuses. Dans le cadre du problème de consensus, on privilégiera la version orale du problème des généraux byzantins. En effet, la version écrite nécessite une réponse de la part de tous les généraux, chose que l'on ne peut garantir avec des processus byzantins, car ils peuvent être défaillants.

1.3 De la nécessité d'utiliser une blockchain

La blockchain est une technologie certes récente, mais révolutionnaire. Elle a fortement remis en question le secteur bancaire avec les cryptomonnaies menant à un fort engouement autour de cette technologie. L'un des exemples de cet engouement est que le cours du Bitcoin a frôlé, à certains moments, les $60,000$ € (Coinbase.com, 2022). Cette technologie a donc fait écho dans le secteur bancaire et d'autres domaines ont alors tenté d'exploiter les avantages de cette technologie à travers de nombreux projets de recherches sur des secteurs autres que la monnaie, tels que l'énergie, la finance et la santé. Ces différentes applications seront davantage traitées au cours de la section 1.4.6. Les avantages sont en effet nombreux, peu d'autres technologies offrent l'immutabilité et le partage de données entre utilisateurs dont la confiance n'est pas avérée.

Le potentiel de cette technologie est donc conséquent, mais l'engouement pour les blockchains et la méconnaissance des concepts associés peuvent mener parfois à des utilisations mal-

avisées (Adams, 2019). Dans l'absolu, une blockchain basée sur une preuve de travail, ne communiquant avec aucun autre nœud, pourrait être mise en place afin de servir de base de données pour contenir cette thèse. Ce cas est caricatural, mais d'autres exemples plus réels peuvent s'avérer malavisés. Par exemple, la décentralisation permise par la blockchain entraîne une perte de contrôle sur l'application de la part des concepteurs, rendant celle-ci plus vulnérable en cas d'attaque basée sur la mauvaise conception de l'application. Ce phénomène est particulièrement vrai pour les *Smart-Contract* (notion qui sera définie en section 1.4.5) où il n'est pas possible de corriger une erreur après la publication du contrat. Parity Multisig Wallet (Technologies, 2017) et King of the Ether Throne (KotET, 2016) sont deux exemples de *smart contract* bloqués à la suite d'une étape non prévue. Une analyse formelle peut prévenir ce genre de situation (Tolmach et al., 2021) mais ce sont des compétences supplémentaires à mobiliser et qui font encore l'objet de recherches.

De surcroît, la sécurité offerte par la blockchain a forcément un coût, notamment pour le consensus. Dans le cas d'un consensus basé sur des élections, ce coût se répercutera sur la bande passante. En effet, comme les nœuds communiquent pour trouver un accord à travers des messages envoyés sur le réseau, ces messages peuvent saturer le réseau et dégrader les performances d'autres applications. Dans le cas d'une preuve de travail, le coût du consensus est énergétique. Il est ainsi estimé qu'en 2018, le réseau Bitcoin avait une consommation équivalente à celle de l'Irlande (de Vries, 2018). La preuve de travail tend à être remplacée, notamment par la PoS (*Proof-of-Stake* ou preuve d'enjeux) dans le cas de Ethereum 2.0, mais ce remplacement amène d'autres questions propres aux fondations mêmes de la définition d'une blockchain et du consensus.

Face à ces coûts, la nécessité de la blockchain est remise en question, aux profits d'alternatives plus classiques, mais plus fiables et moins coûteuses. De nombreux modèles ont été définis afin de trouver les alternatives d'une blockchain et pour orienter la sélection de la blockchain. Koens et Poll listent et comparent 30 modèles différents dans la littérature (Koens & Poll, 2018) et concluent que des alternatives aux blockchains existent, arguant que celles-ci peuvent parfois être ignorées dans certains projets. Ils mettent également en évidence une confusion au sujet du terme "blockchain" dans la littérature. Cependant, dans des cas très spécifiques, la blockchain conserve un intérêt réel.

Cet intérêt se dégage dans l'un des modèles les plus complets de la littérature concernant la nécessité d'utiliser une blockchain, qui est présenté en figure 1.4 et provient du United States Department of Homeland Security (Yaga et al., 2018). Ce modèle pose successivement des questions sur les exigences demandées par l'application et fournit des alternatives dans le cas où toutes ces exigences ne seraient pas satisfaites. Ces différentes questions interrogent sur la nécessité d'avoir les propriétés suivantes :

1. Système répliqué avec une cohérence stricte : comme vu dans la section 1.2.2, la cohérence stricte est le plus haut modèle de cohérence. Son implémentation impose des performances réduites qui impacteront le système. D'autres modèles moins restrictifs existent et se déploient avec de meilleures performances ;
2. Décentralisation : Initialement, le premier projet de blockchain, Bitcoin (Nakamoto, 2008), a été imaginé pour créer un réseau bancaire où chaque membre est acteur. Si un seul membre est acteur, des systèmes centralisés peuvent mieux convenir ;
3. Auditable : en remontant la chaîne de bloc, il est possible de reconstituer un historique de la chaîne et des transactions qui ont été soumises, fournissant une grande vérifiabilité des informations ;

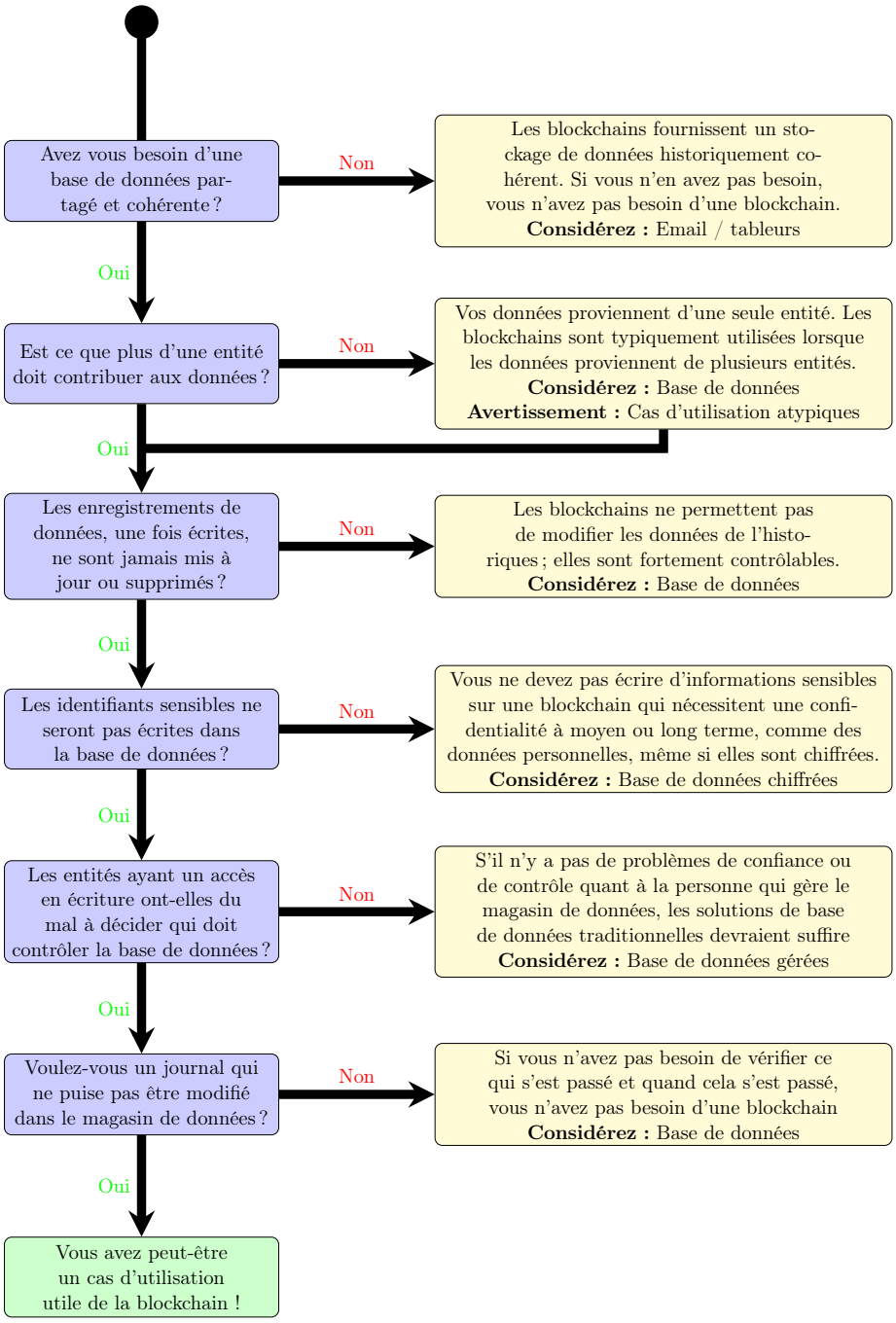


FIGURE 1.4 – Organigramme du *DHS Science & Technology Directorate* pour la proposition d'alternative à la blockchain (Yaga et al., 2018)

4. Transparence : pour pouvoir assurer la conformité des données, les transactions sont vérifiées. Pour cela, les transactions sont donc publiques et vérifiables par tous les utilisateurs ;
5. Pas de tiers de confiance : La blockchain permet de faire collaborer des acteurs sans confiance avérée, le contrôle des données se fait donc par tous les utilisateurs afin d'éviter l'apparition d'un tiers de confiance ;
6. Journalisation pérenne : par sa structure, une blockchain est un immense journal de modification des données, dont l'interprétation donne la base de données courante. Ce journal ne peut être modifié sans corrompre l'intégrité de la structure.

Si un projet nécessite l'ensemble des propriétés décrites, alors l'utilisation d'une blockchain s'avère judicieuse. Dans le cas contraire, des solutions comme des bases de données, simples, chiffrées et/ou gérées conviendront tout aussi bien, voire mieux. Une fois la décision d'utiliser une blockchain actée, il est nécessaire de décider quel doit être le type de blockchain utilisé, et notamment, quel sera le modèle d'accessibilité et de gestion des permissions de la chaîne. Ces décisions impactent l'architecture de la chaîne et le consensus associé. Là également, des modèles existent afin d'aider dans cette décision telle le modèle de Birch-Brown-Parulava (Birch et al., 2016).

1.4 La blockchain dans le détail

Textuellement, le terme *blockchain* se traduit par "chaîne de blocs" et ne décrit qu'une structure de données consistant à empaqueter des informations dans des blocs et à les chaîner ensemble par cryptographie, assurant de cette manière l'intégrité des données. Cependant, au fur et à mesure de la conception des blockchains, le concept de cette technologie s'est étoffé et complexifié par l'ajout de fonctionnalités et d'applications. La technologie a ainsi dépassé sa structure initiale.

Le terme *DLT* est parfois préféré pour évoquer les blockchains, même s'il n'est pas tout à fait équivalent. Il correspond à l'étape suivante de la blockchain. Les DLT (*Distributed Ledger Technologies*) sont des registres sécurisés et distribués entre plusieurs utilisateurs. Ils se rapprochent donc des systèmes répliqués évoqués en section 1.2 puisqu'ils consistent à répliquer, partager et synchroniser une base de données. DLT et blockchains ne sont pas équivalents, car les DLT sont une famille englobant d'autres technologies comme les DAG (*Directed Acyclic Graph*, une généralisation des blockchains aux graphes orientés acycliques au lieu d'une liste chaînée) (Divya & Biradar, 2018) ou les Hashgraph (Baird et al., 2018). Par cette association, les blockchains ajoutent un aspect consensus et réseau de partage qui n'existait pas avec la structure de données seule.

Une fois obtenu ce registre distribué, les nœuds sont incités à participer à la chaîne, pour assurer sa pérennité, bien que des permissions soient rajoutées pour restreindre l'accès en lecture et écriture. Une fois la chaîne partagée au plus grand nombre, des algorithmes permettant de manipuler les données au sein de la chaîne sont développés afin d'assurer la vérifiabilité et la sécurité des opérations effectuées. Ces deux étapes permettent d'ajouter des couches d'intelligence à la blockchain, lui permettant de quitter son rôle de simple base de données.

Enfin, cet outil désormais assimilable à un ordinateur distribué peut être intégré dans des écosystèmes divers, permet de l'utiliser comme outil central pour coordonner ou certifier certaines étapes. Des applications utilisant la blockchain peuvent alors être déployées dans de multiples environnements.

Au travers de cette évolution du terme blockchain, on remarque que sa structure est complexe, et peut être vu comme un modèle en couches de 6 niveaux, tel que représenté dans la figure 1.5. Ce

modèle se base sur celui proposé par Yuan et Wang (2018), à la différence que la couche *Incitation* est remplacée par une couche *Permission*. Dans ce modèle, chaque décision de conception d'une des couches impacte fortement la conception des couches adjacentes. Les 6 couches du modèle présenté dans la figure 1.5 sont discutées dans les sections suivantes.

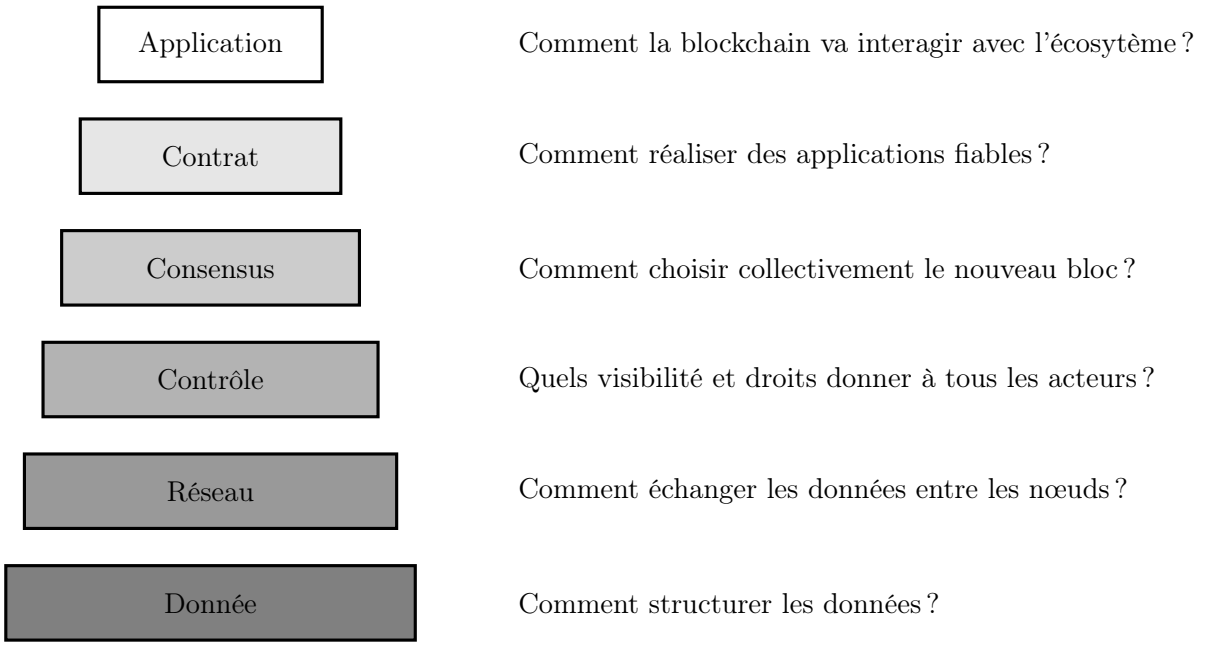


FIGURE 1.5 – Pile d'architecture de blockchain et questions de recherche associées (Yuan & Wang, 2018)

1.4.1 Couche donnée

Comme indiqué précédemment, la blockchain signifie *chaîne de blocs*. Ce chaînage est réalisé par cryptographie à l'aide des fonctions de hachage évoquées en section 1.1.1. La première évocation dans la littérature de ce type de chaînage provient de Haber et Stornetta (1991).

Dans une blockchain, toutes les données sont empaquetées dans un ensemble de blocs ordonnés. Chaque nouvel ensemble de données, chaque nouveau bloc est ajouté à la suite du dernier bloc de la chaîne via une procédure appelée le chaînage, laquelle rend très difficile la modification d'un bloc. Il s'agit donc d'une structure *append-only*, contrairement à une base de données classique. Une fois ajouté, un bloc peut très difficilement être modifié grâce au chaînage, qui se déroule de la façon suivante : Dans chaque bloc, un ensemble de champs forme un en-tête et est laissé disponible pour insérer des informations utiles à la blockchain. Deux champs sont au minimum présents dans l'en-tête : la haché du bloc précédent (ou père) et le haché de l'intégralité du bloc courant suite à d'une fonction de hachage. La figure 1.6 représente un ensemble de blocs chaîné ensemble avec un champ supplémentaire, l'identifiant du bloc.

Ce chaînage garantit l'intégrité puisque si une information au sein d'un bloc au milieu d'une chaîne est modifiée, ajoutée ou supprimée, le haché de ce bloc ne correspond plus au "nouveau" haché du bloc. Comme la propriété de *résistance à la seconde préimage* (cf. section 1.1.1) rend presque nulle la probabilité de trouver une donnée équivalente, le haché du bloc courant doit être remplacé. Pour conserver le chaînage des blocs, le haché père du bloc fils doit être remplacé. Or, comme le bloc fils est modifié, son haché doit également être recalculé. Une modification d'une

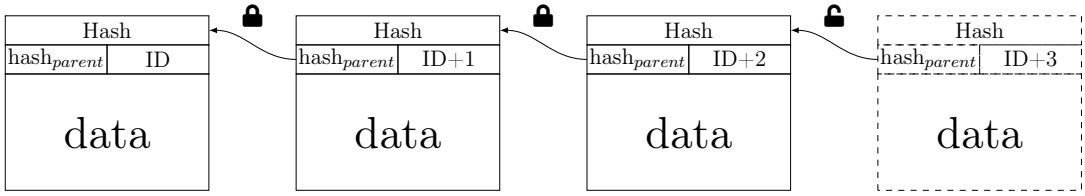


FIGURE 1.6 – Schéma d'une blockchain

donnée dans un bloc implique donc la modification de tous les blocs suivants. Comme les blocs sont répliqués sur un ensemble de nœuds, la comparaison d'un bloc d'un même rang permet donc d'identifier les nœuds possédant des blocs corrompus.

En fonction des implémentations, d'autres champs sont généralement prévus dans l'en-tête : le numéro du bloc, l'identité de celui qui a créé le bloc, la racine d'un arbre de Merkle (structure de données concaténant des hachés de plusieurs transactions et permettant de séparer efficacement le contenu du bloc de l'en-tête) (Merkle, 1988), preuve d'un consensus , etc.

L'agrégation des transactions en blocs, puis l'ordonnancement de ces blocs, aide à définir un ordre total parmi les transactions. Cette organisation permet d'imposer une lecture séquentielle des transactions de la blockchain dans un ordre unique et sans parallélisme puisque les transactions sont lues une à la fois. Chaque transaction est de cette façon exécutée en isolation totale l'une de l'autre, dans la définition des normes ACID. Cette propriété, aussi appelé sérialisation (*serialization* en anglais) est l'une des conditions nécessaires pour s'approcher d'un modèle de cohérence stricte.

Il est à noter que cette agrégation a un impact sur les performances. En effet, les blocs sont généralement de taille fixe (soit en termes de nombre de transactions, soit en termes de taille mémoire totale de transactions) et le débit de génération de bloc est relativement fixe dans un environnement donné. Donc, la taille du bloc est théoriquement directement linéaire avec le débit des transactions ajoutées.

1.4.2 Couche réseau

La blockchain forme un système distribué parmi un ensemble de nœuds qui collaborent afin de pouvoir ajouter de nouveaux blocs. Des messages sont donc diffusés entre les nœuds pour assurer le fonctionnement du consensus afin d'ajouter de nouveaux blocs. Pour diffuser efficacement ces messages, les blockchains s'appuient sur des réseaux pair-à-pair (*peer-to-peer* ou *P2P*) notamment pour garder une architecture décentralisée. En effet, à l'inverse d'un modèle *client-serveur* centralisant toutes les communications autour d'un ensemble de serveurs, dans le cadre d'une architecture pair-à-pair, chaque nœud est à la fois client et serveur, permettant un échange direct de données entre les nœuds.

Dans un réseau pair-à-pair, un lien entre deux n'est pas nécessairement directe, mais correspond à une connexion, par exemple TCP/IP, avec un protocole d'échange dédié. Un réseau pair-à-pair s'inscrit donc par-dessus un réseau existant, et créé une couche d'abstraction ne retenant que les capacités d'échanges entre les nœuds.

Deux types d'échanges se distinguent :

— l'échange *unicast* : un message est envoyé entre deux nœuds identifiés ;

— le *broadcast* : un message est envoyé par un nœud à l'ensemble des autres nœuds. Dans le cas de cette thèse, cette diffusion s'appuiera techniquement sur l'émission identique de messages en unicast à chacun des nœuds.

Les blockchains utilisent les réseaux pair-à-pair pour partager avec l'ensemble des nœuds les transactions proposées et les nouveaux blocs à ajouter. Des algorithmes de propagations d'informations, aussi appelés *protocoles de bavardage*(*gossip protocol* en anglais) (Demers et al., 1987), sont utilisés et s'appuient sur la topologie du réseau pair-à-pair afin d'optimiser la propagation de l'information, particulièrement dans le cas des messages en *broadcast*. Durant une propagation, les nœuds recevant un message à diffuser propagent ce message à leurs voisins, lesquels procèdent à la même opération jusqu'à ce que l'ensemble du réseau possède cette information.

La topologie est donc primordiale pour réduire le délai nécessaire à la diffusion d'un message à tout nœud. En effet, la réduction de la latence de diffusion dans une blockchain réduit la latence globale d'un bloc ($t_{latency} = t_{mining} + t_{diffusion}$ où t_{mining} est le temps nécessaire à la réalisation du consensus), réduisant également la probabilité de *fork* dans certains consensus. Ces *forks* sont des erreurs issues du consensus qui seront évoquées dans la section 1.4.4. Certains modèles topologiques issus de la théorie des graphes sont présentés dans le tableau 1.1. Le diamètre correspond à la plus grande distance possible qui puisse exister entre deux de ses sommets et le degré le nombre de liens reliant un sommet.

Topologie	**Diamètre**	**Nb Liens**	**Degré**
Linear array	$p-1$	$p-1$	2
Ring	$p/2$	p	2
2-D mesh	$2(\sqrt{p}-1)$	$2\sqrt{p}(\sqrt{p}-1)$	4
Hypercube	$log_2(p)$	$log_2(p) \times (p/2)$	$log_2(p)$
Butterfly	$log_2(p)$	$log_2(p) \times 2p$	4

TABLEAU 1.1 – Comparaison de topologies, p étant le nombre nœuds dans le réseau

Deux métriques principales sont utilisées pour quantifier la qualité d'un réseau : le délai maximum et le nombre de sauts maximum. Le premier est le temps maximum nécessaire pour qu'un message soit diffusé à tous les membres du réseau, et le second, le nombre de routages maximal nécessaire pour qu'un message parvienne à un autre membre du réseau. Le nombre de sauts maximum est l'équivalent du diamètre de la théorie des graphes.

Enfin, un protocole pair-à-pair ne définit pas seulement une topologie. Il définit d'une part un algorithme d'insertion pour un nœud, qui permet au réseau d'adopter une structure particulière, mais il associe également des protocoles d'échanges de données (par exemple, un protocole de bavardage ou d'échange de données). Dans l'exemple d'Ethereum, le protocole RLPx gère l'insertion et le routage des messages tandis que l'*Ethereum Wire Protocol* formalise les messages échangés entre les différents nœuds (Lange, 2015).

Graphe complet

La topologie la plus optimale pour la diffusion est l'adoption d'une topologie assimilable à un *graphe complet* dans la théorie des graphes (Bretto et al., 2012) où chaque nœud est relié à tous les autres, à l'instar du graphe représenté figure 1.7. Le nombre de sauts est donc minimal et est égal à 1.

L'inconvénient de cette topologie est que le nombre de liaisons nécessaire augmente avec le nombre de nœuds, augmentant ainsi le nombre de requêtes et donc la charge sur chaque nœud. Ainsi, si un nœud doit envoyer un message en *broadcast* sur une telle topologie constituée de N nœuds, elle doit envoyer $N-1$ messages. Ce type de topologie est ainsi réservée pour des réseaux

de tailles restreintes (de l'ordre d'une centaine de nœuds). Pour fonctionner correctement dans le cadre d'un réseau d'une taille conséquente, d'autres topologies doivent être utilisées.

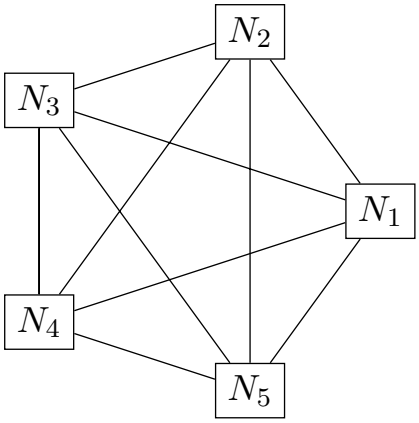


FIGURE 1.7 – Schéma d'une topologie complète composé de 5 nœuds

Topologies intermédiaires

Dans le cadre de réseaux de taille plus conséquente, il est nécessaire de diminuer le nombre de liaisons avec les autres nœuds afin de diminuer la charge. Ce nombre de liaisons est aussi appelé *degré* en théorie des graphes. Cependant, si le degré diminue, le diamètre du graphe augmente nécessairement, risquant ainsi d'augmenter le délai nécessaire pour diffuser un message. Une première solution serait de définir un nœud ou un sous-ensemble de nœuds qui relayeraient l'ensemble des messages au reste des nœuds, mais une telle solution contreviendrait à l'objectif initial de définir un système décentralisé. Cela implique que dans un système idéalement décentralisé, le degré serait identique pour tout nœud.

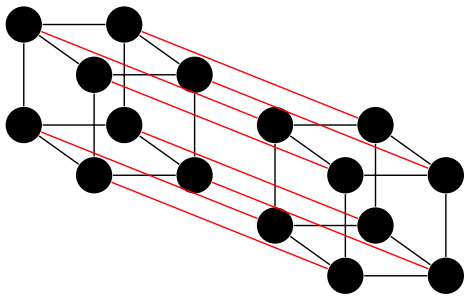


FIGURE 1.8 – Représentation d'un hypercube de dimension 4

Un compromis doit donc être trouvé entre diminution du degré des nœuds et diminution du diamètre du graphe. Parmi toutes les topologies présentées dans le tableau 1.1, le compromis le plus optimal serait assimilable à un hypercube comme représenté à la figure 1.8. Ce compromis garantit, pour un nombre de nœuds p, une diffusion en $log_2(p)$ étapes et assure un degré identique pour tous les nœuds (Gummadi et al., 2003).

Actuellement, des architectures pair-à-pair existent déjà et tentent de répondre à ce besoin : les réseaux structurés et non structurés.

Réseaux Non-Structurés Dans les systèmes non structurés, comme le nom l'indique, l'organisation des nœuds est aléatoire, les informations sont transmises par inondation (*flooding*) (Eng Keong Lua et al., 2005). Ce manque de structure permet une architecture agile et robuste, particulièrement en cas de défaillance. Ce réseau a aussi l'avantage de ne pas émettre beaucoup de messages pour maintenir sa structure. Par la nature aléatoire du réseau, il est difficile de déterminer des lois sur la performance. Cependant, le retour d'utilisation de ces structures montre que les performances (délais, latence...) sont correctes, le degré n'est pas uniforme, mais est majoritairement faible. L'explication de ces performances viendrait du protocole de *bootstrap*, permettant l'insertion des nœuds dans le réseau. La majorité des nouveaux nœuds s'insère par l'intermédiaire d'un petit ensemble de nœuds ou de serveurs, le protocole a par conséquent tendance à attacher ces arrivants avec les mêmes nœuds, créant ainsi un hub de partage d'information. Les informations se diffusent donc rapidement par ce hub, engendrant cependant une centralisation de l'information (Malatras, 2015). Différents protocoles sont utilisés. Bitcoin (Nakamoto, 2008) utilise son propre protocole de bavardage, mais d'autres protocoles existent comme Gnutella (Kan, 2002) ou Freenet (Clarke et al., 2001).

Réseaux Structurés Contrairement aux réseaux précédents, les systèmes structurés imposent des règles pour l'établissement des connexions entre pairs. L'objectif est d'adopter une structure de graphe proche d'un hypercube permettant d'optimiser la répartition de charge entre les nœuds afin de répartir la responsabilité de chaque nœud, de réduire le nombre de requêtes et de réduire la charge de chaque nœud, en limitant le nombre de voisins. Des algorithmes de maintenance sont utilisés pour former et conserver la structure du réseau, lors de l'insertion d'un nœud par exemple. Ces algorithmes ont donc un coût qui se traduit par un certain nombre de messages régulièrement envoyés (Malatras, 2015).

Un ensemble de protocoles structurés est présenté dans le tableau 1.2. Les excellentes performances de ces structures en font des outils de choix pour la diffusion dans une blockchain, à l'instar d'Ethereum et de son protocole RLPx, dérivé du protocole Kademlia (Lange, 2015). La grande part de ces protocoles disposent d'un degré et d'un diamètre proches d'un hypercube et de l'ordre de $O(logN)$, avec N le nombre de nœuds. Comme les règles sont appliquées à tous les nœuds, la décentralisation est optimale et diamètre et degré sont quasiment identiques en tout nœud.

1.4.3 Couche contrôle (d'accès)

Les blockchains sont généralement classés en trois ou quatre catégories en fonction de comment elles gèrent l'inclusion et qu'elles sont les droits des membres à ajouter et vérifier des données dans la chaîne. Cette sélection du type, appelé ici couche *Permission*, impacte fortement les couches *Consensus* et *Contrat* adjacentes. Deux critères sont alors nécessaires pour définir un type : est-ce que la blockchain est privée ou publique, et est-ce que la blockchain est *Permissionless* (sans permission) ou *Permissioned* (avec permission) (Fernández-Caramés & Fraga-Lamas, 2018) ?

Le schéma représenté sur la figure 1.9, et inspiré de Fernández-Caramés et Fraga-Lamas (2018), peut se voir comme la continuation du schéma 1.4, dans le cas où l'utilisation d'une blockchain deviendrait intéressante. Les termes présentés sont encore en débat, et certains auteurs ne font pas de distinction entre privé/permissioned et public/permissionless.

Le premier critère (privé ou public) concerne l'authentification d'un nœud dans la chaîne et de comment un nœud devient membre d'un réseau et prendre part au consensus. Dans une

	Chord	CAN	Pastry	Tapestry	Kademlia	Viceroy	SkipNet
Diamètre	$O(logN)$	$O(d \cdot N^{1/d})$	$O(log_\beta N)$	$O(log_\beta N)$	$O(logN)$	$O(logN)$	$O(logN)$
Nombre de message entré/sortie	$log^2 N$	$2d$	$log_\beta N$	$log_\beta N$	$logN$	$logN$	$logN$
Degré	$logN$	$2d$	$\beta log_\beta N$	$k\beta log_\beta N$	$klogN$	7	$2 \cdot logN$
Architecture	Unidirectional ring	Multidimensional coordinate space	Plaxton-style mesh network and ring	Plaxton-style mesh network	Plaxton-style mesh network	Butterfly Network	Bidirectional rings
Metrics	hop number	hop number	hop number& RTT	hop number & TTL	hop number& oldest node	hop number	hop number
Redondance	1	1	1	k	k	1	1

TABLEAU 1.2 – Comparaisons de différents protocoles pair-à-pair structurés (Malatras, 2015)

blockchain publique, un nœud peut rejoindre librement le réseau tandis que dans le cas d'une blockchain privée, cet accès est restreint et nécessite l'approbation d'un tiers de confiance. Cette question amène des problématiques de sécurité et le principal risque pour la chaîne est de subir une attaque Sybil (Douceur, 2002). Dans ce scénario d'attaque, un malveillant crée plusieurs identités et les fait participer au consensus. Comme ces fausses identités sont nombreuses et potentiellement majoritaires, le malveillant peut ainsi altérer le consensus. Les blockchains privés sont donc immunisés contre ce scénario d'attaque grâce au besoin d'authentification. Cependant, dans le cas des blockchains publiques, le consensus doit empêcher l'apparition d'attaque Sybil. L'utilisation d'une preuve à base d'une ressource est une parade à cette attaque puisque la ressource ne peut être que répartie dans les différentes identités contrôlées par l'attaquant.

La blockchain privée amène cependant des problématiques d'anonymats. En contrôlant la distribution des accès, le tiers de confiance supervisant ce type de blockchain peut connaître l'identité des acteurs du réseau. Une solution à ce problème est d'utiliser d'intégrer des algorithmes de signatures de groupe dans la chaîne pour garantir l'anonymat tout en conservant la traçabilité, comme le propose Dehez-Clementi et al. (2020) avec leur schéma de signature *DOGS*.

Le second critère (avec ou sans permissions) concerne plus spécifiquement les autorisations des différents nœuds et s'ils ont les droits pour vérifier, exécuter et/ou modifier les données de la chaîne. Un système *permissionless* est entièrement décentralisé, sans différence d'autorisation pour les différents nœuds. Il s'agit donc d'un modèle très utilisé dans le cas des cryptomonnaies qui nécessite cette décentralisation. Dans d'autre contexte, comme le contexte IoT, il peut être intéressant d'avoir des autorisations dépendantes des nœuds. Un tiers de confiance est alors défini pour distribuer les différentes autorisations. Il s'agit du modèle *permissioned*. Ces permissions peuvent restreindre les nœuds pour ne pas exploiter certaines transactions, ne pas autoriser la vérification de certaines données ou empêcher la lecture de transactions en les chiffrant, comme le réalise Hyperledger (Androulaki et al., 2018).

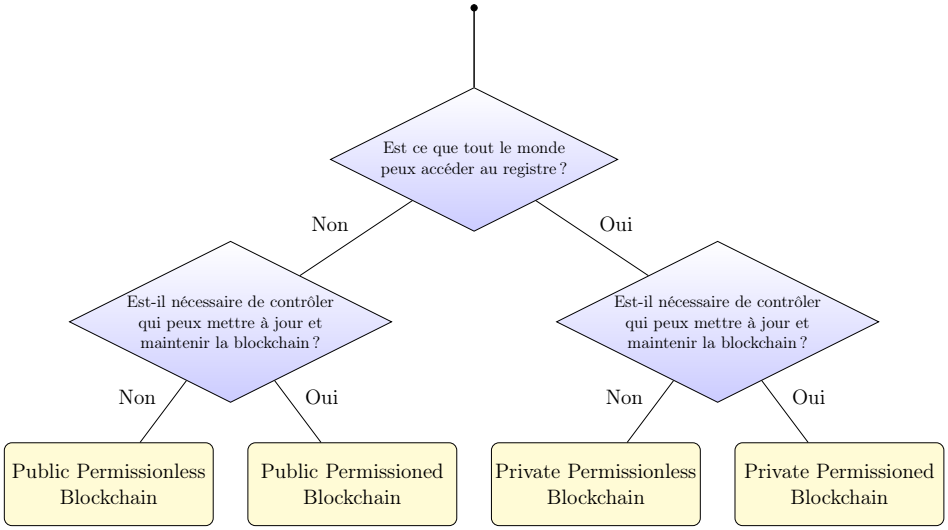


FIGURE 1.9 – Sélection d'un modèle de blockchain (d'après Fernández-Caramés et Fraga-Lamas (2018))

1.4.4 Couche consensus

La couche de consensus est la couche la plus discutée et importante. Elle conditionne la sécurité et le comportement de la chaîne. Le rôle du consensus est de coordonner les divers nœuds du réseau pour définir quel bloc doit être ajouté à la chaîne. Le consensus répond au problème décrit dans la section 1.2.5 et permet d'assurer que les données contenues dans les blocs soient cohérentes sur l'ensemble des nœuds. Contrairement aux algorithmes de base de données répliquée, il répond à des exigences de défaillance et de sécurité, pouvant tolérer sur le réseau des nœuds malveillants ou défaillant.

Comme vu précédemment, on peut distinguer deux grandes familles de consensus : les consensus privilégiant la disponibilité et les consensus privilégiant la cohérence. Par leurs fonctionnements, on appellera les premiers *Proof-based* et les seconds *Voting-based*.

Proof-Based

La famille des consensus basés sur une preuve (aussi notée PoX) vise à utiliser une certaine ressource (CPU, mémoire, bande passante ...) pour former une preuve que les malveillants ne peuvent pas obtenir facilement. Les nœuds, aussi appelés mineurs, entrent ainsi en compétition pour former cette preuve et celui qui arrive à la former peut imposer son bloc à l'ensemble du réseau. La difficulté et le coût associé à la construction de cette preuve assurent de la confiance du nœud soumettant la preuve. Les mineurs sont généralement motivés pour construire cette preuve par une rétribution avec une compensation souvent monétaire. Cette famille de protocoles est donc très souvent utilisée dans les blockchains associés aux cryptomonnaies.

L'un des principaux défauts de cette famille est de permettre l'apparition de *fork*. Dans le cas où deux mineurs trouveraient chacun une preuve pour un même bloc, une division apparaît avec plusieurs versions possibles de la blockchain en compétition. C'est le *fork*. La source de ce *fork* peut être inhérente au protocole ou alors obtenue à la suite d'un partitionnement du réseau. Dans ce cas, un protocole doit être prévu pour sélectionner une des versions et l'imposer au réseau, généralement celle qui a nécessité le plus de ressources.

La possibilité de *fork* est une conséquence de l'appartenance de la famille PoX au groupe *AP* des systèmes distribué selon le théorème CAP (cf 1.2.6), puisqu'un *fork* traduit un défaut de cohérence (plusieurs versions d'une base de données dans un système répliqué) (Angelis et al., 2018). Une blockchain basée sur une preuve n'est donc pas complètement immuable ni strictement cohérente. On dit cependant qu'elle est éventuellement cohérente (Anceaume et al., 2019), c'est-à-dire qu'elle est cohérente au bout d'un certain temps. En effet, pour un bloc proposé, plus le nombre de blocs ajoutés à sa suite est important, plus les ressources nécessaires pour construire une blockchain parallèle sans ce bloc sont importantes, et plus le bloc a de chance d'être immuable et cohérent. Ainsi, un usage est d'attendre quelques blocs à la suite de l'insertion d'une transaction dans un bloc afin de confirmer l'inscription de cette transaction dans la chaîne (Nexo, 2021).

La disponibilité de cette famille AP se traduit également par la possibilité pour une preuve d'émerger dans un système partitionné. En effet, pour être un système tolérant aux défaillances, la preuve n'est pas associée à un mineur particulier, au cas où ce dernier deviendrait inopérant. Un sous-système isolé à la suite d'un partitionnement pourra donc toujours exécuter toute transaction qui lui est soumise, car chaque nœud a la possibilité de proposer le nouveau bloc, contenant donc éventuellement cette transaction. Un système de preuve est donc disponible.

La sécurité du protocole est assurée tant qu'un ou plusieurs malveillants coordonnés ne disposent pas de la majorité de la ressource. Dans ce cas, il s'agit d'une attaque des 51% (Aponte-Novoa et al., 2021). En contrôlant cette ressource, les malveillants peuvent contrôler les blocs qui seront ajoutés et sélectionner les transactions à insérer. L'immuabilité de la chaîne n'est également plus garantie. En disposant de la majorité des ressources, les malveillants peuvent écrire une blockchain parallèle, avec des transactions choisies, et venir l'imposer au reste du réseau, et venir ainsi rendre caduque la blockchain initiale. C'est l'attaque de la *double dépense*.

Par ailleurs, la blockchain est une technologie vérifiable. Cela implique que la preuve doit l'être également et être déterministe pour que chaque acteur du réseau puisse recalculer la correction de la preuve. Cependant, certains protocoles introduisent de l'aléatoire dans la création de la preuve, en faisant par exemple appel aux fonctions de hachages, afin de rendre plus difficile la prédiction du prochain mineur. En introduisant cet aléatoire, le consensus gagne en sécurité, car il devient presque impossible pour un malveillant de concentrer une attaque sur ce prochain mineur. Dans ces systèmes, la blockchain devient assimilable à un tirage aléatoire pour sélectionner les prochains mineurs trouvant une preuve.

Enfin, l'intérêt de baser la construction d'une preuve sur une ressource est de pouvoir résister aux attaques Sybil. Cette attaque, explicitée plus longuement en section 1.4.3, consiste à créer de fausses identités pour s'approprier le droit de proposer un bloc. Dans le cas des consensus à base de preuve, la multiplication de compte n'entraîne pas d'augmentation de la probabilité totale de proposer un bloc, puisque la ressource est fixe. La modélisation de la sélection des prochains mineurs trouvant une preuve prend en compte ce besoin de ressource et est alors pondérée par la quantité de ressources des différents mineurs. Un mineur avec davantage de ressource a alors bien plus de chance de trouver une preuve. L'attaque des 51% décrite précédemment trouve son origine dans cette pondération, puisque sur une période suffisamment longue, le nœud possédant la majorité des ressources construit plus de preuve que le reste du réseau et donc pouvoir réécrire la chaîne.

La ressource utilisée peut être de différentes formes : puissance de calculs, stockage (Proof-of-Space), temps (Proof-of-Elapsed-time)... Des consensus mêlant différentes ressources existent également dans la littérature (Khan et al., 2020). Les exemples emblématiques de la preuve de travail (Proof-of-Work) et la preuve d'enjeu (Proof-of-Stake) sont évoqués dans la suite.

Proof-of-Work La preuve de travail (*Proof-of-Work* ou *PoW*) est un consensus à base de preuve utilisant la puissance de calcul du nœud pour construire sa preuve. Cette preuve est basée sur un puzzle cryptographique difficile à résoudre nécessitant de nombreux calculs pour trouver une solution satisfaisante. Ce concept a été introduit par Dwork et Naor (1993) pour contrer les attaques à déni de service, mais a été principalement popularisé par le protocole Bitcoin (Nakamoto, 2008).

Généralement, la preuve de travail est une exigence sur le haché du bloc à ajouter. Une nonce est alors présente dans l'en-tête du bloc, paramètre que l'on fait varier jusqu'à ce que le haché du bloc réponde à cette exigence. Comme le haché n'est pas prédictif (résistance à la seconde préimage), le mineur doit tester toutes les possibilités de nonce par force brute jusqu'à en trouver une satisfaisant l'exigence. Cette exigence rend le haché très difficile à trouver pour un mineur seul, le temps nécessaire pour trouver un bloc est de l'ordre de quelques années. Cependant, comme la recherche d'une solution par force brute se parallélise très facilement, l'ensemble du réseau de mineurs peut trouver une solution dans un temps satisfaisant. De plus, le temps nécessaire pour trouver un bloc, pour un réseau donné, est relativement constant, permettant une émission de blocs régulière.

Bitcoin utilise la preuve de travail Hashcash (Back, 2002). Ce consensus implique que le haché d'un bloc doit être inférieur à une valeur cible, ce qui est équivalent dans le contexte à trouver un haché commençant avec un certain nombre de zéro. Cette valeur est mise à jour régulièrement pour adapter la valeur au réseau. Bitcoin utilise comme fonction de hachage l'algorithme SHA-256 (Dang, 2015) donnant un haché sur 256 bits, équivalent à 64 nybbles (ou demi-octets, il s'agit du nombre au format hexadécimal). Actuellement, le consensus Bitcoin demande à ce que les 19 premiers nybbles soient égaux à zéro. Un nybble étant codé sur 4 bits, et le haché étant pseudoaléatoire, le réseau doit donc réaliser de l'ordre de $(2^4)^{19} \simeq 7,5 \cdot 10^{22}$ hachés avec une nonce différente pour trouver une nonce satisfaisante. Sachant que le hashrate (nombre de hachés réalisé par seconde) du réseau est de actuellement $1,93 \cdot 10^{20}$ hachés par seconde (Blockchain.com, 2022), le temps nécessaire est de l'ordre de $\frac{16^{19}}{1,93 \cdot 10^{20}} \simeq 381s \simeq 6,3min$. Ce résultat approximatif est dans l'ordre de grandeur des 10 minutes prévues par le protocole. Le calcul présenté ici est une simplification, le véritable calcul implique que le haché soit inférieur à une valeur actuellement de l'ordre de 10^{56} et nécessite des temps moyens de 10 min pour créer un bloc.

Par comparaison, un utilisateur seul avec une bonne carte graphique dispose d'un hashrate de l'ordre de 10^8, le temps nécessaire pour qu'il puisse miner un bloc est donc $\frac{16^{19}}{10^8} \simeq 2,3 \cdot 10^7$ années. Un mineur seul n'a donc pas la possibilité de concurrencer la chaîne principale.

La preuve de travail est un consensus très utilisé, notamment par les cryptomonnaies, car ce consensus passe facilement à l'échelle et ne nécessite pas d'inscriptions préalables à un registre pour participer au consensus. Outre Bitcoin, elle est utilisée par diverses autres chaînes comme Ethereum (Wood et al., 2022) ou le Dogecoin (Jeong et al., 2017). Cependant, ce consensus est très critiqué, car très consommateur d'énergie. En effet, la recherche d'une solution par force brute nécessite de nombreux calculs inutiles, consommant énormément de ressources. Ainsi, la consommation énergétique du réseau Bitcoin est équivalente à la consommation électrique de l'Irlande (de Vries, 2018).

Proof-of-Stake Afin d'économiser de l'énergie en n'utilisant pas une ressource physique comme le réalise la preuve de travail avec l'électricité, la preuve d'enjeu (ou *Proof-of-Stake*, *PoS*) demande à un mineur de prouver sa possession une certaine quantité de jetons. Un mineur est ensuite sélectionné parmi les nœuds proposant des jetons et est autorisé à proposer un bloc. Ce protocole se base sur l'hypothèse qu'un mineur ayant beaucoup de jetons a beaucoup participé

au protocole, et donc qu'il a moins tendance à vouloir le saboter.

La sélection de ce mineur dépend du protocole. En général, le mineur sélectionné n'est pas toujours celui proposant le plus de jetons, pour éviter une centralisation de la chaîne aux mains du plus riche. Un protocole de génération d'aléatoire sécurisé est ainsi souvent utilisé pour sélectionner le mineur, avec une probabilité pondérée en fonction de la quantité de jetons. La chaîne Blackcoin est un exemple de chaîne entièrement PoS avec une sélection aléatoire pondérée sur la quantité de jetons mise en jeu (Vasin, 2014).

Une hybridation avec la preuve de travail est généralement réalisée pour pouvoir générer cet aléatoire de manière sécurisé. Ainsi, la chaîne Peercoin utilise une preuve de travail en baissant la difficulté de minage en fonction d'une quantité de jetons et de l'âge de ces jetons (King & Nadal, 2012).

La preuve de travail et la preuve d'enjeux sont donc souvent liées à l'utilisation d'une cryptomonnaie. Dans le cas d'un système fermé où une cryptomonnaie n'est pas utile, d'autres protocoles peuvent être mis en jeu, généralement basés sur la famille BFT comme nous allons le voir.

Voting-Based

En utilisant une ressource pour former une preuve, les consensus composant la famille précédente se confrontent à des problèmes énergétiques et économiques. Pour appliquer des blockchains dans des réseaux plus restreints où les problématiques de confiance sont plus simples, une optimisation des ressources peut être réalisée en exploitant la seconde famille, les consensus basés sur un vote, qui comme nous allons le voir fournissent par ailleurs de meilleures performances de sécurité.

Cette famille de consensus se différencie de la famille précédente par son mécanisme d'ajout de bloc. Si dans la famille précédente, chaque membre tente de miner un bloc indépendamment des autres mineurs pour ensuite imposer son bloc, les consensus basés sur un vote cherchent à obtenir un accord commun entre les différents nœuds du réseau pour tenter d'ajouter un unique bloc. Cet accord est obtenu à la suite d'un ensemble d'élections d'où la notion de "vote" dans la dénomination du protocole. Le coût de cette famille de protocoles est ainsi un coût principalement réseau que l'on cherche à minimiser pour ne pas perturber les communications. Ainsi, le coût d'un tel protocole se mesure généralement en nombre de messages pour ajouter un bloc.

Ces consensus s'inscrivent donc dans les systèmes CP dans le théorème CAP (cf. section 1.2.6). Ils assurent une cohérence stricte et par conséquent n'admettent pas la possibilité de *fork*, même si le système est partitionné. Grâce à cette propriété, l'attaque par double dépense ne peut être réalisée. À l'inverse, la disponibilité n'est pas assurée selon le théorème CAP. En effet, comme l'accord d'ajout d'un bloc se réalise via un ensemble d'élections, si le système est suffisamment partitionné, les nœuds ne peuvent pas communiquer entre eux, ne peuvent donc pas réaliser les votes et aucun bloc ne peut être ajouté. Dans ce cas, les transactions soumises ne peuvent être éventuellement ajoutées.

La principale restriction de cette famille de consensus est la distribution des droits de vote. Généralement, ces consensus réalisent un ensemble de votes avec un nombre de votants fixe et les droits de vote sont associés aux nœuds et ne peuvent être modifiés. Des exceptions existent et seront évoquées dans la section 2.2.3.

Deux classes de consensus sont généralement définies et ont été évoquées dans la section 1.2.5 : la classe CFT et la classe BFT, avec la classe BFT incluse dans la classe CFT. L'appartenance d'un consensus à la classe strictement CFT ou BFT possède de forts impacts sur les performances

de ce consensus. La classe CFT (*Crash Fault Tolerant*) regroupe les ensembles de protocoles capables de parvenir à un accord et de conserver la cohérence même en cas de défaillance de plusieurs nœuds. Dans cette classe, nous retrouvons des protocoles tels que Paxos (Lamport, 2001) et Raft (Ongaro & Ousterhout, 2014), ce dernier étant utilisé dans la blockchain Hyperledger depuis la version 0.6 (Androulaki et al., 2018). Le protocole Paxos est détaillé plus en profondeur dans l'annexe A.1.1. Ces protocoles sont exclusivement CFT et peuvent éprouver des difficultés à fonctionner correctement en cas de présences de nœuds malveillants. Une sous-classe de protocoles existe, capable de tolérer une portion de nœuds malveillants, ce sont les consensus BFT (*Byzantine Fault Tolerant*).

L'exemple emblématique de protocole BFT est le protocole pBFT (*Practical Byzantine Fault Tolerant*) défini en 1999 par Castro et Liskov (1999). De nombreux variants ont été étudiés au fil du temps en fonction des paradigmes employés, comme dBFT qui est une implémentation où les nœuds peuvent déléguer leur droit de vote à un autre nœud (Coelho et al., 2020), ou les consensus FBA, où plusieurs groupes de nœuds pouvant se chevaucher coexistent et réalisent des consensus dans chaque groupe (Yoo et al., 2019). Il s'agit d'un protocole de réplications de machine à états (Lamport, 1978) assez peu coûteux en ce qui concerne le nombre de messages comparés aux autres protocoles BFT.

Le protocole PBFT est constitué de deux élections successives, pour un nombre de messages quadratique en fonction du nombre de nœuds participants. Ainsi, pour un nombre n de valideurs, le nombre de messages nécessaires pour ajouter un bloc est égal à $2*n^2-n+1 = O(n^2)$ messages.

Ce protocole garantit les propriétés de sûreté dans un réseau asynchrone et synchrone, et les propriétés de vivacité dans un réseau synchrone. Un réseau synchrone suppose que le temps nécessaire à la transmission d'un message est limité par une borne connue, contrairement à un réseau asynchrone où il n'y a pas de borne. Cette différence implique que dans un réseau synchrone, il est possible de détecter qu'un nœud voisin est défaillant s'il ne répond pas au-delà d'un certain temps défini, contrairement au réseau asynchrone où l'on ne peut faire d'hypothèses sur la défaillance du voisin. Le protocole pBFT garantit l'exactitude dans les hypothèses les plus restrictives, la vivacité ne pouvant être garantie dans un réseau asynchrone d'après le théorème FLP (Fischer et al., 1985).

Le protocole pBFT étant utilisé dans les sections 4 et 5, son fonctionnement est détaillé par la suite :

Éléments présents Cet algorithme suit le déroulement dans le cas général du diagramme représenté sur la figure 1.10. Les différents éléments représentés sont :

- c : un des clients. Il s'agit d'un acteur désirant effectuer une requête venant modifier la base de données ;
- i : l'identifiant d'un des nœuds composant le réseau ;
- m : le message ou la requête demandée par le client c ;
- d_m : le condensat du message (le hachage par exemple). Il permet de lier au message m exact un identifiant plus court et difficilement falsifiable ;
- r_m : la réponse, si elle est nécessaire, à la requête m ;
- t : le timestamp du moment où c effectue sa requête ;
- v : une *view*. Il s'agit d'un numéro indiquant une exécution du protocole qui représente une configuration du système à un moment donné. La *view* est ainsi un numéro directement lié au numéro identifiant le nœud primaire, le nœud coordonnant l'ensemble des autres nœuds ;

— n : un numéro d'exécution n. Il sert à ordonner l'ensemble des modifications au sein de la base afin d'assurer la cohérence des modifications. Il est également utilisé lors de la compression des fichiers temporaires pour assurer l'exactitude des données ;

— h et H : un intervalle où n doit être compris.

Le réseau de nœuds se compose de deux types de nœuds : le nœud primaire et les sauvegardes. Le nœud primaire est le nœud coordonnant l'ensemble des interactions des autres nœuds en fonction des requêtes qu'il reçoit. Les sauvegardes acceptent les modifications demandées, sous réserve de validité, par le nœud primaire. Il n'y a qu'un seul nœud primaire au sein du réseau et ce dernier est sélectionné en fonction de la *view*. En cas de défaillance du nœud primaire, un mécanisme de temporisation détecte l'état de ce nœud et modifie la vue par deux élections successives au $^2/_3$ au sein du réseau (cf. 1.4.4). Ce mécanisme assure la vivacité du système en réseau synchrone.

Pour assurer la cohérence, le réseau partage les mêmes valeurs de v, H et h. Les nœuds s'accordent sur ces deux dernières valeurs durant l'étape *Checkpoint* qui ne sera pas détaillée. Cette étape correspond à la création de points de récupération en cas de pannes du réseau, H et h étant les limites de l'intervalle entre deux points de récupération. Dans l'exemple, le réseau est de taille $N = 3 \cdot f + 1$ où f est un nombre de nœuds malveillants.

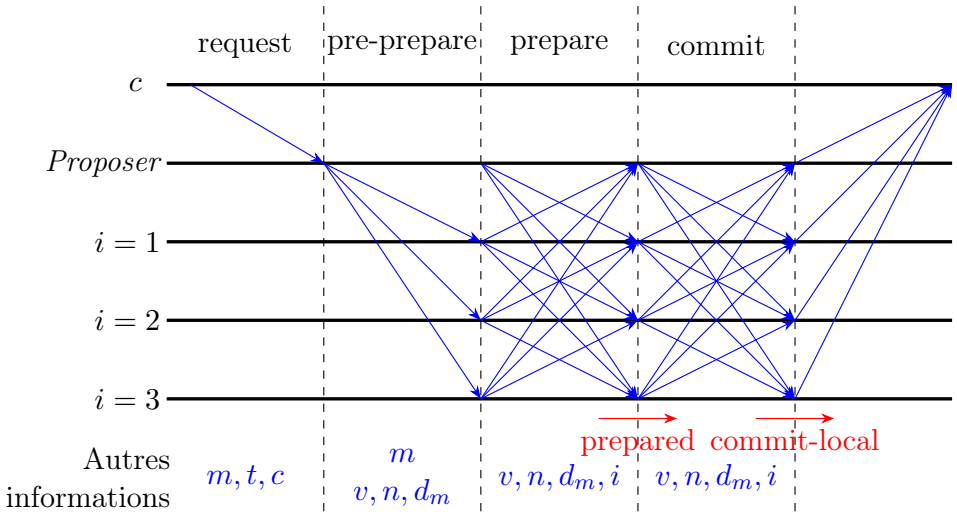


FIGURE 1.10 – Diagramme représentant le flux de message dans un cas général qui a réussi (Castro & Liskov, 1999).

Déroulement d'une exécution La figure 1.10 correspond au déroulement d'une exécution correcte d'une requête. Tous les messages échangés sont authentifiés, soit par exemple avec des signatures numériques (cf 1.1.3), soit pour de meilleures performances, avec des messages MAC (cf 1.1.3). Le protocole dédié à l'envoi de message en *broadcast* n'est pas précisé, il se comprend comme l'envoie d'un message aux autres nœuds du réseau en unicast.

Quand un client c souhaite effectuer une requête modifiant la base de données, il envoie une requête *Request* composée du message m, du timestamp d'émission t, afin d'ordonner les requêtes, et l'identité du client c. Cette requête est donnée à l'un des nœuds du réseau qui le retransmet au nœud primaire.

Pre-Prepare Dès lors, il convient d'associer un unique numéro de séquence à la requête. Une fois la requête reçue, le nœud primaire le transmet à l'ensemble des sauvegardes en multicast accompagné d'un message *Pre-Prepare* composé du numéro de la vue actuelle v, d'un numéro de séquence contenue dans un intervalle spécifié (entre h et H), et du condensat du message d_m. Le message *Pre-prepare* est séparé de la diffusion du message m afin de permettre un envoie sur deux canaux, l'un spécifique aux messages courts pour les commandes (la view v, le numéro n et le condensat d_m), l'autre aux messages longs pour les données (le message m).

Les sauvegardes vont accepter et enregistrer le message *Pre-Prepare* s'ils ont bien reçu le message m correspondant au condensat d_m, qu'il est bien à la *view* v, que le numéro de séquence n'ait pas déjà été utilisé pour un autre message et qu'il est bien dans l'intervalle entre h et H.

Prepare Si une sauvegarde accepte la requête *Pre-Prepare*, celle retransmet alors à tous les nœuds du réseau un message *Prepare* contenant également les variables v, n et d_m, mais également l'identité i du nœud. Chaque sauvegarde enregistre les messages *Prepare* reçus si le message est correct, si le nombre de la *view* est égal au nombre de la *view* courante et si le numéro de séquence est compris entre les bornes.

Prepared Dans le cas où un nœud (nœud primaire également) possède un message m, a reçu (ou émis) un message *Pre-Prepare* correspondant avec un numéro de *view* et un numéro de séquence valide et a également reçu au moins $2f$ messages *Prepare* de nœuds différents et concordants avec ce message *Pre-Prepare*, le prédicat *Prepared* est validé. Le but de cette phase est de s'assurer qu'au moins $f + 1$ nœuds honnêtes se mettent d'accord sur le fait que si une modification m est effectuée dans la vue v, elle s'exécutera avec le numéro de séquence n. Elle répond donc à un problème de synchronisation des modifications et définit donc un ordre total sur les requêtes basé sur le numéro de séquence.

Commit Dans le cas où un nœud posséderait le prédicat *Prepared* validé, il émet en *broadcast* un message *Commit* à l'ensemble des autres nœuds avec les informations v, n, d_m et son identifiant i. Les autres nœuds acceptent ce message tant que son numéro de *view* et son numéro de séquence sont valides.

Commit-local Si un nœud possède le prédicat validé *Prepared* pour un message m, un numéro de *view* et un numéro de séquence n, et qu'il a également reçu $2f + 1$ messages *Commit* (dont possiblement le sien) qui correspondent au message *Pre-Prepare* de m, alors le prédicat *Commit-local* est vérifié. Dans ce cas, le nœud applique la requête du message m. L'état de i reflète alors l'ensemble des modifications de numéro de séquence inférieur à n. Le but de cette phase est de garantir qu'au moins $f + 1$ nœuds honnêtes se mettent s'accordent sur l'exécution de la modification m durant la vue v avec le numéro de séquence n. Cette phase garantit donc l'atomicité de la modification.

Reply Après exécution de la modification, le nœud envoie une réponse r_m au client c dans un message *Reply*, avec le timestamp t de la première requête. Les requêtes *Request* reçues ensuite par les nœuds seront ignorées si leur timestamp est inférieur au timestamp du dernier message *reply*. Le client accepte ensuite la réponse s'il reçoit plus de $f + 1$ messages de réponse avec le même t et la même réponse r_m.

Checkpoint Afin de vider régulièrement la mémoire rapide des nœuds qui stocke les messages, une preuve est régulièrement construite pour sauvegarder l'état du système dans une seconde mémoire plus sûre. Un protocole est défini pour assurer la cohérence de cette preuve, mais ne sera pas explicité.

Cas où le nœud primaire est défaillant Si le nœud primaire tombe en panne, ou est lent, cette défaillance est détectée par les autres nœuds par un système de temporisation. En effet, dès qu'une sauvegarde reçoit une requête, il démarre une temporisation qu'il arrête à l'exécution de la requête. Si la temporisation expire dans une vue v, il émet un message *Vew-Change* demandant à changer de vue. En effet, l'identité du nœud primaire dépend de la vue ($ID_{primaire} = v \bmod N$) avec un ensemble d'autres informations et refuse l'acceptation d'autres messages, sauf si liés aux changements de vue.

Dès que le nœud primaire de la vue $v+1$ reçoit $2f$ messages *Vew-Change*, il émet un message de changement de vue *New-View* avec un agrégat des informations depuis le dernier Checkpoint. Si ces informations sont cohérentes avec les logs des sauvegardes, ils intégreront les changements.

Dans le cas où le nœud primaire adopterait donc un comportement byzantin, il peut corrompre le réseau à deux moments : en sélectionnant les requêtes pour entreprendre une itération de l'algorithme de la figure 1.10 ou lors de la réponse à la suite de cette itération. En effet, il ne peut modifier significativement les décisions prisent lors d'une itération puisqu'elles demandent l'accord d'au moins 2/3 des nœuds.

Dans le premier cas où le nœud primaire sélectionne les requêtes, les sauvegardes peuvent détecter la malveillance puisqu'ils retransmettent les requêtes au nœud primaire. Ainsi, les requêtes non diffusées peuvent être interprétées comme une défaillance du nœud et enclencher l'algorithme de changement de vue. De plus, cette attaque ne perturbe en rien la cohérence des données.

Le système pare l'attaque évoquée dans le second cas en faisant répondre l'ensemble des nœuds du réseau à cette requête, expliquant ainsi les multiples réponses *Reply* sur la figure 1.10. Le client sélectionne donc la réponse majoritaire. Le nœud malveillant peut toutefois répliquer en envoyant un nombre plus important de *Reply* frauduleux. Dans ce cas, les réplicas frauduleux peuvent être détectés avec l'authentification des réponses.

1.4.5 Couche contrat

Formellement, la blockchain est une structure de données pouvant contenir des données quelconques comme toute chaîne de caractère. La grande innovation initiée par Satoshi Nakamoto avec Bitcoin (Nakamoto, 2008) fut d'associer un registre en parallèle à la blockchain et d'y insérer des transactions bancaires. La blockchain n'est donc que l'historique de ces transactions et sa lecture intégrale permet la reconstitution du registre. Ces transactions doivent respecter les propriétés ACID (Haerder & Reuter, 1983), se définissant par les quatre propriétés suivantes :

- Atomicité : une transaction est réalisée ou n'est pas du tout exécutée ;
- Cohérence : une transaction doit être valide et respecter les règles du registre ;
- Isolation : une transaction doit s'exécuter en isolation totale et ne doit donc pas être influé par l'exécution d'une transaction en parallèle.
- Durabilité : si une transaction est exécutée correctement, ses effets sont persistants et doivent survivre si le registre devient défaillant.

L'immuabilité et la tolérance aux pannes de la blockchain impliquent la propriété de durabilité. La sérialisation des transactions implique la propriété d'isolation. L'atomicité et la cohérence

des transactions sont vérifiées lors du consensus. En effet, dans une blockchain, les nœuds vérifient la conformité des transactions lors de l'ajout d'un bloc. Si une transaction n'est pas valide, le nœud ne valide pas le bloc et ne l'ajoute pas (quelques blockchains font exceptions comme Hyperledger (Androulaki et al., 2018)). La blockchain implique ainsi une transparence des données pour que le réseau puisse vérifier les transactions.

En 1997, Szabo imagine le concept de contrat intelligent (*smart contract*) (Szabo, 1997). Il s'agit de contrats réalisés sur un réseau qui s'exécutent automatiquement une fois les termes d'exécutions rencontrés sans impliquer de tiers de confiances afin d'accélérer et de sécuriser le processus. Cette idée est intégrée dans différentes blockchains et particulièrement Ethereum (Buterin, 2014). Ainsi, les blockchains embarquant des *smart contracts* permettent d'insérer un code informatique au lieu d'une transaction. Ce code embarque la logique du contrat et peut s'exécuter automatiquement quand les conditions sont rencontrées. Le contrat devient donc décentralisé, sans tiers de confiance, et est vérifiable par tous les membres du réseau.

La capacité d'exécution par une blockchain de code informatique rend donc cette blockchain complète au sens de Turing, permettant donc de considérer la blockchain comme un ordinateur décentralisé, capable d'exécuter n'importe quel programme. Dès lors, en adaptant contrats et consensus, la blockchain peut fonctionner comme une grappe de serveur et additionner les puissances de calcul de chaque membre pour réaliser une tâche commune. Des projets vont ainsi dans ce sens, par exemple, pour la manipulation d'image de haute résolution (Rapuano et al., 2020). Ce nouveau paradigme d'ordinateur décentralisé sécurisé et vérifié offre donc de nouvelle perspective d'applications sur lesquelles nous reviendrons au chapitre 3.

1.4.6 Couche application

La dernière couche du modèle présenté dans la figure 1.6 est la couche Application, laquelle regroupe les différentes applications et cas d'utilisation de la blockchain. De nombreux projets sont en cours de développement pour définir des applications décentralisées (aussi appelé *DApp*), mettant la chaîne aux cœurs d'écosystème ou définissant de nouveaux modèles d'affaires entre les mineurs.

Si l'exécution des contrats intelligents est vérifiable et sécurisable grâce à la vérification par les paires, les interactions entre la blockchain et le mode physique sont source de failles de sécurité. En effet, cette exécution d'un contrat intelligent dépend des entrées des programmes, qui peuvent être difficilement vérifiés. Une confiance doit donc être accordée aux éléments interagissant avec la blockchain. Dans le cas d'applications décentralisées construites sur des blockchains publiques, des entités appelées *Oracles* (Caldarelli, 2020) sont utilisées afin d'importer de l'information "externe" dans la chaîne formant par conséquent des tiers de confiances. Dans le cadre d'écosystèmes dédiés, les protocoles développés doivent être réfléchis pour éviter ces failles. Un exemple de pratique est de ne sauvegarder dans la blockchain que des hachés de données stockés dans une base de données sécurisée, menant à des interactions blockchain-serveurs pouvant potentiellement former un goulot d'étranglement des performances.

Plusieurs domaines d'utilisation ont été fortement révolutionnés par la blockchain. Si certains auteurs en compte jusqu'à 6 domaines impactés (Belotti et al., 2019 ; Wüst & Gervais, 2017 ; J. Xie et al., 2019), trois principaux émergents :

— La gestion de la chaîne d'approvisionnement (*Supply Chain Management* ou *SCM*). La transparence et l'inaltérabilité de la chaîne sont exploitées pour améliorer la traçabilité et le suivit des marchandises (Tian, 2016) offrant une vue complète et précise de ces marchandises. De plus, la flexibilité du système, son automatisation sécurisée et son ouverture

aux clients permettent de renverser la conception où les entreprises *proposent* des biens à une conception où le client *demande* les produits (Wüst & Gervais, 2017) ;

— Le contrôle d'identité. Dans le cadre des systèmes publics sans permissions, les blockchains peuvent être considérés comme une interface anonyme permettant la mise en place d'interactions soumissent à des règles établies et vérifiées par l'ensemble des utilisateurs. La blockchain permet donc d'établir des échanges d'actifs monétaires (cryptomonnaie (Yadav et al., 2022)), réel (biens immobiliers par exemple (Nandi et al., 2020)) ou fictif (propriété intellectuelle avec les NFT (Muthe et al., 2020), décisions avec le E-Voting (Hjálmarsson et al., 2018)). Dans le cadre de blockchains avec permission, la blockchain peut s'intégrer pour réaliser du contrôle d'accès, et s'assurer la circulation des règles et l'administration de la politique de sécurité des systèmes d'information d'une manière décentralisé et sécurisé. Ainsi, des recherches sont effectuées pour associer blockchain et langage XACML (Kencana Ramli et al., 2012), afin de spécifier ce contrôle d'accès (Esposito et al., 2021 ; Guo et al., 2019).

— Les contrats intelligents. Grâce aux *smart contracts*, des systèmes complexes peuvent fonctionner intégralement de manière autonome, trouvant des applications dans le domaine de l'énergie (Andoni et al., 2019), de l'automobile (Manimuthu et al., 2022) ou des contrôles de réseaux SDN (Almakhour et al., 2021). Si l'on pousse l'autonomie à sa limite, on peut dès lors parler d'organisations autonomes décentralisées (*Decentralized Autonomous Organization* ou *DAO*). Ce genre d'organisation se fixe comme objectif de créer une gouvernance complètement décentralisée (L. Liu et al., 2021).

1.5 Conclusion

En conclusion, la blockchain est issue de la rencontre entre le domaine de la cryptographie et celui du système distribué. Il s'agit d'une structure de données *secure by design* répondant aux problématiques de cohérence sur un réseau grâce à son consensus. Cette problématique correspond à la définition d'un ordre total unique sur plusieurs nœuds alors que des retards dans les communications peuvent apparaître. Deux types de consensus coexistent, l'un basé sur la construction d'une preuve individuelle, l'autre sur une décision collective grâce à un ensemble de votes. Dès lors qu'un cas d'utilisation pertinent est trouvé, une blockchain peut être conçue pour répondre aux exigences.

Cette technologie dispose cependant de nombreux inconvénient, mais peut trouver son utilité dans des cas particulièrement précis où aucune alternative n'est suffisante. Dès lors qu'un cas d'utilisation pertinent est trouvé, une blockchain peut être conçue pour répondre aux exigences. La conception d'une blockchain peut se décomposer en six couches, dont chacune des couches influe sur la couche adjacente. Des innovations diverses comme la structure de données ou le maillage réseau peuvent être apportées. Enfin, la mise en place de *smart contract* change complètement le paradigme et les possibilités d'utilisations de la blockchain.

Enfin, la mise en place de n'importe laquelle des six couches nécessite de définir de nombreux paramètres. Cependant à leurs niveaux respectifs et dans leurs interactions, ces couches impactent sensiblement la performance de la chaîne au demeurant que ses paramètres apparaissent devoir se fixer de manière statique (pour ne pas dire *offline*), limitant (voire empêchant) toute réponse dynamique de la chaîne aux multiples évolutions de l'environnement (besoins utilisateur, charge réseau, défaillance de lien ...). La pertinence de chacune de ces couches doit pouvoir être critiquée et au mieux mesurée. Comme les performances dépendent beaucoup de l'environnement d'exécution, des protocoles standards doivent être mis en place pour mesurer efficacement

les performances, et pour pouvoir ensuite les améliorer.

2

Problématiques de performance et de sécurité : état de l'art

D'un point de vue applicatif, la blockchain est une formidable technologie permettant à différentes parties prenantes de regrouper sur une même base de données plusieurs informations et d'assurer l'intégrité de toute information sans toutefois dépendre d'un tiers de confiances pouvant manipuler *a posteriori* ces données. Les possibilités offertes par cette technologie motivent donc son intégration dans des écosystèmes IoT afin de permettre notamment des interactions sécurisées et vérifiables entre différents agents du système, donnant naissance au concept de BIIoT (*Blockchain for Industrial Internet of Things*) (R. L. Kumar et al., 2022).

Cependant, la blockchain est une technologie lourde : le coût (financier, en ressources...) de la sécurité offert par la blockchain impacte fortement les performances en écriture. Le débit maximal d'ajout de transaction est limité par le consensus de la chaîne tandis que les applications IoT peuvent nécessiter des échanges rapides et sécurisés. Une connaissance approfondie des performances de cette chaîne et des informations ajoutées sont donc importantes afin de dimensionner le réseau pour atteindre les performances attendues. Une mauvaise configuration de la chaîne peut dans ces conditions avoir des incidences importantes sur les performances et la sécurité d'une application.

Prenons l'exemple d'une blockchain qui est associé à une application nécessitant une latence faible. Si le consensus choisi est un consensus PBFT et autorise l'ajout de nœud sans régulation, l'application fait face à des problématiques de qualité de service en cas de passage à l'échelle puisque la durée d'ajout d'un bloc est supérieure à la latence désirée. Dans un autre exemple, des problématiques peuvent apparaître si une blockchain avec un consensus basé sur une preuve est déployée sur une plateforme IoT où chaque élément est mineur. Les éléments IoT disposant par définition de ressources limitées, si la blockchain autorise l'ajout de nouveau avec une grande quantité de ressource (un puissant serveur par exemple), des problématiques de sécurités émergent. La configuration de la chaîne est donc liée à l'application associée.

Nous proposons ainsi de revenir sur les différentes étude et solution de métrologie existante dans la littérature. Cette étude et mesure des performances est l'objet de nombreuses études qui seront détaillées dans la section 2.1.

Par la suite, si une blockchain est confrontée à un nombre important de sollicitations de la part de multiples utilisateurs, il peut être intéressant de mettre en place des interactions avec divers éléments pour améliorer les performances de la chaîne et mieux distribuer les ressources nécessaires à l'exécution du service auquel dépend la blockchain. Le consensus peut également être modifié pour atteindre de meilleures performances. Des études présentant des architectures

innovantes et des optimisations de consensus sont présentées dans la section 2.2.

2.1 Mesures des performances de blockchain

En établissant plusieurs paradigmes de consensus et d'architectures de chaînes, la littérature scientifique a constitué un ensemble de blockchains pouvant s'intégrer dans divers écosystèmes. Cependant, afin de déterminer quelle blockchain retenir pour une application donnée, les différentes propositions de chaîne de blocs doivent être évaluées objectivement puis comparées. Les sections suivantes traitent de l'analyse de la littérature relative à la mesure des performances des blockchains.

2.1.1 Comparaison générale des consensus

La couche la plus importante dans le modèle présenté en section 1.4 est la couche dite de "consensus". Cette dernière est connue pour impacter les performances de la chaîne, impliquant une connaissance poussée des avantages et inconvénients des différents protocoles de consensus afin de sélectionner le plus approprié en fonction des attentes et contraintes applicative. Plusieurs études scientifiques se sont intéressées à cette question et sont détaillées dans la suite.

On peut ainsi citer G.-T. Nguyen et Kim (G.-T. Nguyen & Kim, 2018) qui réalise une comparaison entre 32 consensus de blockchains. L'objectif de cette étude est de réaliser une compréhension générale des différents consensus. Cette liste de consensus est d'abord divisée en deux catégories, *Proof-based* et *Voting-based* comme explicité dans la section 1.4.4, puis en sous-catégories de consensus, comme la preuve de travail, la preuve d'enjeux et les consensus byzantins. Un même consensus peut donc être implémenté avec des choix de conception différents. Ainsi, les auteurs dénombrent 8 définitions différentes de la preuve de travail, pouvant chacune être implémentés de manière différente. Un consensus est donc défini dans un sens très large, dont les performances seront liées à cette implémentation. Il ressort également que le positionnement dans la catégorie *Proof-based* ou *Voting-based* est plus important que le choix d'un consensus précis, car des performances très générales, et non chiffrées, ressortent de cette division. Ainsi, les systèmes *Voting-based* sont limités dans le nombre de nœuds et imposent une permission, à l'inverse des systèmes *Proof-based*.

Bamakan et al. (2020) réalisent une étude de comparaison similaire, en considérant moins de protocoles, mais définissent un plus grand nombre de critères d'évaluations qu'ils regroupent en 4 ensembles. Un premier groupement de critères se forme autour du débit de transactions ajoutées par la chaîne. Sont alors évalués le débit, la latence (ou le temps nécessaire pour ajouter une transaction), le temps de vérification des blocs et la taille des blocs. Un second groupe concerne l'incitation à participer à la chaîne, et un troisième, la gouvernance, incluant le modèle de permission. Enfin le dernier groupe concerne la sécurité et la résistance aux attaques Sybil, aux attaques des 51% et de doubles dépenses. Le premier groupe se différencie des trois autres, car il s'agit de performances mesurables et chiffrables, là où les autres sont issus d'un ensemble fini. Par conséquent, les trois derniers groupes de critères sont relativement faciles à définir. La gouvernance et l'incitation sont des choix de conceptions de la chaîne et la sécurité relève de lois mathématiques aisément définies. Les performances chiffrées sont plus difficiles à obtenir et dépendent également du contexte de mesure.

2.1.2 Insuffisance de cette approche

Cette approche par comparaison de blockchains repose uniquement sur le consensus et fonctionne sur des critères de nature "subjectives", souvent relatifs à des caractéristiques intrinsèques au protocole, et rarement évalue les performances de la blockchain considérant des critères de types QoS (latence...). L'exemple de Kaur et al. (2021) montre ces difficultés. Dans cette étude, les auteurs réalisent une comparaison de 4 protocoles de blockchain utilisé pour des blockchains publiques et sans permission (PoW, PoS, DPoS et PoA) sur la base de 15 critères. Une part de ces critères se définissent clairement avec le consensus (modèle de permission, ressource exploitée pour construire la preuve, droits de participation au consensus, sécurité et incitation au minage). Une seconde part utilise des appréciations "subjectives" (scalabilité, charge sur le CPU et degrés de décentralisation qui utilisent des termes "high" ou "low") qui ne mettent pas en avant les limites des consensus. Enfin, une dernière part de valeurs chiffrées (latence, débit maximal en nombre de transactions) incorrectes, car trop simplifiées. Ainsi, les auteurs ont donnés comme valeurs de référence pour la preuve de travail, les estimations de la chaîne principale de Bitcoin. Cependant, comme le met en avant Bamakan et al. (2020) en détaillant plusieurs performances pour plusieurs chaînes, la mesure d'un critère dépend de l'implémentation du consensus, pouvant amener à des mesures avec des ordres de grandeur différents, sur un même critère pour deux implémentations différentes d'un même consensus. Par exemple, Bitcoin nécessite 10 minutes pour miner un bloc tandis qu'Ethereum ne nécessite que 10 secondes.

Par ailleurs, même dans le cas d'une même implémentation d'un protocole, deux blockchains indépendantes peuvent être personnalisées avec des paramètres différents, menant à des performances distinctes. Ainsi, Croman et al. mesure les performances de Bitcoin et déduit la taille idéale d'un bloc en fonction du temps nécessaire pour miner un bloc avant d'en déduire le débit moyen du réseau. De la même manière, Geyer et al. montre un lien entre la taille d'un bloc et le débit maximal en transaction dans le cadre d'Hyperledger Fabric (Geyer et al., 2019). La taille du bloc étant un choix arbitraire et possiblement indépendant d'une chaîne à l'autre. Elle peut avoir des impacts sur le débit de sorte que l'approche générale explicitée dans la section 2.1.1 se révèle insuffisante pour mesurer rigoureusement les performances d'une blockchain. De plus, les mesures données par Kaur et al. (2021) et Bamakan et al. (2020) concernent des valeurs obtenues sur des blockchains publiques. Les valeurs mesurées sur des blockchains privées et sur des réseaux plus petits peuvent être considérablement différentes.

2.1.3 Mesure des performances d'une blockchain et comparaison

Comme de plus en plus de blockchains émergent, avec de nouveaux consensus ou des variations de consensus existant, il devient important de pouvoir évaluer empiriquement une blockchain. Il convient en réalité d'adopter un cadre de mesure stricte et répétable avec des métriques ciblées.

La modélisation d'un modèle en couche à l'instar de celui présenté en section 1.4 est une première étape, car elle permet de localiser la métrique sur une blockchain. Dans le papier cité précédemment de Croman et al. (2016), la blockchain était modélisée par cinq couches (réseau, consensus, stockage, vue et *side plan*) permettant de définir des métriques localisées (latence et débit pour la couche de consensus, temps de *bootstrap* (introduction dans le réseau) et débit effectif en fonction du délai de propagation pour la couche réseau, coûts par transaction pour la couche de stockage). Ce modèle en couches est également utilisé par d'autres auteurs, notamment autour d'un modèle en cinq couches (application, exécution, donnée, consensus, réseau) et représenté sur la figure 2.1 ou équivalent (Dinh et al., 2017 ; Dong et al., 2019 ; Fan et al., 2020). Le modèle 2.1 est similaire à la pile présentée en figure 1.5. Si l'ordre des couches est différent,

on peut retrouver toutes les couches d'un des modèles dans l'autre.

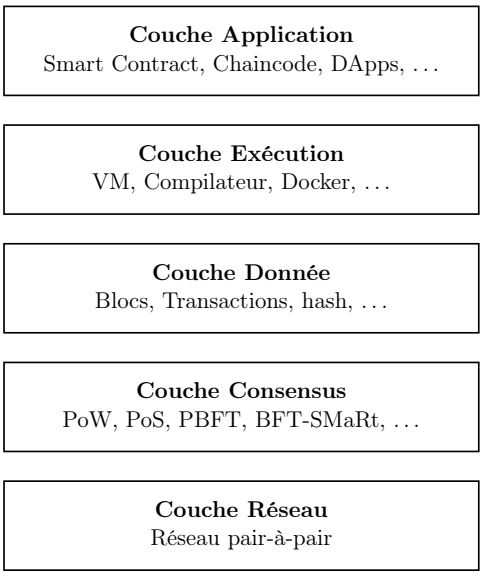


FIGURE 2.1 – Modèle de Blockchain en 5 couches (Dinh et al., 2017 ; Dong et al., 2019 ; Fan et al., 2020)

Mesure par *benchmarking*

Dans le cadre de blockchain déployé sur un réseau privé, il est aisé de réaliser un *benchmark* pour évaluer une blockchain. Si une évaluation de performances consiste souvent à mesurer une variable dans différents contextes, le *benchmark* ajoute la nécessiter de contrôler l'environnement d'exécution. En effet, les consensus de blockchains sont très liés au matériel sur lequel ils s'exécutent. Dans le cadre des chaînes à base de preuve, l'accès à la ressource influe sur la vitesse de construction de la preuve, et dans le cas des chaînes basées sur des votes, la capacité de réseau, et la capacité du nœud à traiter beaucoup de messages, influent fortement sur la vitesse d'exécution des votes. Les résultats des *benchmarks* ne sont donc valables que sur un environnement donné. L'évolution de cette donnée selon plusieurs paramètres est plus significative et permet de trouver les différents goulots d'étranglement qui ralentissent les consensus.

Un premier *benchmark* est proposé par Pongnumkul et al. (2017). Celui-ci compare les blockchains Ethereum et Hyperledger Fabric 0.6 sur une même plateforme, avec l'aide de *smart contracts* dont l'exécution est identique sur les deux blockchains. Les auteurs mesurent le temps d'exécution totale, la latence moyenne et le débit moyen d'un grand nombre de *smart contract* identiques. Les résultats montrent que la blockchain basée sur le consensus PBFT dispose d'une meilleure latence et d'un plus haut débit que la blockchain basée sur le consensus PoW. Une critique est cependant à relever : le consensus PBFT est basé sur le vote et donc peu scalable, à l'inverse du consensus PoW. Les auteurs n'ont donc pas étudié l'influence de l'augmentation du nombre de nœuds qui devrait faire diminuer les performances de Hyperledger Fabric. L'analyse générale que l'on pouvait retrouver en section 2.1.1 est donc toujours intéressante pour prévoir les biais de comparaison.

En développant Blockbench (Dinh et al., 2017), Dinh et al. fournissent un framework complet pour pouvoir mesurer et comparer différentes blockchains. Ils comparent ainsi les blockchains Ethereum, Parity et Hyperlgedger 0.6, et étudient les consensus PoW, PoA et PBFT respectif

(mais d'autres chaînes peuvent être intégrées grâce à une API non définie). La méthodologie proposée par Blockbench s'assure également que la blockchain est adaptée à la plateforme sur laquelle elle s'exécute. Certains consensus comme Ethereum possèdent des variables pour adapter le consensus au nombre de nœuds (cf 3.4), elles doivent donc être adaptées. Plusieurs métriques sont utilisées pour étudier une blockchain, principalement le débit et la latence, mais adaptées selon différents *smart contracts* en fonction de la couche mesurée. Ainsi, 4 *smart contracts* complexes sont définis pour étudier la couche application, un *smart contract* vide est utilisé pour étudier la couche de consensus, un *smart contract* réalisant un algorithme de tri est utilisé pour stresser la couche d'exécution et un *smart contract* réalisant des lectures/écritures pour la couche de données.

Les résultats de Dinh et al. sont les suivants : Hyperledger est systématiquement plus performant que Ethereum et Parity dans tous les *benchmarks*, mais il ne parvient pas à passer à une échelle supérieure à 16 nœuds, en accord avec les attentes de scalabilité de ces consensus. Les principaux goulots d'étranglement d'Hyperledger et d'Ethereum sont les protocoles de consensus alors que pour Parity il s'agit de la signature des transactions. Ethereum et Parity entraînent une surcharge importante en termes d'utilisation de la mémoire vive et du disque. Leur moteur d'exécution est également moins efficace que celui d'Hyperledger. Ce dernier résultat est peu surprenant, puisque les *smart contracts* d'Ethereum et Parity s'exécutent sur une machine virtuelle à la différence d'Hyperledger. Des résultats ultérieurs (Dinh et al., 2018) comparant les versions 0.6 et 1.0 de Hyperledger montrent que le nouveau système d'ordonnancement d'Hyperledger n'est pas plus que le consensus PBFT dans des cas très particulier.

Un autre exemple de *benchmark* est Hyperledger Caliper (Foundation, 2022), développé pour comparer Ethereum et Hyperledger Fabric 1 et 2. Ce *benchmark* mesure la latence et le débit en différenciant les requêtes en lecture et l'exécution des transactions. Par ailleurs, il n'adopte pas la même définition de "transaction ajoutée" en fonction du consensus. En effet, dans le cas des consensus à base de vote, la cohérence est forte. Si un nœud ajoute une transaction, alors il est accepté et est ajouté par le reste des nœuds. La latence d'une transaction correspond donc à la différence de temps entre son émission et le premier ajout de cette transaction dans l'un des nœuds. Dans le cas des chaînes à base de preuve où la cohérence n'est que probable, la définition de Caliper est différente, car la transaction pourrait disparaître suite à un *fork*. La latence correspond donc alors à la différence de temps entre son émission et son ajout par au moins 90% du réseau, où l'on considérer que la transaction est immuable (Foundation, 2018).

Enfin, les problématiques de mesure des blockchains s'étendent également aux DAG (*Directed Acyclic Graph*), qui nécessitent également un consensus. Les DAG sont une généralisation des blockchains. Là où un bloc d'une blockchain peut avoir un seul bloc fils, les blocs des DAG peuvent en avoir plusieurs, ce qui augmente les performances. Ainsi, DAGbench (Dong et al., 2019) mesure les performances de différents DAG (IOTA (Divya & Biradar, 2018), Nano (LeMahieu, 2018), Byteball (Churyumov, 2016)) en adoptant également un modèle en couche en en ciblant comme métriques le débit, la latence, l'évolutivité, le taux de réussite, la consommation de ressources (CPU, mémoire, réseau), la taille des transactions et le coût financier.

Par *monitoring* en exploitation

Si la méthode de *benchmarking* s'adapte très bien dans le cas des blockchains privés, une telle approche est peu adaptée pour le cas des chaînes publiques où il n'est pas possible de standardiser la charge et les participants. Dès lors, deux options sont possibles pour évaluer les performances : un *benchmark* est réalisé au sein d'un réseau privé en espérant qu'il soit valable en production une fois déployé avec une charge de travail non prévisible, où l'on réalise du *monitoring*. Le

Référence Article	**Plateforme**	**Réalise expériences**	**Modèle en couche**	**Métriques utilisées**			
				Débit/Latence	Scalabilité	Ressources	Autre
G.-T. Nguyen et Kim (2018)	32 consensus	□	□	□	□	∅	∅
Bamakan et al. (2020)	30 consensus	□	□	⊠	⊠	∅	blocktime, récompense, sécurité, coûts, ...
Kaur et al. (2021)	PoW, PoS, DpoS, PoA	□	⊠	⊠	⊠	∅	blocktime, récompense, sécurité, ...
Croman et al. (2016)	Pow, pBFT	□	⊠	⊠	⊠	Réseau	temps bootstrap, coût tx, blocksize
Dinh et al. (2017)	Ethereum, Parity, HLF v0.6	⊠	⊠	⊠	⊠	RAM	tolérance aux fautes
Dong et al. (2019)	Iota, Nano, Byteball	⊠	⊠	⊠	⊠	CPU, RAM, réseau	blocksize
Pongnumkul et al. (2017)	Ethereum, HLF v0.6	⊠	⊠	⊠	□	∅	∅
Foundation (2022)	famille HL, Ethereum	□	⊠	⊠	□	CPU, RAM, réseau	∅
Zheng et al. (2018)	Ethereum, Parity, HLF, CITA	□	⊠	⊠	□	CPU, RAM, réseau	Tps consensus, Taux de propagation, tps de MAJ, ...
Baliga et al. (2018)	HLF v1.0	⊠	□	⊠	□	CPU	nombre de channel/chaincode
Benahmed et al. (2019)	Ethereum, EOS, HL Sawtooth	⊠	□	⊠	⊠	CPU, RAM	∅
Han et al. (2020)	HLF v0.6 & v1.0, Ripple, Tendermint, Corda	⊠	□	⊠	⊠	∅	débit requête
Rouhani et Deters (2017)	Ethereum, Parity	⊠	□	⊠	□	RAM	∅

TABLEAU 2.1 – État des lieux de la mesure de performances des blockchains

monitoring consiste à évaluer les performances du système public en exploitation sous une charge de travail réaliste. Zheng et al. (Zheng et al., 2018) ont proposé un cadre détaillé de surveillance de la performance en temps réel en utilisant une approche basée sur les journaux par un appel de procédure à distance (RPC). Les métriques employées sont relativement équivalentes à un *benchmark* et correspondent au calcul des débits et latences.

2.1.4 Bilan des métriques mesurées

Le tableau 2.1 regroupe l'ensemble des articles scientifiques trouvés dans la littérature et cités dans les sections précédentes, auquel sont ajoutés les articles de Baliga et al. (Baliga et al., 2018), Han et al. (Han et al., 2018, 2020) et Rouhani et Deters (Rouhani & Deters, 2017), qui tous trois proposent des méthodes de mesures de performances sur diverses blockchains.

Si l'on examine quelles technologies de blockchain ont été prises en compte dans les études rapportées (voir la colonne "Plateforme" dans la table 2.1), Ethereum et Hyperledger Fabric (*HLF*) sont les solutions les plus largement adoptées (la première étant utilisée dans dix études, la seconde dans six). Ce n'est pas surprenant, car elles sont toutes deux leaders en termes de parts de marché (50% des projets mis en œuvre étant hébergés sur ces plateformes (Udokwu

et al., 2018)) et qu'elles sont en accès libre. La comparaison de ces deux plateformes est intéressante, bien qu'elle ne présente pas des objectifs similaires. En effet, alors qu'Ethereum est un représentant de la catégorie *Proof-based* et est plus adapté aux grands livres distribués sans permission, Hyperledger représente la famille *Voting-based* et est plus adapté aux blockchains avec permission. La notion de scalabilité devient essentielle pour comparer ces deux approches, puisque, comme le montrent les études de cette liste prenant en compte la scalabilité, Hyperledger dispose de meilleures performances qu'Ethereum, mais son passage à plus grande échelle fait effondrer ces performances.

La blockchain Hyperledger Fabric est étudiée dans les différents articles sous deux versions : souvent la version 0.6 et plus occasionnellement, la version 1.0. Il s'agit en fait de deux approches différentes. La version 0.6 de Hyperledger Fabric implémente le consensus PBFT alors que la version 1.0 n'implémente pas PBFT (mais Raft, qui est un consensus CFT et non BFT) et repose sur une autre approche d'ordonnancement de la blockchain, amenant ainsi des résultats différents (Hors, 2019).

Par la suite, on peut observer dans le tableau 2.1 (voir colonne "Réalise expériences") que près de la moitié des études examinées ont effectué des évaluations expérimentales de leurs solutions. Cette observation émane de la composition du corpus proposé. Ainsi, les trois premiers articles de la liste sont des articles de comparaisons théoriques des consensus. Ils comparent un grand nombre de consensus d'un point de vue théorique et donc sans proposer de cadres d'études pratiques de performances de blockchains. De plus, certaines métriques sont proposées et sont hors de propos pour certains protocoles (coût d'une transaction pour des blockchains sans cryptomonnaies, récompense si ajout d'un bloc ...). Les propositions de Foundation et de Zheng et al. ne proposent pas de résultats alors qu'ils définissent beaucoup de métriques applicables sur différends de consensus. La raison est que ces propositions proposent des frameworks pour surveiller les blockchains sans pour autant avoir un objectif de comparaisons de chaînes. L'étude de scalabilité n'est pas proposée dans leurs approches, bien qu'il s'agisse d'une métrique clivante lors de la comparaison de blockchain et que les frameworks proposés pourraient l'étudier en changeant le nombre de nœuds. Dans le cadre d'articles présentant des écosystèmes dans lesquels s'intègre la blockchain, peu d'évaluations sont proposées, car l'accent est davantage mis sur les choix de conception architecturale et fonctionnelle que sur l'évaluation comparative des performances (Leduc et al., 2021).

Pour l'ensemble des études proposées, le débit et la latence sont les paramètres les plus utilisés puisque dans la quasi-totalité de la littérature examinée. Une comparaison est souvent réalisée avec les transactions Visa ($20000 tx/s$), pour montrer que les transactions blockchains (ex. : Bitcoin avec $7 tx/s$) ne peuvent supplanter le système bancaire actuelle (Bamakan et al., 2020). Le débit correspond au nombre de transactions réussies par seconde (une transaction étant réussie si elle a été validée et engagée sur un nouveau bloc) ; la latence correspond au délai entre l'émission d'une transaction et son engagement sur un nouveau bloc. La définition du débit peut être amenée à évoluer, pour prendre en compte le délai nécessaire pour propager le nouveau bloc qui impacte la terminaison des consensus à base de preuve (Foundation, 2022).

On peut observer que seul un petit nombre d'études a évalué les aspects de sécurité. La principale raison est que le niveau de sécurité d'une technologie blockchain donnée est défini mathématiquement à partir du protocole de consensus supportant la chaîne. Par exemple, dans un consensus de type preuve de travail, le nombre de mineurs honnêtes doit être supérieur à 51% ; ce nombre doit être égal à 67% dans les algorithmes de consensus de type *Byzantine Fault Tolerant* (BFT).

D'autres métriques, liées à la consommation des ressources matérielles, sont régulièrement

utilisées, afin de chercher les goulots d'étranglement des différents consensus. Dans le cadre des preuves de travails, le goulot est au niveau du CPU alors que dans le cas des consensus à base de vote, il est relatif à la bande passante, particulièrement dans le cas où le nombre de nœuds augmente.

Comme explicité dans les sections précédentes, la moitié des articles présentés développent un modèle en plusieurs couches de la blockchain. La mise en place de ce modèle permet de développer des métriques ciblant certaines couches, par exemple le consensus. Ainsi, Dinh et al. définissent plusieurs types de *smart contracts* différents et mesurent le débit de ces divers contrats, qui donnent des temps d'exécutions différents. Cette idée sera reprise par d'autres articles.

De manière générale, la littérature actuelle ne s'attache pas suffisamment à analyser correctement la mesure dans laquelle un choix de conception architecturale donné peut influencer les limites de capacité à long terme (en transactions/seconde Tx/s) du système proposé, et donc par définition, sur la QoS (*Quality of Service* ou qualité de service en français) globale (de bout en bout). Dans notre corpus, seuls Croman et al. et Geyer et al. cherchent à déterminer l'influence de la taille du bloc sur le débit. Par exemple, la latence dépend directement du débit et peut être influencée négativement par un délai élevé de propagation des transactions. En outre, si le débit est inférieur au taux de demande de transaction, une congestion est susceptible de se produire, ce qui entraîne une augmentation de la latence. De telles interactions entre les paramètres liés à la blockchain et à l'infrastructure sont rarement analysées et prises en compte dans la littérature, ce qui nécessite des recherches supplémentaires.

2.2 Solutions d'optimisation des performances de blockchain

La section 2.1 a mis en évidence les problématiques de mesure des performances des blockchains. La connaissance de ces performances pour un environnement donné facilite d'une part la sélection d'une blockchain adaptée lors de la conception d'une nouvelle architecture intégrant cette technologie. D'autre part, elle permet d'identifier les différents goulots d'étranglement au sein des architectures empêchant les solutions d'atteindre les critères de qualité de service requis. Une fois ces goulots localisés, il est alors possible de chercher des solutions pour pouvoir optimiser les performances des applications.

La littérature scientifique fournit également de nombreuses possibilités d'optimisation en fonction de ces goulots. Considérant le modèle en 5 couches proposé en figure 2.1, des optimisations peuvent être proposées dans chacune des couches dans le cas où elles se trouvent limitantes. Des solutions pour chacune de ces couches sont proposées et certaines seront décrites par la suite :

— Application : la section 2.2.1 propose quelques propositions d'améliorations dans le cas où l'architecture de l'écosystème est source de diminution de performances. Dans ce cas précis, des modifications de l'architecture sont proposées, notamment grâce à du l'apprentissage automatique (*Machine Learning* en anglais) (Miglani & Kumar, 2021) ;

— Exécution : dans le cas des *smart contracts*, des améliorations peuvent être réalisées sur la machine exécutant le contrat. Ainsi, dans le cas d'Ethéreum, des améliorations peuvent être réalisées sur la machine virtuelle EVM (Hughes et al., 2021) ou sur le compilateur transformant le contrat en bytecode (Peng et al., 2019) ;

— Données : maintenir une unique base de données cohérente nécessite beaucoup de ressources. Des propositions avancent que, grâce à des codes d'effacement, des mineurs avec des capacités de stockage plus faible peuvent quand même participer au consensus en

éclatant les blocs sur plusieurs nœuds (Perard et al., 2018). Pour aller plus loin, les responsabilités des données peuvent également être éclatées en plusieurs groupes pour réduire la charge requise sur un nœud. C'est le *sharding*, qui sera évoqué dans la section 2.2.2 ;

— Consensus : le consensus est souvent le goulot d'étranglement d'une blockchain, particulièrement dans le cas d'un consensus basé sur un vote que l'on charge à distribuer sur plus de nœuds. Des propositions pour accélérer ce consensus sont présentées en section 2.2.3.

— Réseau : si le débit de propagation limite les performances de la chaîne, il peut être alors intéressant de mettre en place un réseau pair-à-pair plus efficace, en s'appuyant par exemple sur des réseaux structurés comme ceux présentés en section 1.4.2.

2.2.1 Optimisation d'architectures

La blockchain est utilisée dans de nombreuses applications pour faire le lien entre différents acteurs d'un système. De nombreux domaines exploitent ce paradigme dans des domaines aussi variés que ceux décrits dans l'introduction de ce chapitre 2. Les performances applicatives de la blockchain sont donc confondues avec les performances du système. Ainsi, dans le cas où la couche application est identifiée comme le goulot d'étranglement, il convient d'optimiser la gestion des interactions ou des performances des agents du système. L'étude réalisée par Miglani et Kumar présente ainsi de multiples propositions d'optimisation d'architecture basée sur une blockchain et optimisée par apprentissage automatique.

Dans les réseaux IoT et dans les réseaux de *mobile edge computing*, il est courant de réaliser du *Computation Offloading* afin de décharger sur une plateforme externe des calculs pour profiter de plus de ressources matérielles. Ce procédé nécessite de mettre en place une base de données centralisée, sécurisée et partagée par tous les appareils IoT afin que chaque appareil puisse connaître les ressources disponibles dans le réseau et puisse réaliser une réservation de ces ressources. La blockchain est donc une solution privilégiée pour permettre une communication décentralisée entre les appareils en bout et le cloud à travers des *smart contracts*. Cependant, une association aléatoire entre les serveurs du cloud et les demandes implique une perte d'énergie, de temps et de ressources. Des propositions ont ainsi été réalisées pour associer *Machine Learning* et optimisation de l'*edge computing*. Ainsi, C. Xu et al. propose le développement de *smart contracts* exploitant le *Deep Reinforcement Learning* (ou *DRL*) pour minimiser les coûts énergétiques dans la migration vers le cloud, avec des gains énergétiques de l'ordre 30% (C. Xu et al., 2017). L'utilisation de DRL dans des *smart contracts* a également été utilisée par Dai et al. pour optimiser cette fois le cache d'un ensemble de *datacenter* composant un cloud dans le cadre de l'*edge computing* pour des appareils IoT communicants en 5G de manière décentralisée (Dai et al., 2019). Outre les délais réduits et les ressources mieux réparties, l'utilisation de DRL dans des *smart contracts* permet également d'améliorer la confidentialité des utilisateurs, comme le montre l'exemple de (D. C. Nguyen et al., 2020). D'autres modèles d'optimisations peuvent également être adoptés, comme les modèles bayésiens (Asheralieva & Niyato, 2021) ou les algorithmes génétiques (Qiu et al., 2019).

2.2.2 *Sharding*

La tenue d'une base de données unique implique que les transactions doivent se réaliser les unes à la suite des autres sur tous les réplicas, sans possibilité d'exécutions parallèles, ce qui limite fortement le débit. Il s'agit donc d'un défaut de la couche de données. La méthode de

Sharding, utilisé actuellement dans des bases de données partitionnées Spanner (Corbett et al., 2013), peut être exploitée dans les blockchains pour améliorer fortement le débit.

Le sharding est rapidement étudié au cours de cette section, car les protocoles développés dans les parties suivantes s'intègrent parfaitement dans le cas de blockchain exploitant le sharding. Cependant, nous n'adopterons qu'une application au sein des blockchains. L'intégration au sharding fait donc partie des perspectives, car elle impose d'autres problématiques non traitées dans cette thèse.

Définition du *sharding*

Dans ce paradigme, la base de données supportée par la blockchain est divisée en plusieurs fragments indépendants appelés *shards* et chacun de ces shards est laissé à la responsabilité d'un sous-ensemble de nœuds. Dès lors, dans le cas d'une exécution d'une transaction, le consensus ne s'applique pas sur l'entièreté des nœuds, mais uniquement dans le sous-ensemble associé au shard. Dans le cas où une transaction concerne des données sur des shards différents, un consensus particulier est appliqué. Le sharding peut donc être vu comme une "blockchain de petites blockchains".

Ce partitionnement permet donc une exécution en parallèle des transactions, améliorant le débit d'ajout des transactions, puisque deux transactions appliquant des modifications sur des shards différents peuvent s'exécuter en même temps. Les capacités et débits d'ajout en transactions augmentent donc linéairement avec le nombre de shards, ce qui implique donc, dans le cas où les shards sont formés d'un nombre de nœuds fixe, que le débit augmente linéairement avec le nombre de nœuds (Yu et al., 2020). Ainsi, le débit moyen des blockchains réalisant du sharding présenté dans le tableau 2.2 sont plus proche de l'objectif des transactions Visa ($20000 tx/s$) (Bamakan et al., 2020).

Le sharding est donc particulièrement pertinent dans le cadre de chaînes publiques sans permission, qui sont donc fortement sujettes à être déployées à grande échelle que dans le cas de blockchain privé, tandis que les blockchains privées avec permission ne profitent pas de ces avantages puisqu'elles se déploient dans des réseaux plus restreints. De plus, le sharding implique une diminution de la décentralisation, ce qui est plus problématique dans les réseaux restreints.

Problématiques associées

À cause de leur architecture, les blockchains réalisant du sharding doivent réaliser deux consensus, répondant à des problématiques différentes : l'*Intra-Consensus-Safety* et la *Cross-Shard-Atomicity*. Le tableau 2.2 présente une réponse à ces deux problématiques par des chaînes réalisant du sharding.

Intra-Consensus-Safety Au sein d'un shard, un consensus a lieu pour pouvoir accepter et ajouter une transaction entre les différents membres du shard. Les problématiques associées sont les mêmes que celles discutées en section 1.4.4 et l'on retrouve donc des consensus dérivés de consensus basés sur des preuves ou des votes. Ainsi, parmi les 6 chaînes représentées sur le tableau 2.2, 5 réalisent un consensus BFT et 2 utilisent la preuve de travail. La surreprésentation des protocoles BFT s'explique par la taille des shards. Plus petits, les protocoles BFT peuvent s'appliquer sans perdre leurs performances à cause d'une trop grande mise à l'échelle.

Les shards, à la différence des blockchains classiques, sont sujets à l'attaque des 1%. Si un shard est responsable de 1% du réseau, alors un attaquant n'a besoin que de corrompre ces 1% pour contrôler entièrement un shard et les données associés. L'attaque est donc plus facile que sur

Nom de la chaîne	Protocole Intra-Consensus	Méthode pour Cross-shard	Débit
Monoxide (J. Wang & Wang, 2019)	PoW-based Chu-ko-nu mining	Lock-free	$1.23 \sim 2.56M$ tx/s
Elastico (Luu et al., 2016)	PBFT	$\emptyset$	$48k$ tx/s
Omniledger (Kokoris-Kogias et al., 2018)	dérivé de ByzCoin (PoW et PBFT)	Lock/Unlock	$28.8k$ tx/s
RapidChain (Zamani et al., 2018)	50% BFT	Lock/Unlock	$128k$ tx/s
Ethereum 2.0 (Buterin, 2020)	BFT-based PoS	Lock/Unlock	$134k$ tx/s
Chainspace (Al-Bassam et al., 2018)	MOD-SMART (PBFT)	Lock/Unlock	< 400 tx/s

TABLEAU 2.2 – Comparaison de blockchains réalisant du sharding, tiré de Yu et al.

une blockchain classique. Le consensus interne au shard doit donc répondre à ce genre d'attaque. Pour sécuriser et empêcher la prise d'un shard par un ensemble de nœuds malveillant, le protocole de sélection des nœuds associés à un shard doit sélectionner ces nœuds aléatoirement de manière sécurisée de telle manière à ce que le malveillant ne puisse pas introduire un biais le favorisant lors de la sélection. Dans le cadre des blockchains BFT, les nœuds doivent régulièrement être échangés entre différents shards pour éviter la prise de contrôle par des adversaires s'adaptant aux réseaux (Yu et al., 2020).

La taille des shards a un fort impact sur ce consensus. En effet, augmenter la taille d'un shard améliore ainsi la sécurité puisque davantage de ressources sont nécessaires pour un attaquant afin de prendre le contrôle d'un shard. Cependant, cette augmentation augmente la latence des réponses de ce shard. En effet, même si des consensus BFT linaire et fortement scalable peuvent être appliqué comme dans Omniledger ou Ethereum 2.0, le temps nécessaire pour accepter une transaction augmente toujours avec le nombre de shards. Un compromis doit donc être trouvé sur la taille de ces shards. Le protocole proposé dans le chapitre 4 pourrait donc résoudre dynamiquement ce compromis en fonction des débits de transactions proposés.

Cross-Shard-Atomicity Si les transactions ne faisant intervenir que des ressources au sein d'un même shard ne posent pas problème, car les problématiques liées à leur exécution sont identiques à celle d'une blockchain, les transactions liant plusieurs shards sont beaucoup plus problématiques. En effet, en séparant les données en fragments pour améliorer le débit, la cohérence n'est plus maintenue sur l'ensemble du réseau, mais seulement au sein des shards. Ainsi, lors de l'exécution d'une transaction entre fragments, le nœud réalisant l'exécution doit connaître l'état des shards associés à la transaction, réaliser l'exécution et envoyer le nouvel état aux shards, sachant que ces shards peuvent être sollicités en parallèle, donc que leurs états ne correspondent plus à celui reçu par le nœud exécutant la transaction. Des mécanismes de verrouillage doivent donc être utilisés pour maintenir l'état des shards durant l'exécution de la transaction afin de garantir l'atomicité de cette transaction (cf. 1.4.5).

De tels mécanismes sont bloquants ce qui suppose que l'on travaille dans un réseau synchrone et que l'on peut donc détecter si le nœud orchestrant la transaction entre fragments est défaillant. Dans le tableau 2.2, Omniledger, RapidChain, Ethereum 2.0 et Chainspace adoptent ce type de mécanisme dans leurs implémentations. Monoxide, quant à lui, suppose que les transactions entre fragments sont asynchrones, permettant la réalisation d'opérations sans verrouillage au prix d'une incompatibilité avec les *smart contracts*, d'une latence supplémentaire et d'une répétition

de messages.

En conclusion, le sharding est une proposition efficace pour augmenter les performances des blockchains en éclatant en plusieurs fragments les données, et en permettant une exécution parallèle des transactions. Par ailleurs, cette technologie impose de fortes contraintes pour assurer l'atomicité de l'exécution des transactions si cette dernière concerne plusieurs shards. La problématique du consensus au sein d'un shard reste cependant la même que pour une blockchain de taille réduite. Une analyse et une optimisation des paramètres associés comme proposés dans cette thèse sont bénéficiaires à cette problématique.

2.2.3 Cas des chaînes BFT : optimisation par sélection des validateurs

Dans le cadre des blockchains BFT, le nombre de messages nécessaire pour parvenir est généralement quadratique, c'est-à-dire de l'ordre de $O(n^2)$. Des protocoles plus récents ont été développés, basés sur l'utilisation de signature à seuil comme LinSBFT (Qi et al., 2020). Cependant, étant quand même linéaires avec le nombre de validateurs. Les performances diminuent également lorsque l'on souhaite augmenter le nombre de nœuds dans un réseau.

Une solution pour augmenter le nombre de nœuds dans le réseau est de diviser l'ensemble de ces nœuds en deux sous-ensembles disjoints :

— les *validateurs*. Cet ensemble de nœuds réalise le protocole BFT et ordonnance l'ensemble des transactions soumissent dans des blocs. Leurs décisions sont ensuite diffusées au second sous-ensemble ;

— les *non-validateurs*. Cet ensemble de nœuds récupère le résultat des validateurs pour mettre à jour leur blockchain. Ils ne participent pas au consensus, mais peuvent relayer bloc et transactions à l'ensemble du réseau.

La figure 2.2 montre un exemple de partitionnement d'un réseau composé de validateurs et non-validateurs. Une fois que ce partitionnement est créé, un protocole de communication dédié aux validateurs peut être mis en place. En effet, comme l'ensemble de ces nœuds est relativement petit et que le nombre de messages est conséquent à cause du consensus, il peut être intéressant de mettre en place une architecture complète pour réduire le délai d'acheminement des messages.

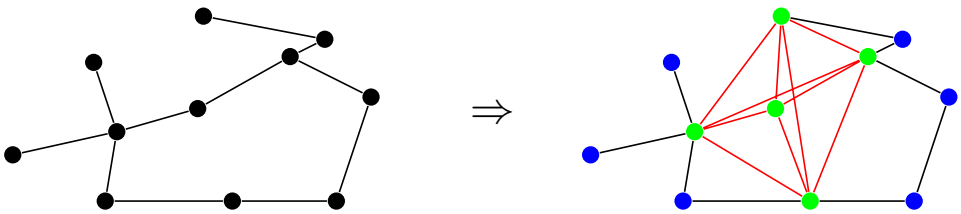


FIGURE 2.2 – Exemple de partitionnement en validateurs et sous-validateurs. Les nœuds de couleur bleue sont non-validateurs et ceux de couleurs sont des validateurs.

La problématique principale dans cette division en deux sous-ensembles est la définition des critères pour réaliser cette sélection. La section 2.2.3 présente une sélection des validateurs grâce à des outils de *Machine Learning* pour sélectionner les validateurs les plus réactifs du réseau. Cependant, afin que la chaîne ne puisse pas être contrôlée par un attaquant ayant manipulé les critères de sélection, une sélection aléatoire des validateurs est nécessaire. La section 2.2.3 présente un ensemble de chaînes sécurisant cette sélection aléatoire grâce à un protocole basé sur une preuve.

Adaptation des paramètres et sélection des validateurs par DRL

Un réseau concret est généralement composé de nœuds non homogènes. Certains nœuds répondent donc plus vite que d'autres ou disposent de liens avec une bande passante plus élevée, permettant un envoi plus rapide des messages. Dans le cas d'un consensus par vote, il est intéressant de favoriser la sélection de ces nœuds en tant que validateur puisqu'étant plus réactif et rapide. Dans ce cas, le consensus se réalisera plus rapidement, améliorant le débit et les performances.

Ainsi, M. Liu et al. propose dans leur article une adaptation en temps réel des paramètres de la blockchain (M. Liu et al., 2019). Les auteurs mettent en place une sélection des validateurs à l'aide d'une optimisation par DRL. Des réseaux de neurones sont donc utilisés pour sélectionner un ensemble de paramètres A minimisant le système linéaire 2.1 pour un réseau dans un état S.

$$\begin{aligned} \max_{A} \quad & Q(S, A) \\ \text{s.t.} \quad & G(\Upsilon) < \eta_s; G(\lambda) < \eta_l \\ & T^F \leq \omega \cdot T^I \end{aligned} \tag{2.1}$$

L'état du réseau S prend en compte la taille des transactions, le nombre de fois Υ que les nœuds ont été validateurs, leurs localisations géographique lié à un paramètre λ, leurs capacités de calcul et le taux de transmission avec chacun de leurs voisins. Le réseau de neurones chercher donc à sélectionner le meilleur ensemble de paramètres A qui maximise le débit en nombre de transactions. Ces paramètres sont composés des validateurs sélectionnés, de la taille du bloc et du temps d'attente T^I nécessaire pour former un bloc.

Le système ajoute également deux contraintes : l'une de décentralisation et une autre de latence. Afin d'imposer un changement régulier des nœuds et d'éviter une centralisation dans une petite zone géographique, des contraintes de répartitions sur le nombre de fois validateur et spatiale sont imposés avec les valeurs η_s et η_l, grâce à la mesure d'indices de Gini. Cette mesure statistique permet de rendre compte de la répartition d'une variable, 0 indiquant l'égalité et 1 l'inégalité parfaite. Par la suite, l'ajustement de la taille, de l'intervalle des blocs et la sélection des producteurs de blocs les plus efficaces en calculs amène à un compromis que la seconde contrainte permet de trouver dans un temps réalisable T^F.

Les résultats avancés par M. Liu et al. montrent des augmentations de 30% du débit et une baisse de la latence de 50%, comparé à une sélection statique des nœuds. Ces résultats sont très significatifs, mais soulèvent d'autres problématiques. Ainsi les auteurs n'expliquent pas qui doit réaliser le calcul du nouvel état ni comment la proposition est transmise à l'ensemble du réseau. Dans le cas où un serveur central exécute le framework, il s'agit d'un tiers de confiance. Dans le cas où les nœuds doivent exécuter le framework, celui-ci se base sur *PyTorch* , librairie qui peut être gourmande en ressource et non adaptée à tous les usages IoT.

De plus, l'entraînement du réseau de neurones nécessite une grande base d'entraînement. Supposons que l'on souhaite sélectionner 20 validateurs parmi 50 nœuds, la base d'entraînement doit contenir au moins une fois l'occurrence de chacun de sous-groupe possible, nécessitant $\binom{50}{20} = 4,7 \cdot 10^{13}$ échantillons.

Par ailleurs, la sélection des validateurs étant déterministe et prédictible, cette proposition n'est pas à l'abri d'un attaquant qui arrive à se faire sélectionner comme nouvel ensemble de validateurs en manipulant les variables d'entrées de l'algorithme.

TABLEAU 2.3 – Liste de blockchain avec un consensus hybride basé sur BFT

Chain Name	**When**	**Whom**	**How Many**
Albatross (Berrang et al., 2019)	Tours de tailles fixes	PoS + VRF	
Algorand (Gilad et al., 2017)	Chaque tours	PoS + VRF	$> 500k$
ASHWAChain (Arora et al., 2020)	Chaque commit de PoW	PoW	
ByzCoin (Kogias et al., 2016)	Tours de tailles fixes	PoW	1004 (eval)
DRBFT (Zhan et al., 2021)	//	PoS + VRF	$10M$
Gosig (Li et al., 2020)	Chaque tours	VRF	$> 10k$
Imporoved PoS (Wu et al., 2020)	Échec du consensus	PoS + VRF	100
LinSBFT (Qi et al., 2020)	Tours de tailles fixes	PoS	64
Tendermint (Buchman et al., 2018)	Chaque tours	PoS	125
Sawtooth (Hyperledger, 2022)	//	//	

Hybrid BFT based consensus

Dans les plateformes blockchain basées sur la BFT comme Hyperledger ou Sawtooth-PBFT (Hyperledger, 2022), la sélection des validateurs est généralement définie lors de la mise en place de la plateforme, ce qui entraîne un ensemble de validateurs fixe tout au long de l'utilisation opérationnelle. Ce type de plateforme/stratégie est communément appelé *Consortium Blockchain* (Belotti et al., 2019), qui s'éloigne d'une solution blockchain véritablement décentralisée, et dans les grands réseaux, le consensus devient presque équivalent à un PoA (Proof-of-Authority). Les autorités centrales comme les banques (Consensys, 2021) sont souvent sélectionnées comme validateurs pour assurer la confiance du comité de validateurs. Si le pool de validateurs doit être mis à jour, l'intervention de l'autorité centrale est requise avec des négociations fermées entre les validateurs.

Afin d'adopter plus de flexibilité dans le choix des validateurs, une autre stratégie consiste à sélectionner un ensemble aléatoire de validateurs après une période de temps spécifique, empêchant l'émergence d'un tiers de confiance, comme c'était le cas avec la première stratégie. Ce sont les consensus hybrides basés sur BFT.

La principale condition à remplir est que tous les nœuds doivent se mettre d'accord sur le prochain groupe de validateurs, et c'est là que les protocoles basés sur la preuve entrent en jeu. Une sélection partielle de stratégies de pointe est présentée dans le tableau 2.3, lequel est présenté plus en détail dans ce qui suit. Le tableau 2.3 présente plusieurs chaînes avec : le moment où l'ensemble des validateurs est modifié, les protocoles associés (détaillés dans les paragraphes suivants) et le nombre de nœuds avec lesquels la solution a été testée (dans l'article associé).

PoW Le protocole PoW (*Proof-of-Work*) évite les attaques Sybil en garantissant que l'adhésion et la participation au protocole sont limitées par la capacité du matériel (Nakamoto, 2008). ByzCoin (Kogias et al., 2016) et ASHWAChain (Arora et al., 2020) utilisent ce protocole pour sélectionner un noyau de validateurs afin de protéger la chaîne de ce type d'attaques. Ils ont proposé de fusionner les protocoles PoW et PBFT (l'ajout d'un nouveau bloc dans la chaîne PoW donne le droit de prendre part au consensus dans la chaîne PBFT). Dans Byzcoin, une approche par fenêtre glissante élimine les validateurs de la chaîne PBFT afin de conserver un nombre fixe de validateurs, représentés statistiquement en proportion de leur puissance de hachage.

PoS La sélection des validateurs sur la base de la PoW est gourmande en énergie. Pour surmonter ce problème, la méthode PoS (*Proof-of-Stake*) peut être appliquée à la place, comme le proposent Buchman et al. avec Tendermint (Buchman et al., 2018). Dans leur approche, chaque nœud qui souhaite prendre part au consensus doit effectuer un dépôt de sécurité en utilisant la cryptomonnaie de la chaîne, et les 125 premiers nœuds avec les dépôts les plus élevés sont ensuite sélectionnés et inclus dans le pool de validateurs. Le *proposer* de chaque époque est sélectionné selon un algorithme déterministe round-robin, qui sélectionne le *proposer* en fonction des poids relatifs des dépôts. Si un nœud non-validateur propose un enjeu supérieur à l'enjeu le plus bas, il le remplacera. La blockchain LinSBFT présentée dans (Qi et al., 2020) adopte une stratégie PoS similaire.

VRF L'aléatoire est un problème critique dans les systèmes distribués sécurisés. Comme les blockchains sont des grands registres distribués sécurisés et déterministes, tous les nœuds doivent se mettre d'accord sur un mécanisme impartial dans un environnement déterministe, ce qui ne laisse pas beaucoup de place à l'aléatoire. Les fonctions aléatoires vérifiables (*Verifiable Random Function* ou *VRF*) (Goldberg et al., 2018) sont des fonctions pseudoaléatoires fournissant une preuve publiquement vérifiable de l'exactitude de la sortie. Elles sont utilisées pour simuler un mécanisme pseudoaléatoire qui peut être utilisé pour sélectionner des validateurs. Une VRF doit garantir les propriétés suivantes : unicité, résistance totale aux collisions, caractère pseudoaléatoire et vérifiabilité publique, propriétés très proches des fonctions de hachages (cf 1.1.1). L'algorithme (Gilad et al., 2017) est basé sur un consensus PPoS (*Pure Proof-of-Stake*), qui combine PoS et VRF. Afin d'éviter la situation des protocoles PoS où les jetons sont concentrés dans un petit groupe, 'The rich get richer', PPoS est basé sur *cryptographic sortition*. À chaque tour, une VRF est calculée à partir d'une graine partagée basée sur le bloc précédent et la sortie pseudoaléatoire est utilisée pour sélectionner le validateur du nouveau tour. Afin de contrer l'attaque Sybil, la sélection du validateur est pondérée par la possession de la cryptomonnaie intégrée. Avec ce système, même les nœuds les plus pauvres ont la possibilité de prendre part au consensus. La blockchain Albatros (Berrang et al., 2019) utilise un processus similaire, où un nombre aléatoire obtenu à l'aide d'une VRF est utilisé pour sélectionner une liste de validateurs sur la base d'une liste de nœuds pondérée par les jetons détenus par ce nœud. Wu et al. proposent un protocole similaire, mais utilisent une métrique basée sur les performances du matériel et du réseau au lieu d'un jeton de cryptomonnaie (Wu et al., 2020). Comme Algorand, Gosig (Li et al., 2020) utilise une VRF pour sélectionner une liste de *proposers potentiels*, mais avec la différence que tous les nœuds ont le même poids pour être sélectionnés. Enfin, dans DRBFT (Zhan et al., 2021), un système de vote est présent entre les nœuds et une VRF est utilisée pour sélectionner un ensemble de validateurs parmi les nœuds ayant le plus grand nombre de votes.

Toutes les solutions de blockchain présentées précédemment sont basées sur l'idée qu'un petit groupe de validateurs est plus rapide qu'un grand groupe, mais l'état actuel des choses ne met pas en évidence une blockchain avec une taille variable du groupe de validateurs. Comme nous l'avons expliqué précédemment, les performances d'une blockchain BFT sont intrinsèquement liées à la taille du groupe de validateurs. À notre connaissance, aucune étude n'a jamais proposé un protocole de consensus de blockchain permettant l'autoadaptation de la taille du pool dans le temps afin de répondre au débit d'entrée.

2.3 Positionnement

L'étude de l'état de l'art nous montre les difficultés à évaluer les performances d'une blockchain. La performance principale mesurée est le débit et la latence des transactions, mais d'autres mesures sont parfois utilisées en second plan et sont souvent définies en rapport avec le consensus ou les choix de conceptions. En revanche, les performances mesurées sur une blockchain ne sont pas universelles. La spécificité du réseau de nœuds rend impossible la reproductibilité exacte des résultats sur un autre plan d'expérience.

La mesure des performances d'une application, qui bien qu'elle ne soit pas aussi précise qu'un *benchmark* mesurant les différentes briques constituant une blockchain, doit ainsi prendre en compte ces différentes spécificités. Une méthodologie stricte doit donc être pensée pour mesurer les performances d'une application afin d'éviter des performances uniquement valables dans des environnements et contextes spécifiques. Nous proposerons une telle méthodologie au cours du chapitre 3.

Dès lors que les performances de l'application sont mesurées, il convient de se poser la question de comment améliorer les performances. Les mesures de performances des blockchains montrent que le consensus est l'étape réduisant le plus les performances des applications, particulièrement dans le cas des consensus BFT. On relève également que le paramètre perturbant le plus les performances d'une blockchain est le passage à l'échelle. En effet, les performances des blockchains basées sur un vote s'effondrent lorsque le réseau accueille plus de nœuds.

Afin d'améliorer les performances de ces consensus, des modifications aux protocoles ont été proposées. La modification la plus profonde est le sharding, qui en séparant les données sur plusieurs groupes d'utilisateurs, permet d'accélérer fortement le débit. Cependant, ce choix d'architecture amène des problématiques, notamment de sécurité et de cohérence, et est principalement valable sur des chaînes publiques avec un très grand nombre d'utilisateurs. Dans le cas d'un réseau privé plus restreint, mais quand même conséquent, de l'ordre d'une centaine d'utilisateurs, la meilleure solution pour exploiter un consensus BFT reste une séparation des nœuds en deux sous-ensembles. Des protocoles existent donc pour sélectionner de manière sécurisée les validateurs, en exploitant un second consensus à base de preuve, mais la question du nombre de validateurs est laissée à la discrétion de l'administrateur de ces réseaux. Pourtant, les protocoles BFT sont très sensibles à ce paramètre, et le choix d'un nombre implique des performances maximales à une application qui ne sera pas toujours valable dans le temps. Un ajustement dynamique de ce paramètre permettrait d'améliorer la cohérence et la sécurité du réseau en fonction des performances demandées. Nous proposerons une adaptation de ce type dans le chapitre 4.

Enfin, des propositions existent afin d'accélérer le consensus en sélectionnant les validateurs les plus performants, mais les outils demandés impliquent une perte de décentralisation ou requièrent des ressources de calculs que ne disposent pas forcément les nœuds du réseau. Une sélection plus rapide est efficace des validateurs est donc souhaitable. Nous proposerons ainsi deux algorithmes réalisant ce type de sélection dans le chapitre 5.

2.4 Conclusion

Pour conclure ce chapitre, nous avons pu voir que la mesure des performances d'une blockchain peut se réaliser de différentes façons à travers des *benchmarks* et du *monitoring*. Ces différentes études permettent de déterminer les paramètres agissant comme goulots d'étranglement qui limitent les performances des applications. Une fois ceux-ci localisés, des optimisations peuvent

être mises en place. Le consensus est identifié dans la majorité comme goulot d'étranglement. Des alternatives aux consensus usuelles ont donc été développées pour répondre aux attentes de passage à l'échelle et de débit.

La mesure de performances des consensus détaillés en section 2.1 de ce chapitre s'attarde sur l'étude de la configuration de la chaîne à travers des contrats standards. L'application n'est pas mesurée alors qu'elle fait appel à des contrats dont le temps d'exécution est différent. Une adaptation de ces *benchmarks*, cette fois dédié à l'étude de l'application, est réalisée dans le chapitre suivant.

3

Une première proposition autour de l'évaluation des *Smart contracts*

Le chapitre précédent a détaillé un ensemble de métriques et de stratégies d'évaluations de blockchains. Ces mesures se centraient principalement autour de la mesure du débit et de la latence de la chaîne dans des conditions fixes, afin de mesurer chaque brique composant la blockchain. Dans le cadre de l'évaluation d'une application, il n'existe pas de plan de mesure objectif des performances de cette application. En effet, les performances d'une blockchain dépendent de nombreux paramètres, dont ceux liés à l'implémentation des *smart contracts* constituant l'application. Afin d'aider les intégrateurs de solutions logicielles à mieux comprendre les enjeux de performances, nous proposons une méthodologie complète pour mesurer l'impact d'une configuration sur la qualité globale des performances du service à long terme.

La première étape de cette méthodologie consiste en l'identification des liens entre les différents paramètres de l'application. Ces paramètres disposent de plusieurs influences sur les mesures de performances QoS. Aussi, il est important de noter quels paramètres sont adaptables pour rejoindre les performances souhaitées. La section 3.1 se chargera de présenter les interactions entre les différents constituants modifiables de la chaîne.

À partir de ces liens, une méthodologie est ainsi mise en place pour évaluer rigoureusement les contrats constituant une application. Cette méthodologie est présentée en section 3.3. Il en résultera une métrique capitale dans l'évaluation d'un contrat : la *capacité*. L'impact des contrats sur cette capacité sera également mesuré.

Ces différentes mesures se feront dans le contexte du *Smart Farming*. Des contrats ont été proposés (notamment dans Leduc et al. (2021)) dans ce contexte et sont présentés dans la section 3.2.

Enfin, avant de conclure, des expériences ont été réalisées sur la blockchain Ethereum qui, jusque le 15 septembre 2022, fonctionnait grâce à la preuve de travail. Les consensus basés sur la preuve de travail ont la particularité de pouvoir abstraire leurs performances des capacités de calcul des éléments supportant la chaîne. Cette abstraction peut se réaliser grâce à la notion de difficulté qui sera étudiée en section 3.4.

3.1 Identifications des liens entre les éléments

La QoS (*Quality of Service*) peut se définir comme la capacité à véhiculer dans de bonnes conditions un type de trafic donné. Elle peut se quantifier de manière objective en mesurant et/ou

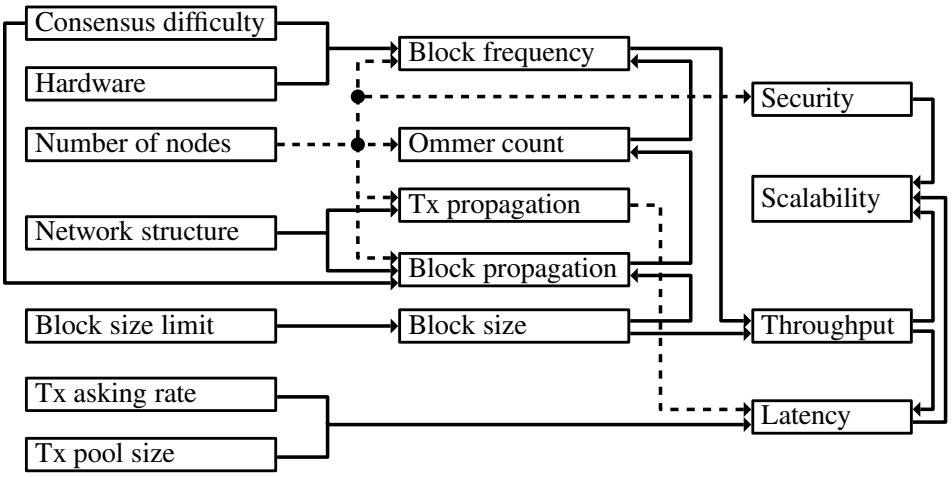


FIGURE 3.1 – Modèle d'interaction des performances en ce qui concerne les paramètres liés à la blockchain et à l'infrastructure (notez qu'il n'y a aucune différence entre les flèches pleines et les flèches en pointillés, elles ne sont utilisées que pour la clarté de la figure).

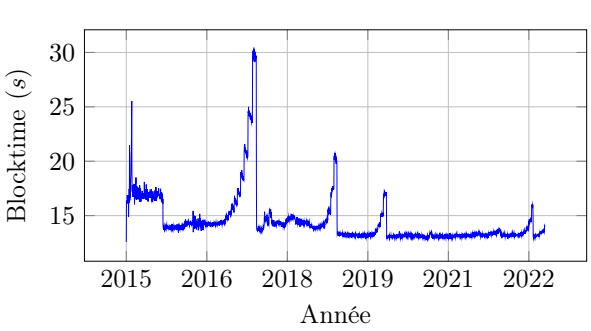
en calculant différents critères qui déterminent la qualité du réseau par rapport aux données transmises (retard, perte de paquets, bande passante, etc.) (Rifaï et al., 2011). La blockchain répond donc à cette définition en véhiculant des transactions et ses performances sont donc évaluées à travers ce prisme.

Comme nous l'avons vu dans la section 1.4, seules des interactions limitées entre les paramètres liés aux blockchains et à l'infrastructure sont formalisées dans la littérature, alors qu'il est essentiel d'avoir une compréhension globale de ces interactions pour répondre aux exigences de QoS. Il n'est évidemment pas si simple de développer un modèle/représentation unique de ces interactions, car il pourrait y avoir autant de modèles qu'il y a de technologies blockchain (par exemple, en raison des différents mécanismes de consensus). Dans cette section, nous tentons de clarifier, de manière graphique dans la figure 3.1, les interactions des paramètres des technologies blockchain basées sur PoW, et plus particulièrement dans le cas de la blockchain Ethereum (Wood et al., 2022). Les paramètres énumérés sur le côté gauche de la figure correspondent à des paramètres spécifiques à l'application (par exemple, l'architecture de réseau mise en œuvre, le nombre de nœuds/utilisateurs), tandis que les paramètres au centre font référence à des caractéristiques spécifiques/intrinsèques à la technologie blockchain (c'est-à-dire des paramètres non configurables) ; les paramètres sur le côté droit font référence à des mesures de performance QoS. Une flèche allant d'un cadre A à un cadre B indique que le paramètre A a une influence, à un degré plus ou moins grand, sur le paramètre B (ou la métrique de performance B). Les paragraphes suivants traitent des interactions identifiées.

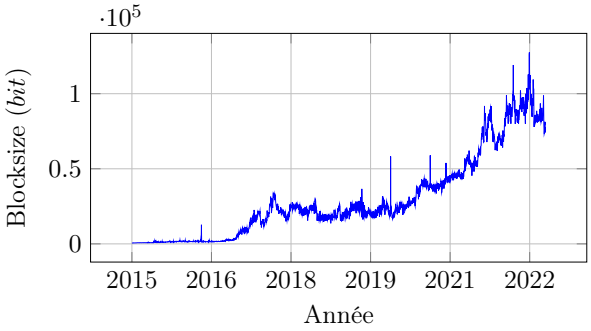
Tout d'abord, la difficulté du consensus réside dans la complexité de la génération d'un bloc dans la chaîne. Il est connu dans la littérature que le temps nécessaire pour résoudre ce défi est lié à la puissance de calcul du réseau, qui est composé du *"nombre de nœuds de calcul"* et des ressources *"hardware"* associé (par exemple, les *threads* alloués, la mémoire, les processeurs) (Pierro & Rocha, 2019). Ces trois paramètres (difficulté de consensus, matériel, nombre de nœuds) influencent inévitablement le paramètre "Fréquence (de génération) de bloc", comme le souligne la figure 3.1.

Dans la plupart des protocoles de consensus de type PoW, un contrôle asservi est généralement chargé de maintenir le délai de minage dans un intervalle de temps défini en adaptant la difficulté

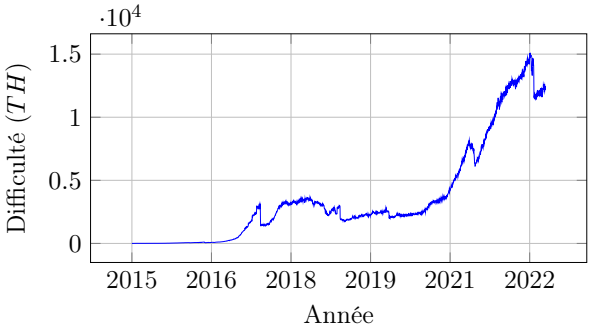
du consensus au fil du temps pour prévenir de l'augmentation de la capacité de calcul globale du réseau. Ainsi, l'exemple de la chaîne principale d'Ethereum nous montre que la difficulté du minage représentée en figure 3.2c suit le même profil que la capacité de calcul, et donc le nombre de hachés réalisés à la seconde, du réseau comme représenté figure 3.2d. Par conséquent, la "fréquence (de génération) des blocs" tend vers une limite (même si de fortes variations peuvent toujours se produire). L'exemple d'Ethereum sur la figure 3.2a nous montre que cette limite tend vers les 10 secondes, bien que des variations existent, notamment dus à des éléments associés au consensus.



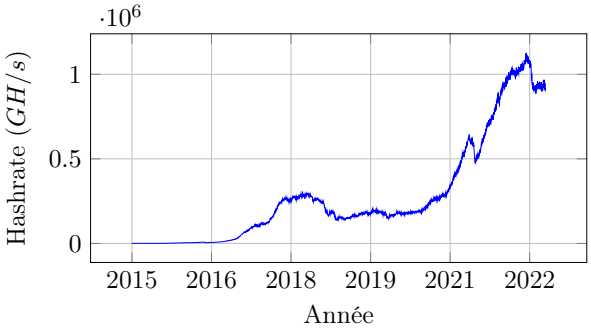
(a) Durée moyenne pour miner un bloc



(b) Taille moyenne des blocs



(c) Difficulté moyenne



(d) Taux de hachage moyen

FIGURE 3.2 – Valeurs principales sur la chaîne principale d'Ethereum, d'après Etherscan.io (Etherscan.io, 2022)

La structure du réseau, qui comprend les mécanismes de distribution des données, regroupe les paramètres qui influencent le délai de diffusion des transactions et des blocs dans la chaîne. Des délais élevés de propagation des blocs, associés à une fréquence élevée de génération des blocs, augmentent la probabilité de formation de blocs concurrents dans les nœuds du réseau blockchain. De tels blocs concurrents, qui sont appelés blocs "oncle" (ou *omer*) dans Ethereum, pourraient éventuellement ne pas être inclus dans la chaîne principale.

Le nombre de nœuds dans le réseau de la blockchain a une influence sur les performances globales du système, car il influe sur le processus de propagation des blocs et des transactions. En effet, plus le nombre de nœuds est élevé, plus le nombre de messages à propager sur le réseau est important. Le temps de propagation des blocs doit donc être réduit au maximum grâce à une dissémination intelligente des blocs, en se basant notamment sur des réseaux structurés comme Ethereum le réalise avec Kademlia (Maymounkov & Mazières, 2002).

Une autre interaction importante à prendre en compte est celle entre le "nombre de nœuds" et la "sécurité", car plus le nombre de nœuds est important, plus le niveau de sécurité est élevé.

Cette interaction s'applique aux technologies de blockchain basées sur des votes, mais aussi aux technologies utilisant un consensus basé sur une preuve. Dans ce dernier cas, ce n'est pas l'ajout d'un nœud en temps que membre qui vient améliorer la sécurité, mais la ressource sur laquelle se base la preuve que le nœud apporte qui vient améliorer la sécurité.

Le paramètre *Taille du bloc*, qui est limité par la "Limite de taille du bloc" définie lors de la configuration (par exemple, la limite de gaz dans Ethereum), a une influence directe sur le paramètre *"Propagation du bloc"*, ainsi que sur les performances du "débit" du système. À l'instar de la difficulté, cette valeur peut varier au cours du temps, comme dans Ethereum comme le montre la variation de la taille de bloc représenté sur la figure 3.2b. Cependant, dans cet exemple, cette variation est issue d'une décision commune des mineurs, sans réel objectif. Comme le souligne la figure 3.1, le débit est étroitement lié aux paramètres de fréquence de génération des blocs et de taille des blocs, car le produit des deux donne la bande passante de la mémoire où les transactions sont écrites.

La latence dans les réseaux blockchain dépend directement du paramètre de débit, bien qu'elle puisse être influencée négativement par des délais élevés de propagation des transactions. En outre, si le débit est inférieur aux taux de demande de transactions, des effets de congestion se produisent, car il y a une plus forte demande d'ajout de transaction que la chaîne n'est capable d'en ajouter, entraînant une augmentation de la latence.

La latence L_{BC} d'une transaction peut donc être divisée en la somme de trois délais : le délai L_q passé dans la file d'attente (qui diverge en cas de congestion), le délai L_{bg} de génération du bloc grâce au consensus et le délai L_{bp} de propagation du bloc à travers le réseau. En considérant qu'une transaction peut être effacée en cas de *fork* avec une probabilité p_{fork}, la latence peut se définir comme (Wilhelmi et al., 2022) :

$$L_{BC} = \frac{L_q + L_{bg} + L_{bp}}{1 - p_{fork}} \tag{3.1}$$

Compte tenu de la discussion ci-dessus, nous affirmons dans ce document qu'il est fortement important d'évaluer à quel niveau de performance un système donné basé sur la blockchain peut atteindre/supporter à long terme. Dans cette étude, nous sommes particulièrement intéressés par l'identification du débit maximal réalisable lorsque la blockchain est dans un état stable et non saturé (ce qui contribuerait inévitablement à une augmentation de la latence dans de tels cas). Cette limite de débit est appelée *capacité (à long terme)* et est introduite dans un contexte donné, ici le *Smart Farming.*

3.2 Contexte IoT et *Smart Farming*

Des études expérimentales ont été réalisées dans le cadre d'un article paru dans *Journal of Cleaner Production* (Leduc et al., 2021) concernant le *Smart Farming.* Au vu des différents avantages que fournit la blockchain, cette technologie est prometteuse pour permettre la rencontre des différents acteurs de l'agriculture sans faire intervenir un tiers de confiance.

Ainsi, pour illustrer une méthodologie de mesure des performances, l'application *FarMarket* est proposée. Il s'agit d'une application permettant un échange sécurisé de biens entre différents acteurs du monde agricole. Ces échanges reposent donc sur la blockchain, évitant ainsi la formation de tiers de confiances potentiellement malveillants.

3.2.1 État des lieux des initiatives de *Smart Farming*

Un certain nombre de solutions et de plateformes agricoles basées sur la blockchain sont en émergence (Juma et al., 2019), depuis que des startups telles que Skuchain [1], Provenance [2], AgriDigital [3] (X. Xu et al., 2019) et Farm Share [4] à de plus grandes entreprises comme Cargill Risk Management (Dujak & Sajter, 2019).

Même si la blockchain est utilisée à des fins différentes, comme la minimisation des prix, l'origine des produits et la réduction de l'influence des multinationales agricoles en faveur d'économies plus localisées (Galvez et al., 2018 ; Hang et al., 2020 ; Thomason et al., 2018), son objectif premier reste d'améliorer la transparence et la traçabilité tout au long de la chaîne alimentaire (Feng et al., 2020 ; G. Zhao et al., 2019). Un état des lieux plus complet des solutions utilisant la blockchain dans le cadre du *Smart Farming* est disponible en annexe A.2.

De manière générale, la littérature actuelle concernant le *Smart Farming* ne s'attache pas suffisamment à analyser correctement la mesure dans laquelle un choix de conception architecturale donné peut influencer les limites de capacité à long terme (en transactions/seconde Tx/s) du système proposé, et donc par définition, sur la QoS globale (de bout en bout). Également, elle n'adopte pas la rigueur décrite par les outils de mesures de performances décrits dans la section 2.1. De plus, les interactions entre les paramètres liés à la blockchain et à l'infrastructure sont rarement analysées et prises en compte dans la littérature, ce qui nécessite des recherches supplémentaires.

3.2.2 Vue d'ensemble de FarMarket

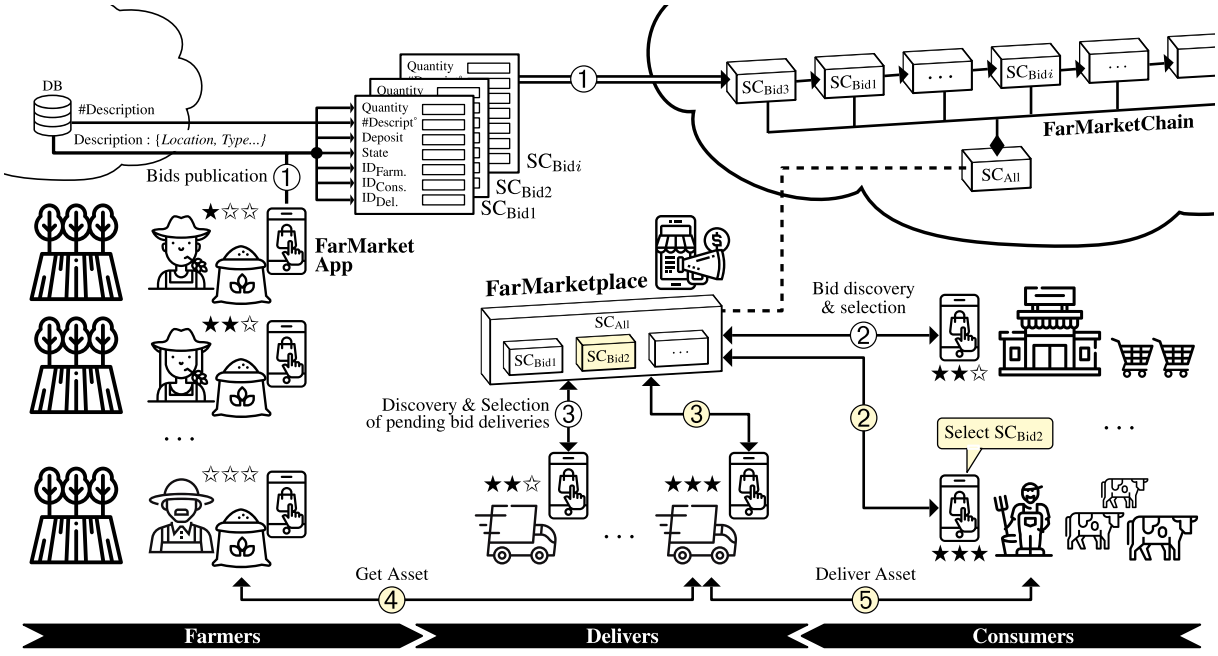


FIGURE 3.3 – Vue d'ensemble de l'écosystème FarMarket et des interactions associées entre les parties prenantes et le marché de soutien.

1. http://www.skuchain.com/, dernier accès mai 2020.
2. https://www.provenance. org/, dernier accès en mai 2020.
3. https://www.agridigital.io/, dernier accès en mai 2020.
4. http://farmshare.org, dernier accès en mai 2020.

Le marché numérique *FarMarletplace* proposé dans cette étude fait partie d'un écosystème plus large appelé *FarMarket*. Les éléments constitutifs de cet écosystème sont présentés dans la section 3.2.2 et les modèles de contrats intelligents spécifiés pour les échanges sont détaillés dans la section 3.2.2.

Descriptions des services de l'écosystème et de l'architecture support

Une vue d'ensemble des différentes parties prenantes et des composants logiciels/matériels soutenant l'écosystème *FarMarket* est représentée dans la figure 3.3. Cet écosystème est conçu pour :

- collecter les offres/contrats agricoles publiées par les agriculteurs ou d'autres fournisseurs partageant les mêmes idées (voir ① dans la figure 3.3) ;
- notifier aux consommateurs que de nouvelles offres/contrats sont disponibles, leur permettant de sélectionner/acheter un ou plusieurs contrats/offres (voir ②) ;
- notifier aux livreurs que de nouvelles offres de livraison sont disponibles, leur permettant de sélectionner une ou plusieurs offres (voir ③), ce qui – *s'il est accepté par l'agriculteur et le consommateur* – implique que le livreur doit collecter et livrer le bien associé à l'offre/contrat (voir ④ et ⑤), après quoi il sera payé ;
- permettent à toutes les parties prenantes d'évaluer la qualité du service, à savoir (i) le consommateur peut évaluer la qualité du service de livraison (par exemple, la ponctualité, le professionnalisme) et les biens agricoles reçus (ii) le livreur peut évaluer la qualité de l'agriculteur et du consommateur (par exemple, la ponctualité, l'exactitude de l'emplacement spécifié) ; et (iii) l'agriculteur peut évaluer la qualité du service de livraison.

Pour réaliser les fonctionnalités ci-dessus, trois blocs de construction principaux ont été conçus et intégrés dans l'écosystème FarMarket.

1. *FarMarketchain* : Il s'agit de la blockchain et des contrats intelligents associés. Une base de données, désignée par DB dans la figure 3.3, fonctionne avec la blockchain pour éviter de stocker de longues chaînes de caractères dans la blockchain elle-même, ce qui est coûteux (seul le hachage de la chaîne correspondante est ajouté à la blockchain).
2. *FarMarketplace* : Il s'agit de la plateforme de marché numérique. Elle héberge la blockchain et la base de données, et joue le rôle d'intermédiaire entre les différents acteurs de l'écosystème.
3. *FarMarketApp* : Il s'agit de la DApp qui permet aux parties prenantes de bénéficier de l'ensemble des services proposés par l'écosystème FarMarket.

L'utilisation d'une base de données extérieure pour stocker les données est une pratique courante dans le développement de blockchain. En effet, il peut devenir coûteux de stocker des informations brutes dans un registre distribué, car chaque transaction implique généralement des frais, et le stockage uniquement du haché des informations stockées dans une base de données (permettant de vérifier l'intégrité des données en comparant ce haché à tout moment) est une alternative largement adoptée. Cette base de données est, dans notre cas, un serveur, mais peut être remplacée par un cloud privé (Sumathi & Sangeetha, 2020) ou IPFS (InterPlanetary File System) pour permettre un cadre pair-à-pair entièrement décentralisé (N. Singh & Vardhan, 2019 ; Steichen et al., 2018). Comme le contrôle d'accès aux données n'est pas l'objectif principal de ce document, toutes les données sont librement accessibles, bien que des stratégies de contrôle d'accès plus avancées puissent être adoptées à l'avenir, comme la définition d'une politique XACML (Ramli et al., 2014) ou l'adoption d'une solution basée sur la blockchain (Di Francesco Maesa et al., 2017 ; Esposito et al., 2021).

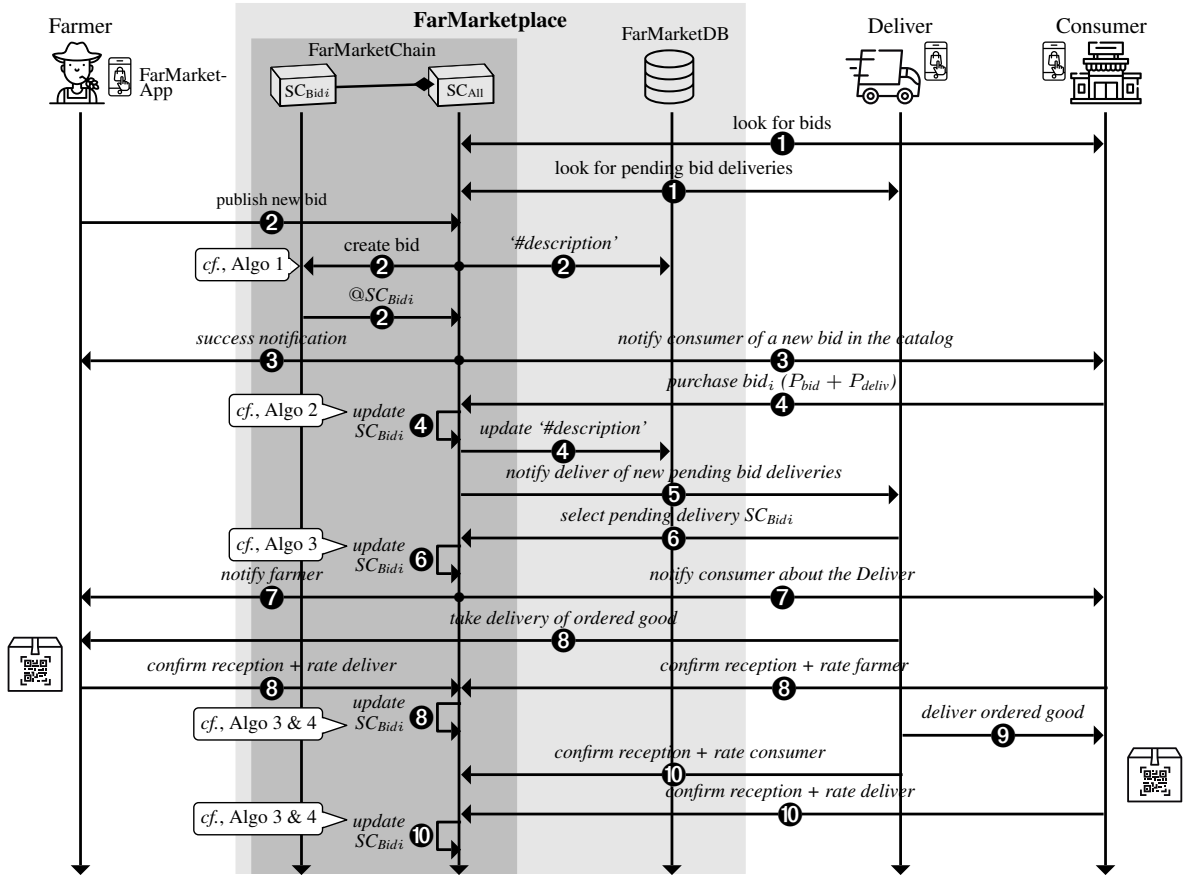


FIGURE 3.4 – Protocole de messagerie soutenant l'écosystème FarMarket

Smart Farming contracts

L'ensemble des interactions (ou communications) qui se produisent entre les éléments constitutifs et les parties prenantes présentés précédemment sont détaillées dans la figure 3.4 sous la forme d'un diagramme de séquence.

Tout d'abord, les consommateurs et les livreurs peuvent rechercher des offres et des propositions de livraison en attente (*cf.*, ❶ dans la figure 3.4), et éventuellement s'abonner à la plateforme FarMarketplace pour être avertis dès qu'une nouvelle offre est publiée (le protocole Message Queuing Telemetry Transport est utilisé à cet égard). Les agriculteurs peuvent publier de nouvelles offres en spécifiant – *via la FarMarketApp* – les informations relatives à leur offre (*cf.*, ❷). Cette action appelle la fonction de création d'offres détaillée dans l'algorithme 1, où $\mathrm{SC}_{\mathrm{Bid}i}$ fait référence à un *smart contract* donné (i faisant référence au i^{ime} contrat). En fait, deux paramètres d'entrée sont envoyés par la *FarMarketApp* à l'algorithme 1, à savoir, *(i) l'ID de l'agriculteur* et *(ii) Description* (composé de plusieurs éléments d'information comme détaillée dans le tableau 3.1). À partir de ces deux paramètres d'entrée, cinq attributs immuables – *également appelés "variables d'état"* – sont extraits/dérivés pour être stockés dans la *FarMarketchain*, à savoir : (i) l'ID du fermier (ii) la valeur *hash* de la description de l'offre (iii) le prix de l'offre (iv) le solde du contrat, et (v) l'état du contrat. Les paramètres d'entrée communiqués par *FarMarketApp* et les variables d'état dérivées sont résumés dans la table 3.1.

À ce stade, le contrat est disponible sur la place de marché et les consommateurs ont été

	Variable	Description
Entrées de l'application	ID_{User}	Identifiant de l'utilisateur se référant soit à un agriculteur, soit à un livreur, soit à un consommateur, respectivement désignés par $\mathrm{ID}_{Far}, \mathrm{ID}_{Del}, or \mathrm{ID}_{Con}$
	$\mathrm{ID}_{User}^{\mathrm{balance}}$	Solde du portefeuille de l'utilisateur
	Montant	Montant payé par un consommateur pour acheter un contrat/une offre donné, noté par $\mathrm{SC}_{\mathrm{Bid}i}$
	$\mathrm{P}_{\mathrm{Far}}$, $\mathrm{P}_{\mathrm{Del}}$	Prix (i) des produits agricoles à payer à l'agriculteur, calculés à l'aide de la fonction *priceBid()*, qui prend comme entrées le type et la quantité de produit et (ii) la prestation de service qui dépend, entre autres, de la distance entre le consommateur et l'agriculteur.
	Desc	Description stockée dans la base de données (voir DB dans les figures Fig :FarMarketEcosystem et 3.4), consistant en : (i) l'emplacement de l'agriculteur et du consommateur désigné par $\mathrm{loc}^{\mathrm{ID}_{Far}}$ et $\mathrm{loc}^{\mathrm{ID}_{Con}}$, respectivement ; (ii) le type ; (iii) la quantité de biens agricoles ; et (iv) des commentaires supplémentaires.
	$R_{User1 \to User2}$	Note d'évaluation indiquant le degré de satisfaction de *User1* concernant le 'service' fourni par (ou le comportement de) *User2*. Toutes les combinaisons possibles de notes d'évaluation sont contenues dans un ensemble noté par $\mathcal{R} = \{R_{\mathrm{Far}\to\mathrm{Del}}, R_{\mathrm{Del}\to\mathrm{Far}}, R_{\mathrm{Del}\to\mathrm{Con}}, R_{\mathrm{Con}\to\mathrm{Far}}, R_{\mathrm{Con}\to\mathrm{Del}}\}$
Variables d'états	$\mathrm{SC}_{\mathrm{Bid}i}^{state}$	Variable d'état faisant référence à l'état du contrat $\mathrm{SC}_{\mathrm{Bid}i}$ à un moment donné. Les états possibles sont les suivants : {*Available, WaitForDeliverer, WaitForDelivery, OnDelivery, Delivered*}
	$\mathrm{SC}_{\mathrm{Bid}i}^{\mathrm{ID}_{User}}$	Variable d'état se référant à une partie prenante donnée (*cf.*, ID_{User})
	$\mathrm{SC}_{\mathrm{Bid}i}^{\#\mathrm{Desc}}$	Variable d'état faisant référence au haché de la description de l'enchère (*cf.*, "Desc"), obtenue à l'aide de la fonction *priceBid(...)*.
	$\mathrm{SC}_{\mathrm{Bid}i}^{\mathrm{balance}}$	Variable d'état se référant au solde du contrat intelligent/de l'offre i
	$\mathrm{SC}_{\mathrm{Bid}i}^{\mathrm{P}_{\mathrm{Far}}}$, $\mathrm{SC}_{\mathrm{Bid}i}^{\mathrm{P}_{\mathrm{Del}}}$	Variables d'État se référant aux prix à payer à l'agriculteur et au livreur (*cf.*, $\mathrm{P}_{\mathrm{Far}}$, $\mathrm{P}_{\mathrm{Del}}$)
	$\mathrm{SC}_{\mathrm{Bid}i}^{R_{User1\to User2}}$	Variable d'état se référant aux notes de satisfaction précédemment décrites (*cf.*, $\mathcal{R}$)

TABLEAU 3.1 – Variables d'état et d'entrée relatives à l'application FarMarket

informés de son existence (❸). Lorsqu'un consommateur sélectionne une offre à acheter (*cf.*, ❹), la fonction d'achat de $\mathrm{SC}_{\mathrm{Bid}i}$ est exécutée, comme détaillé dans l'algorithme 2. Le but de cette fonction/algorithme est de vérifier que le contrat est dans le bon état (devrait être *Available* pour l'achat) et que le consommateur a suffisamment d'argent dans son portefeuille numérique. La somme d'argent requise pour acheter le contrat doit être égale au prix de l'offre plus le prix de la prestation du service, qui est désigné par P_{Del}. On notera que P_{Del} est une entrée de la fonction d'achat, ce qui signifie qu'elle est calculée en dehors du contrat intelligent (la fonction de calcul du prix de la prestation de service dépasse le cadre de cet article ; cependant, elle peut être calculée sur la base de paramètres tels que l'emplacement du consommateur et le type de biens à livrer) et ensuite ajoutée à la variable d'état correspondante (cf. ligne 3 de l'algorithme 2). Si ces conditions sont satisfaites, les variables d'état suivantes sont mises à jour : (i) abonné/bénéficiaire du contrat avec l'identifiant du consommateur ; (ii) prix de livraison du contrat ; (iii) localisation du consommateur, qui fait partie de la description de l'offre ; (iv) solde du contrat crédité du montant requis ; et (v) état du contrat réglé sur *WaitForDeliverer*. De plus, le solde du portefeuille du consommateur est mis à jour en conséquence (cf. ligne 7 dans l'algorithme 2). Une fois que l'état du contrat a été mis à jour en *WaitForDeliverer*, les livreurs qui ont souscrit à des offres de livraison d'offres en attente en sont informés (cf. ❺).

Lorsqu'un livreur sélectionne une offre de livraison en attente (cf. ❻), la fonction de livraison de $\mathrm{SC}_{\mathrm{Bid}i}$ est exécutée, comme détaillée dans l'algorithme 3. Cette fonction/algorithme vérifie

Algorithme 1 : create_SC_{Bid_i}

Input : ID_{User}, Desc
1 $SC^{ID_{Far}}_{Bid_i} \leftarrow ID_{User}$; // Initialise l'ID du propriétaire du contrat
2 $SC^{\#Desc}_{Bid_i} \leftarrow hash(Desc)$; // Calcule la valeur #Desc
3 $SC^{P_{Far}}_{Bid_i} \leftarrow priceBid(Desc_{quant}, Desc_{type}...)$; // Calcule le prix de l'offre
4 $SC^{balance}_{Bid_i} \leftarrow 0$; // Initialise le solde de l'offre
5 $SC^{state}_{Bid_i} \leftarrow$ Available; // Initialise l'état du contrat
Output : SC_{Bid_i}

Algorithme 2 : purchase_SC_{Bid_i}

Input : $loc^{ID_{Con}}$, ID_{User}, Desc, amount, P_{Del}
1 **if** $SC^{state}_{Bidi} == Available\&\ amount == \left(SC^{P_{Far}}_{Bidi} + P_{Del}\right)$ **then**
2 $SC^{ID_{Con}}_{Bid_i} \leftarrow ID_{User}$; // Mise à jour de la localisation du consommateur
3 $SC^{P_{Del}}_{Bid_i} \leftarrow P_{Del}$; // Fixe le prix de la prestation de services
4 $Desc \leftarrow Desc \cup \{loc^{ID_{Con}}\}$; // Mise à jour de la description de la BD
5 $SC^{\#Desc}_{Bid_i} \leftarrow hash(Desc)$; // Calcule la nouvelle valeur #Desc
6 $SC^{balance}_{Bid_i} \leftarrow$ amount; // Dépôt
7 $ID^{balance}_{Con} \leftarrow ID^{balance}_{Con} -$ amount; // Mise à jour du solde
8 $SC^{state}_{Bid_i} \leftarrow$ WaitForDeliverer; // Mise à jour de l'état du SC
Output : SC_{Bid_i}

l'état du contrat. Si l'état est *WaitForDeliverer*, alors le livreur devient le fournisseur officiel du service de livraison (*cf.*, ligne 2 dans l'algorithme 3), l'état du contrat devient *WaitForDelivery*, et le consommateur et le fermier sont tous deux informés de l'identité du livreur (*cf.*, ❼). À ce stade, le livreur doit prendre livraison des marchandises commandées (*cf.*, ❽). Au moment de l'échange des marchandises, le livreur et le fermier doivent tous deux confirmer la réussite de la réception, ce qui, dans la pratique, entraîne l'appel de la fonction de livraison *delivery*_SC_{Bid_i}, conduisant à un changement de l'état du contrat en *OnDelivering* (*cf.*, lignes 4-5 dans l'algorithme 3). Dans l'étape finale, le livreur livre les biens au consommateur (*cf.*, ❾). Tous deux confirment la réussite de la livraison/réception (*cf.*, ❿), ce qui entraîne un changement de l'état du contrat en *Delivered* (*cf.*, lignes 6-7 dans l'algorithme 3), après quoi les paiements sont effectués à l'agriculteur et au livreur, et les soldes mis à jour en conséquence (*cf.*, lignes 8-10).

Algorithme 3 : delivery_SC_{Bid_i}

Input : ID_{User}
1 **if** $SC^{state}_{Bidi} == WaitForDeliverer$ **then**
2 $SC^{ID_{Del}}_{Bid_i} \leftarrow ID_{User}$; // Établie le contrat du livreur
3 $SC^{state}_{Bid_i} \leftarrow$ WaitForDelivery ; // Mise à jour de l'état du SC
4 **else if** $SC^{state}_{Bidi} == WaitForDelivery\ \&\ SC^{ID_{Del}}_{Bidi} == ID_{\text{User}}$ **then**
5 $SC^{state}_{Bid_i} \leftarrow$ OnDelivering ; // Mise à jour de l'état du SC
6 **else if** $SC^{state}_{Bidi} == OnDelivering\&\ SC^{ID_{Con}}_{Bidi} == ID_{\text{User}}$ **then**
7 $SC^{state}_{Bid_i} \leftarrow$ Delivered; // Mise à jour de l'état du SC
8 $ID^{balance}_{Farm} \leftarrow ID^{balance}_{Farm} + SC^{P_{bid}}_{Bid_i}$; // Mise à jour du solde de l'agriculteur
9 $ID^{balance}_{Del} \leftarrow ID^{balance}_{Del} + SC^{P_{Del}}_{Bid_i}$; // Mise à jour du solde du livreur
10 $SC^{balance}_{Bid_i} \leftarrow 0$; // Remise à zéro du solde du SC
Output : SC_{Bid_i}

Afin de donner aux parties prenantes la possibilité d'évaluer la qualité du service, comme nous l'avons vu précédemment dans la section 3.2.2, une autre fonction est définie dans le contrat intelligent pour rendre les scores de satisfaction immuables. Cette fonction est détaillée dans l'al-

gorithme 4, permettant aux consommateurs, agriculteurs et livreurs de s'évaluer mutuellement par le biais d'un score d'évaluation désigné par $R_{User1 \to User2}$ (cf. table 3.1). Ces scores d'évaluation font référence au niveau de réputation/satisfaction liée à une partie prenante donnée de FarMarketplace. Notons que les fonctions utilisées pour calculer ces scores d'évaluation ne sont pas incluses dans le cadre de ce document.

Algorithme 4 : rating_$\mathrm{SC_{Bid}}i$

Input : $\mathrm{ID_{User}}$, $\mathcal{R}$; // Emplacement du consommateur

1 **if** $SC_{Bidi}^{state} == Delivered$ & $ID_{User} == SC_{Bidi}^{ID_{Far}}$ **then**
2 | $\mathrm{SC}_{\mathrm{Bid}i}^{R_{\mathrm{Far}\to\mathrm{Del}}} \leftarrow \mathcal{R}_{\mathrm{Far}\to\mathrm{Del}}$; // Définie la note d'évaluation
3 **else if** $SC_{Bidi}^{state} == Delivered$ & $ID_{User} == SC_{Bidi}^{ID_{Del}}$ **then**
4 | $\mathrm{SC}_{\mathrm{Bid}i}^{R_{\mathrm{Del}\to\mathrm{Far}}} \leftarrow \mathcal{R}_{\mathrm{Del}\to\mathrm{Far}}$; // Définie la note d'évaluation
5 | $\mathrm{SC}_{\mathrm{Bid}i}^{R_{\mathrm{Del}\to\mathrm{Con}}} \leftarrow \mathcal{R}_{\mathrm{Del}\to\mathrm{Con}}$; // Définie la note d'évaluation
6 **else if** $SC_{Bidi}^{state} == Delivered$ & $ID_{User} == SC_{Bidi}^{ID_{Con}}$ **then**
7 | $\mathrm{SC}_{\mathrm{Bid}i}^{R_{\mathrm{Con}\to\mathrm{Far}}} \leftarrow \mathcal{R}_{\mathrm{Con}\to\mathrm{Far}}$; // Définie la note d'évaluation
8 | $\mathrm{SC}_{\mathrm{Bid}i}^{R_{\mathrm{Con}\to\mathrm{Del}}} \leftarrow \mathcal{R}_{\mathrm{Con}\to\mathrm{Del}}$; // Définie la note d'évaluation

Output : $\mathrm{SC_{Bid}}i$

3.3 Mesure de la *capacité* et résultats

3.3.1 Définition d'un plan d'expérience

Dans Ethereum, différentes boucles de régulation sont mises en œuvre pour équilibrer la sécurité/robustesse (liée au coût de calcul) de la blockchain et la QoS – *principalement en termes de débit et de latence* – offerte pour soutenir les contrats intelligents. En effet, la puissance de hachage influence directement le temps de résolution d'un bloc, c'est-à-dire le délai de minage d'un bloc, qui, par définition, influence la latence. À cet égard, dans Ethereum, la difficulté de minage des blocs (une estimation statistique du nombre de hachages qui doivent être générés pour trouver une solution valide) est réorientée dans le temps pour contrôler ce délai de minage afin de trouver un équilibre entre taux de hachage et difficulté. À cet égard, la chaîne principale trouve cet équilibre dans les figures 3.2c et 3.2d puisqu'elle a été lancée sur une période relativement longue.

La figure 3.5 met en évidence l'évolution à long terme de la difficulté dans notre configuration. Dans cette expérience, une chaîne a été initialisée avec une difficulté arbitraire bien inférieure à la difficulté correspondante à la puissance de hachage du réseau de nœuds, puis la difficulté a été mesuré sur une période de 12 heures. On peut observer que la difficulté est fréquemment reciblée ; cependant, elle tend à converger vers une asymptote. En fait, le système réagit en augmentant/diminuant la difficulté si les blocs précédents sont générés plus rapidement/plus lentement qu'un temps de bloc minier spécifié, qui varie de 9 à 17 secondes.

Par rapport à d'autres travaux de recherche de pointe, nos expériences d'évaluation des performances se concentrent sur les performances de la QoS à long terme, c'est-à-dire lorsque la QoS ne varie plus en raison du contrôle par régulation. Pour accélérer le contrôle et éviter les problèmes de temps de réponse, la difficulté initiale du bloc de genèse (*genesis block*, le premier bloc contenant la configuration de la chaîne) de la blockchain est fixée directement à sa valeur à long terme à l'état d'équilibre. Cette valeur correspond à $10 \times \#_{total}$, où $\#_{total}$ correspond à la somme de chaque hashrate (le nombre de hachés réalisés par un nœud chaque seconde) des ordinateurs du réseau. Par rapport à d'autres études, cela nous permet également de nous

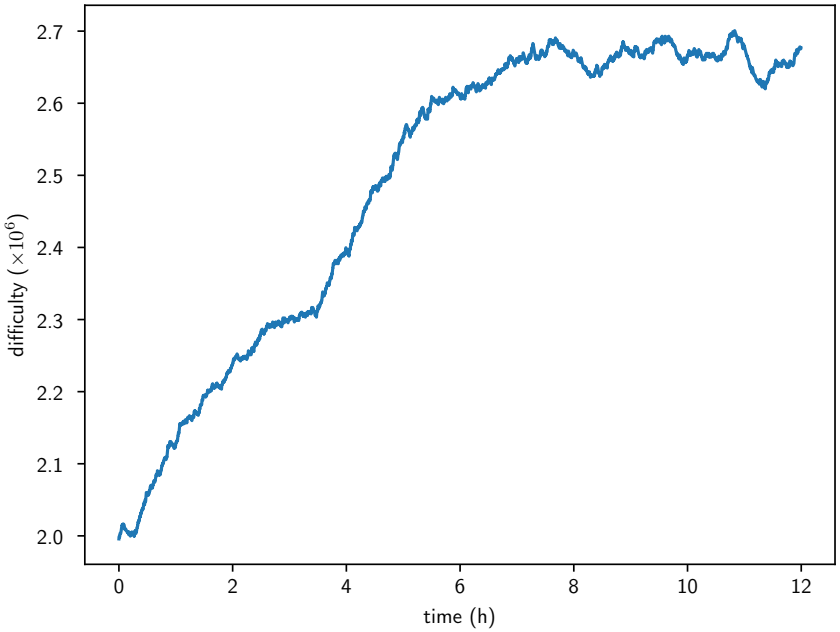


FIGURE 3.5 – Difficulté (de minage) par horodatage des blocs

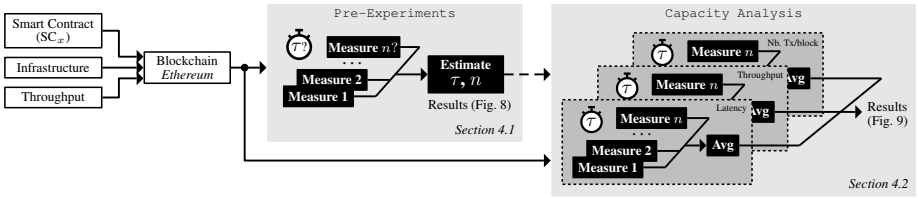
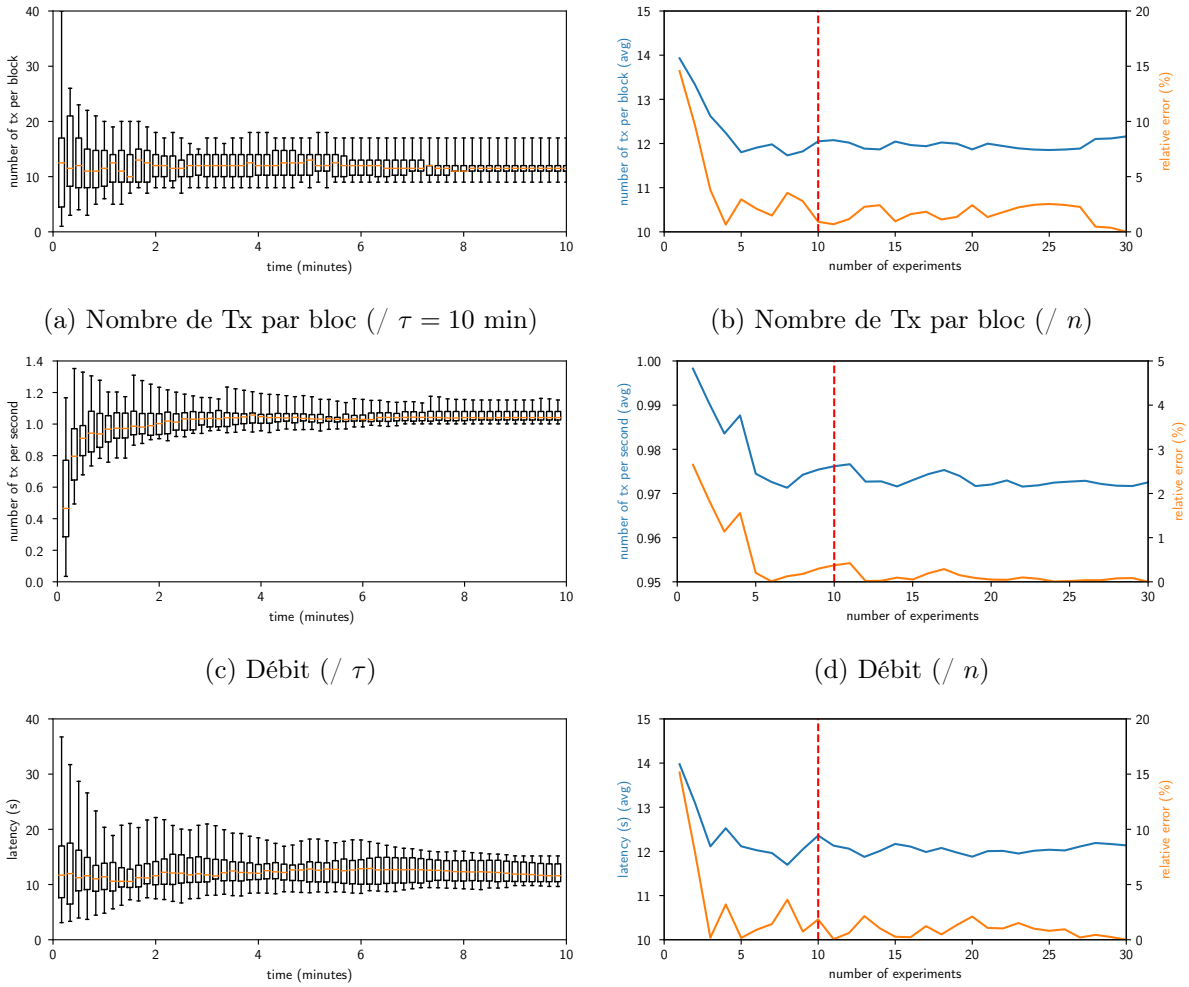


FIGURE 3.6 – Méthodologie de détermination des plans d'expériences

concentrer sur la capacité réelle de la chaîne et d'atténuer la différence de ressources matérielles (par exemple, le nombre de *threads*, la mémoire autorisée, les processeurs).

Cette étude est réalisée avec l'implémentation du contrat intelligent proposé et compilé grâce au compilateur Solidity (v0.5.1) sur une chaîne partagée avec trois ordinateurs fonctionnant avec Geth 1.9.6 sur Ubuntu 18. Les trois nœuds sont définis comme des mineurs, l'un étant responsable de la génération des transactions. La latence et le débit étant tous deux influencés par les délais de propagation des blocs (*cf.*, figure 3.1), les nœuds sont connectés via un commutateur offrant une large bande passante afin de maintenir les délais (qui ne sont pas contrôlables) au minimum, et ainsi permettre aux expériences d'être reproductibles. Pour compléter le bloc de genèse, la limite de gaz, qui influence la taille du bloc, est fixée arbitrairement à 16777216 gaz. Le gaz (*gas*) est une unité de mesure en Ethereum du coût d'exécution de chacune des opérations élémentaire. Ainsi, donner une limite au gaz dans un bloc limite donc le nombre d'exécutions élémentaire et crée de facto une taille maximale du bloc. Par conséquent, pour s'assurer que la plateforme expérimentale présenterait le comportement attendu en termes de difficulté ($\#_{total}$) et d'influence du nombre de *threads*, une analyse expérimentale préliminaire a été réalisée, et qui est illustré dans la section 3.4.3.

Sur la base de cette configuration, des préexpériences ont été réalisées afin de déterminer le nombre (n) et la durée (τ) des expériences à reproduire dans la deuxième étape d'évaluation. Comme mentionné précédemment, la latence et le débit offerts par la chaîne centrale ont été considérés comme les mesures de performance (les deux étant obtenus en comparant les horodatages lorsque les transactions sont générées pour un contrat et soumises à la chaîne). La figure 3.7 affiche l'évolution de ces deux métriques, en plus du nombre de transactions par bloc. On peut noter que les expériences ont été réalisées après une période de génération de blocs de 30 minutes afin de garantir la stabilité de la difficulté.



(a) Nombre de Tx par bloc (/ $\tau = 10$ min)

(b) Nombre de Tx par bloc (/ n)

(c) Débit (/ τ)

(d) Débit (/ n)

(e) Latence (/ τ)

(f) Latence (/ n)

FIGURE 3.7 – Évolution des métriques QoS pour différents nombres d'expériences (n) et dans le temps (τ)

La figure 3.7a donne un aperçu de l'évolution du nombre de transactions par bloc pendant dix minutes, pour dix expériences, une transaction étant soumise par seconde. On peut observer que la convergence est relativement rapide et que la blockchain reste (raisonnablement) stable. Cela est également confirmé par l'observation des mesures de QoS, à savoir la latence (voir la figure 3.7e) et le débit (voir figure 3.7c). Le débit est calculé comme le nombre de transactions pour un bloc divisé par le délai de minage de ce bloc ; la latence est basée sur la différence

de temps entre l'émission et la validation de la transaction. Après dix minutes, la latence se situe dans la fourchette attendue (avec un écart-type inférieur à une seconde et un délai moyen de minage de $11,6$ seconde par bloc). Ces expériences nous ont permis de choisir une durée de simulation de $\tau = 10$ min pour la deuxième étape d'évaluation expérimentale (un nombre suffisant d'échantillons étant disponible, 52 blocs en moyenne). Dans une étape ultérieure, la confiance a été analysée en considérant un plus grand nombre d'expériences.

La figure 3.7b montre l'évolution du nombre moyen de transactions par bloc lorsque les expériences ont été ajoutées. L'erreur relative correspond à la différence entre la moyenne pour un nombre donné et la moyenne pour 30 expériences. Les figures 3.7d et 3.7f donnent un aperçu de la même analyse pour les métriques de débit et de latence, respectivement. On peut observer que l'état stable est atteint avec dix expériences et que l'augmentation du nombre d'expériences ne modifie pas significativement la précision. Par conséquent, dans ce qui suit, chaque configuration est répétée $n = 10$ fois.

Comme notre objectif principal est d'évaluer l'ensemble de l'écosystème, la section suivante vise à définir le service maximal qui peut être offert pour soutenir l'émission de contrats intelligents. Les transactions et les blocs sont les fonctions d'évaluation qui sont utiles aux exploitants pour sélectionner les acteurs les plus appropriés. Il est donc important d'identifier la capacité offerte par la chaîne pour stocker les informations liées aux contrats. À cet égard, les expériences sont répétées non seulement en fonction des paramètres identifiés dans cette section, mais aussi en augmentant le taux d'émission des transactions.

3.3.2 Analyse de la capacité de la FarMarketChain

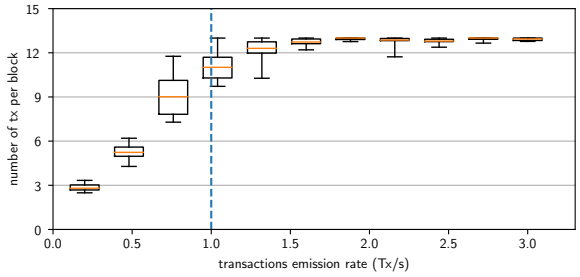
La figure 3.8 montre l'évolution des trois métriques identifiées précédemment, mais cette fois pour un taux d'émission de transactions donné allant de 1 $Tx/s(commeprcdemment)21$ Tx/s. Chaque point consiste en des expériences de dix minutes, répétées dix fois. Comme la `gasLimit` a été choisie arbitrairement (1 `blockSize` = 16 777 216 gaz), nous avons répété le même ensemble d'expériences sur une autre chaîne avec une `gasLimit` dix fois plus grande. Les figures 3.8d, 3.8b, et 3.8f correspondent à cette deuxième expérience. Nous supposons que, comme la taille des blocs sera 10 fois plus grande, la limite sur le débit validé sera 10 fois plus grande.

Nombre de transactions par bloc

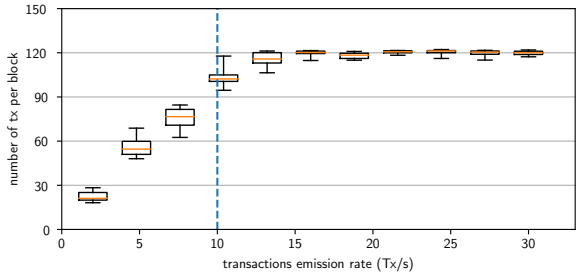
Les figures 3.8a et 3.8b donnent un aperçu de l'évolution du nombre de transactions à l'intérieur d'un bloc en fonction du taux d'émission des transactions. Deux comportements émergent avant et après la valeur du débit d'émission que nous désignons par *capacité*. Avant que la *capacité* ne soit atteinte, l'évolution du nombre de transactions par bloc est linéaire et correspond au délai moyen de minage d'un bloc (il varie, mais est expérimentalement proche de 11 secondes) multiplié par le taux d'émission. Une fois que la *capacité* est dépassée, le délai nécessaire pour remplir un bloc est inférieur au délai de minage d'un bloc, ce qui conduit au remplissage des blocs avec le nombre maximum de transactions. Par conséquent, le nombre de transactions par bloc est approximativement constant, avec une variation minimale entre les deux expériences.

Latence

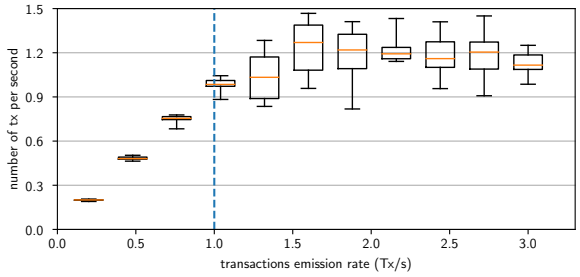
Les figures 3.8e et 3.8f affichent la latence moyenne des transactions en fonction du débit. Nous pouvons à nouveau extraire deux comportements : avant et après la *capacité*. Avec un taux d'émission inférieur à la *capacité*, le bloc n'étant pas rempli, la transaction est validée et insérée dans le bloc suivant lorsqu'elle atteint un nœud. La latence est donc égale au temps d'attente



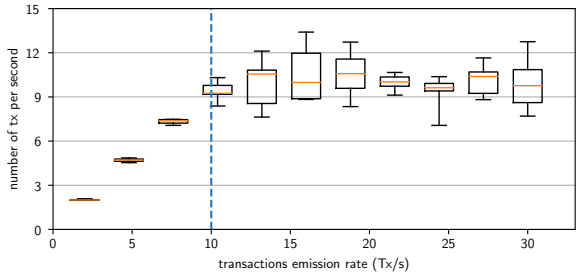
(a) Nombre de Tx par Bloc (`gasLimit` × 1)



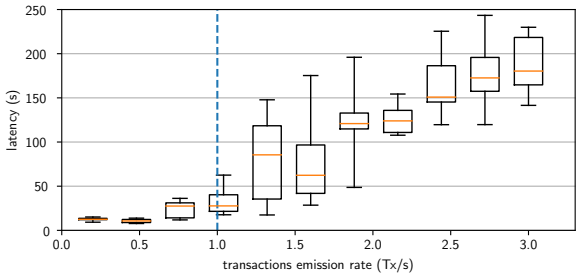
(b) Nombre de Tx par Bloc (`gasLimit`×10)



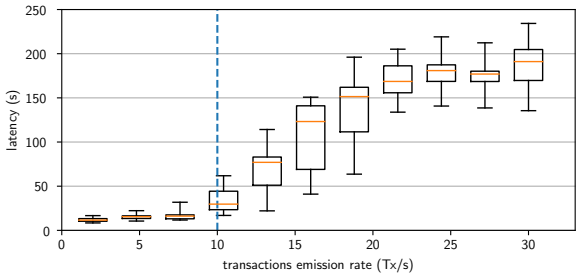
(c) Débit (`gasLimit`×1)



(d) Débit (`gasLimit`×10)



(e) Latence (`gasLimit`×1)]



(f) Latence (`gasLimit`×10)

FIGURE 3.8 – Analyse expérimentale de la *capacité* de la blockchain

du bloc suivant, qui correspond au délai de minage. Ce délai diffère très peu entre les deux expériences (en raison du contrôle par régulation de la difficulté) ; par conséquent, la latence ne diffère que marginalement. Après la *capacité*, le système est surchargé. Comme le débit de validation est inférieur au taux d'émission, la transaction est mise en file d'attente, ce qui entraîne une augmentation significative de la latence.

Débit

Les figures 3.8c et 3.8d donnent un aperçu de l'évolution du débit de validation. Dans la première partie, le débit de validation est égal au taux d'émission, car chaque transaction émise est validée (ce qui conduit à la fonction d'identité, avec des différences minimes entre les deux expériences). Dans la deuxième partie, les blocs sont saturés de telle sorte que le débit validé est limité à une constante, ce qui correspond à la *capacité* et est définie par le rapport entre le nombre de transactions par bloc et le délai de minage d'un bloc.

Comme supposé, le nombre de transactions contenues dans un bloc est proportionnel à la

taille du bloc. En augmentant la taille du bloc d'un facteur de 10, nous augmentons également le nombre de transactions dans un bloc d'un facteur de 10. Le débit maximal validé, c'est-à-dire la *capacité*, est également proportionnel à la taille du bloc.

3.3.3 Discussions

Comme le montre la revue de la littérature sur les solutions d'*e*-agriculture basées sur la blockchain présentée dans la l'annexe A.2, un grand nombre d'études de recherche n'ont pas fourni de résultats d'évaluation des performances de leurs solutions, et encore moins ont comparé leurs solutions avec d'autres approches de l'état de l'art. Dans les rares cas où une évaluation a lieu, la méthode de mesure des performances n'est pas décrite avec suffisamment de rigueur pour permettre la reproductibilité de la mesure. Un banc d'expérimentation des applications de Smart Farming, semblable à ceux présentés en section 2.1 et avec la même exactitude.

À cet égard, nous proposons de comparer notre proposition à un autre *smart contract* agricole, celui proposé par Tian (Tian, 2017). Ce contrat est plus léger à émettre (les frais de transaction sont de 894 159 de gaz au lieu de 1 148 305 de gaz comme avec les contrats proposés de FarMarket), ce qui devrait permettre de remplir des blocs avec un plus grand nombre de contrats.

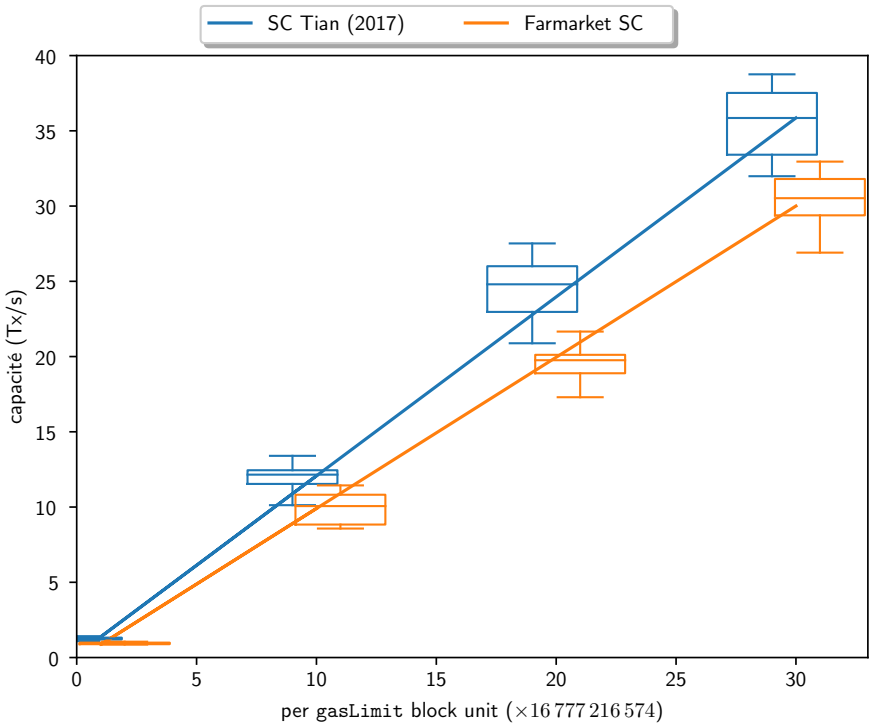


FIGURE 3.9 – Comparaison de l'évolution de la *capacité* pour deux contrats intelligents

Cette petite taille est confirmée expérimentalement dans la figure 3.9 en considérant les mesures de la *capacité* pour différentes tailles de blocs (définies en termes de `gasLimit`) et les régressions linéaires entre ces valeurs, affirmant la linéarité supposée dans la section précédente. On peut observer que le taux de pente est plus faible (de 19%) pour l'approche proposée par rapport au contrat intelligent de Tian (Tian, 2017), ce qui est en accord avec le fait que le contrat proposé est 28% plus lourd en coût de transaction du gaz. Cela soutient un lien entre le coût de transaction d'un contrat et la pente (c'est-à-dire le nombre de transactions par unité de taille

de bloc). Compte tenu de cela, une analyse plus approfondie pourrait fournir une prédiction de la taille de bloc nécessaire pour atteindre un débit donné, pour un contrat donné, de sorte que le taux de transaction d'émission reste inférieur à la *capacité* de la blockchain. Dans la mise en œuvre d'Ethereum, la taille des blocs peut être ajustée en modifiant le `gasLimit` des blocs. Deux méthodes sont disponibles : premièrement, le bloc de genèse peut être modifié lors de la création d'une chaîne. Deuxièmement, en modifiant le calcul de la limite de gaz du bloc suivant dans le code de la chaîne.

Cependant, il est important de se rappeler que l'augmentation de la taille du bloc peut augmenter les délais de propagation, et donc, une attention particulière doit être considérée pour s'assurer que la capacité du réseau est suffisante. De plus, comme la *capacité* dépend linéairement du débit de génération des blocs, il est également lié au délai d'extraction d'un bloc. En reconfigurant le contrôle de régulation de la difficulté (c'est-à-dire en modifiant l'implémentation de la blockchain), un tel délai peut être adapté pour supporter la *capacité* attendue. Ici, une attention spécifique doit être considérée, car elle pourrait favoriser l'apparition de blocs concurrents à travers le réseau. Enfin, cette *capacité* peut être augmentée en optimisant le contrat (c'est-à-dire en l'allégeant). D'un point de vue macroscopique, la complexité de l'algorithme qui écrit les données sur la chaîne doit être aussi faible que possible (Wood et al., 2022). L'utilisation d'un optimisateur tel que GASOL (Albert et al., 2019) est une option acceptable pour réduire les frais de gaz de contrat à cet égard.

3.3.4 Limites de l'approche

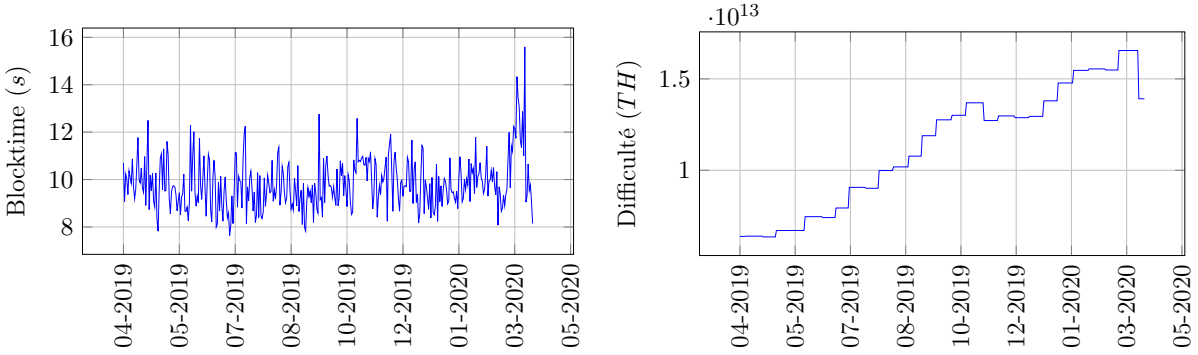
Même si les expériences ont été réalisées dans un état proche du régime permanent pour le délai d'extraction, le réseau considéré dans cette étude n'était pas soumis à une latence réseau élevée ou à une corruption des données. Lors de l'utilisation de la blockchain sur Internet, ce délai pourrait être plus important (par exemple, environ $12,6$ secondes en considérant la chaîne Bitcoin (Decker & Wattenhofer, 2013)), ce qui, dans certaines applications, pourrait entraîner des problèmes de latence et de débit de bout en bout, comme discuté dans Bez et al. (2019) ; Fan et al. (2020). L'un des impacts majeurs de l'augmentation des délais de transmission est la probabilité plus élevée de l'apparition d'oncles, formant des *forks*. En effet, la probabilité qu'un nœud n'ayant pas encore reçu le nouveau bloc trouve une solution alternative augmente avec le délai de transmission. Le contrôle de régulation sur le délai répond en augmentant la difficulté pour diminuer cette probabilité. Le contrecoup de cet ajustement est que le délai de minage d'un bloc augmente, ce qui explique pourquoi le délai de minage de la blockchain principale Ethereum est d'environ $14,4$ secondes (Pierro & Rocha, 2019), valeur plus élevée que le délai de notre réseau qui est proche de $12,6$s.

Nous soulignons également le fait que dans cette étude nous n'avons pas considéré l'évolution potentielle de la `gasLimit`. En effet, les mineurs pourraient être intéressés à augmenter cette limite (pour diminuer le nombre de blocs à miner) de sorte que la `capacité` évoluerait comme défini précédemment. Les clients tels que Geth implémentent la limitation de la variation entre deux blocs telle que définie dans Wood et al. (Wood et al., 2022) (approximativement $0,1\%$). Pour des expériences d'une durée de 10 minutes (c'est-à-dire avec une moyenne de 52 blocs), cela pourrait correspondre à une augmentation de la taille des blocs de 5% à la fin de l'expérience dans le pire des cas. La simplification est donc raisonnable.

3.4 Une adaptation : la difficulté

À travers ces expériences, nous avons pu nous abstraire dans le cadre de la preuve de travail des capacités de calculs des machines exécutant la chaîne et comme nous avons négligé le plus possible les temps de propagation, nous avons pu nous abstraire de la distribution de la chaîne entre plusieurs nœuds. Cette abstraction a été rendue possible par une boucle de régulation sur la difficulté, qui permet au système blockchain de conserver le délai de minage d'un bloc dans un intervalle fixe, quels que soient le nombre et la puissance de calcul des mineurs du réseau. Cette boucle est indispensable dans une blockchain publique et sans permissions, qui peut donc subir de grande variation non prédictible de puissance de calcul globale.

D'autres concepts similaires à la difficulté d'Ethereum sont également utilisés dans d'autres consensus basés sur la preuve de travail. Ainsi, Bitcoin exploite aussi sa propre "difficulté" qui est également liée au nombre de hachés moyens nécessaires pour trouver un bloc. Cette difficulté est représentée sur la figure 3.10a durant une période d'un an et est redéfinie toutes les deux semaines de telle manière à ce que le réseau construise 2016 blocs en exactement deux semaines, correspondants à un temps de minage moyen de 10 minutes, comme le confirme la figure 3.10a. Cependant, cette définition de difficulté est moins réactive que celle d'Ethereum puisqu'elle implique une attente de deux semaines pour recalculer la difficulté, alors qu'Ethereum le réalise à chaque bloc.



(a) Durée moyenne pour miner un bloc

(b) Difficulté moyenne des blocs

FIGURE 3.10 – Valeurs principales sur la chaîne principale Bitcoin, d'après Bitcoinity (Bitcoinity.org, 2022)

3.4.1 Définition de la difficulté

La définition de la difficulté d'un bloc utilisée dans la version actuelle d'Ethereum a été extraite du *yellow paper* d'Ethereum (Wood et al., 2022), et est donnée en (3.2) $\forall n > 0$:

$$H_d^n = max\left(H_d^0, H_d^{n-1} + \left\lfloor \frac{H_d^{n-1}}{2048} \right\rfloor \times max\left(y - \left\lfloor \frac{H_t^n - H_t^{n-1}}{9} \right\rfloor, -99\right) + \epsilon\right) \tag{3.2}$$

$$y = \begin{cases} 1 & \text{si } H^{n-1} \text{ n'admet pas d'oncles} \\ 2 & \text{dans le cas contraire,} \end{cases}$$

Où H^n se réfère à un bloc avec un numéro de séquence noté n, H_d^n est la difficulté de ce bloc, H_t^n est l'horodatage (en secondes) lorsque le bloc a été généré,

epsilon est la "bombe de difficulté" conçue pour forcer les utilisateurs à mettre à jour leur chaîne (il faut noter qu'un nombre élevé de blocs – *environ* 5 000 000 – doit être considéré pour activer une telle "bombe"), et y est un terme dépendant de l'apparition d'oncles dans le bloc précédent. Les fonctions *max* incluses dans la formule garantissent que la difficulté ne tombe pas en dessous de la valeur initiale, tout en limitant sa vitesse d'évolution. Cette relation peut être interprétée comme expliqué par Wood et al. : avant la bombe de difficulté, si aucun oncle n'apparaît, et si le délai entre deux blocs est compris entre 9 et 18 secondes, la difficulté ne change pas. Cependant, si le délai entre deux blocs n'est pas dans cet intervalle, la difficulté doit être réduite ou augmentée pour favoriser le prochain délai entre deux blocs dans cet intervalle.

3.4.2 Relation entre la difficulté et le temps de minage

Dans le contexte d'Ethereum, le contrôle par régulation lié à la difficulté repose sur une relation supposée donnée dans (3.3) où τ est le temps de minage moyen (c'est-à-dire le délai moyen pour miner un bloc), d est la difficulté de la chaîne (reposant sur une estimation statistique du nombre de hachages qui doivent être générés pour trouver une solution valide pour extraire un bloc), et $\#_{tot}$ est le taux de hachage du réseau (c'est-à-dire le nombre de hachages réalisés par un nœud par seconde). Comme la durée du bloc est fixée à environ dix secondes dans Ethereum, cette relation est souvent simplifiée par $d = 10 \times \#_{tot}$ (cf. section 3.3.1).

$$\tau = \frac{d}{\#_{tot}} \tag{3.3}$$

Cette relation peut se démontrer. D'après le *yellow paper* d'Ethereum (Wood et al., 2022), le haché h d'un bloc doit vérifier la relation suivante :

$$h \leq \frac{2^{256}}{d} \tag{3.4}$$

Le haché étant constitué de 256 bits et le résultat étant quasi aléatoire, la probabilité pour qu'un bloc donné soit une solution de l'inéquation (3.4) est donc de $\frac{1}{d}$. Les hachés étant indépendants les uns des autres grâce aux propriétés des fonctions de hachage, la détermination d'une preuve de travail est assimilable à un ensemble d'expérience de Bernoulli.

Supposons une machine donnée, disposant d'une puissance de calcul lui fournissant la capacité de réaliser des hachés avec une fréquence $\#$. Si l'on néglige le temps nécessaire pour construire un nouveau bloc (sans haché) en cas de succès, le nombre d'occurrences d'un bloc respectant la relation (3.4) durant une période Γ suit donc un processus de Poisson de paramètre $\Gamma\# \cdot \frac{1}{d} = \Gamma \cdot \lambda$ avec $\lambda = \frac{\#}{d}$.

Ainsi, le délai entre deux occurrences, c'est-à-dire le délai de minage, suit une distribution exponentielle de paramètre λ (Karr, 1984), impliquant une espérance égale à $\frac{1}{\lambda} = \frac{d}{\#}$ (relation (3.3)).

Ce modèle fournit également une explication de la propriété d'additivité du hashrate, car un réseau composé de N nœuds résulte en un système de $(n_i)_{i \in N}$ nœuds indépendants cherchant la solution du puzzle, chacun ayant un hashrate donné noté $\#_i$. Par conséquent, si l'on néglige les délais de propagations, chaque nœud n_i représente un processus de Poisson, avec un temps de minage suivant une loi exponentielle de paramètre $\frac{\#_i}{d}$. Grâce aux propriétés des processus de Poisson, le temps entre deux blocs du réseau suit donc une distribution exponentielle de paramètre $\frac{\sum_{i \in N} \#_i}{d}$. Globalement, le hashrate d'un réseau peut être déterminé en additionnant le hashrate de l'ensemble du réseau.

3.4.3 Validité du modèle

Pour vérifier la cohérence du modèle proposé et de la plateforme de référence, la relation entre la durée de la blockchain et le hashrate a été testée. Pour ce faire, une blockchain avec les mêmes paramètres initiaux (notamment la même difficulté) a été lancée avec des taux de hachages différents. La durée de l'expérience (dix minutes) était suffisamment courte pour négliger le changement de difficulté dû au contrôle par régulation. Dans l'expérience qui a produit le nombre maximum de blocs, ce qui est plus susceptible d'être influencé par ce contrôle, 64 blocs ont été produits, ce qui pourrait modifier la difficulté dans Ethereum selon l'équation (3.2) jusqu'à 3%.

En effet, pour éliminer l'influence du facteur temps de propagation, la blockchain a été exécutée sur une seule machine. On peut alors se placer dans l'hypothèse que notre réseau est suffisamment petit pour impliquer que des oncles apparaissent très rarement ($y \simeq 1$). On peut supposer aussi qu'il est très peu probable qu'un bloc mette plus de 27s a être miné ($P(X > 27s) \simeq e^{-\frac{27}{10}} = 6.7\%$ d'après le modèle, pour un délai moyen proche de 10 secondes), alors en éliminant la bombe, le max, un bloc peut modifier la difficulté d'un bloc par un facteur :

$$\frac{k}{2048} \text{ ; avec } k \in \{-1, 0, 1\}$$

Dès lors, après 64 blocs, la variation de la difficulté est inférieur à $\Delta d = |(1 \pm \frac{1}{2048})^{64} - 1| < 3,2\%$, d'après le modèle.

La variation du hashrate a été réalisée en modifiant le nombre de *threads* alloués au minage. Chaque *thread* utilise un cœur de la machine et procède de manière indépendante grâce au *multithreading*. Tant que les *threads* ne sont pas en compétition entre eux par manque de cœur physique, on peut considérer qu'ils ont le même hashrate, car ils ont été exécutés sur des cœurs similaires.

Dans cette hypothèse, le problème est équivalent à la relation donnée dans 3.5, où $\#_{thread}$ fait référence au hashrate d'un seul *thread* supposé constant, et nb_{thread} au nombre de *threads* utilisés. La figure 3.11 fournit un aperçu des résultats obtenus pour une expérience réalisée dix fois par nombre de *thread*, avec le nombre de *thread* allant de 0 à 4. La relation entre le temps Interbloc et le nombre de *threads* est inversement proportionnelle. La figure 3.12 confirme cette hypothèse en indiquant une relation linéaire entre l'inverse du délai et le nombre de *threads*, suivant une régression linéaire avec $\rho = 0,91$. Lorsque l'on considère un grand nombre de *threads*, cette relation devient obsolète puisque les *threads* ont commencé à se faire concurrence sur la même machine.

$$\tau = \frac{d}{\#_{thread} \times nb_{thread}} \tag{3.5}$$

3.4.4 Résolution du calcul de la difficulté

Dans les sections précédentes, nous avons utilisé l'approximation $d = 10 \times \#_{tot}$, qui implique que le temps de minage d'un bloc en Ethereum est approximativement autour de 10 secondes. Cette relation n'est pas démontrée et plusieurs analyses ont cherché à estimer le temps de confirmation d'une preuve de travail (Vora & Gor, 2022 ; W. Zhao et al., 2019).

Nous allons ici présenter une estimation de ce délai de minage basé sur la définition de la difficulté (3.2).

Si l'on reprend cette définition et que nous la simplifions en supposant que la bombe de difficulté n'est pas activée et est donc négligeable ($\epsilon \simeq 0$), que l'on est bien au-dessus de la difficulté

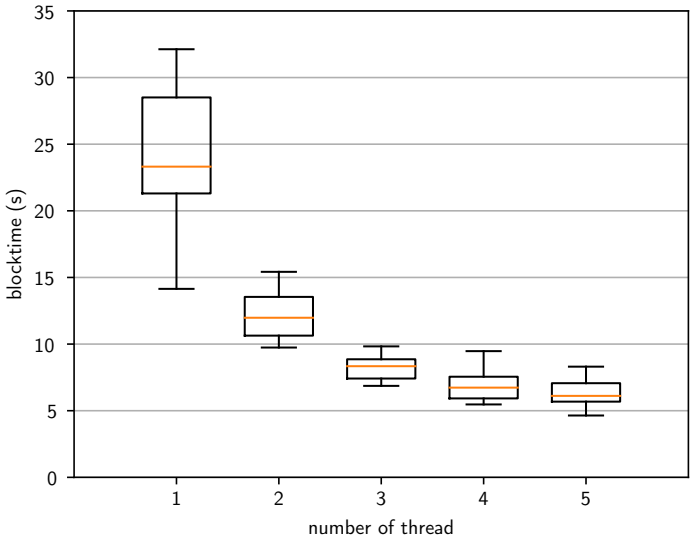


FIGURE 3.11 – Évolution du temps de minage moyen mesuré en fonction du nombre de *threads* sur une machine

initiale ($H_d^n >> H_d^0$ pour n suffisamment grand) et que la probabilité d'un délai de minage supérieur à 900 secondes est négligeable, on obtient la relation (3.6). Pour plus de simplicité, H_d a été remplacé par d, la durée de minage par τ et la valeur 9 par l.

$$d_n - d_{n-1} = \left\lfloor \frac{d_{n-1}}{2048} \right\rfloor \times \left(y - \left\lfloor \frac{\tau_n}{l} \right\rfloor \right) \tag{3.6}$$

$$y = \begin{cases} 1 & \text{si } H^{n-1} \text{ n'admet pas d'oncles} \\ 2 & \text{dans le cas contraire,} \end{cases}$$

Dès lors, nous allons faire deux hypothèses fortes. Tout d'abord, la durée de minage des blocs est assimilable à une distribution de paramètre λ. Ce paramètre dépend de la difficulté, varie avec elle et tend vers une asymptote grâce à la boucle de régulation. La première hypothèse est que l'on est si proche de cette asymptote que l'on considère λ constant. La seconde hypothèse est que comme on est très proche de cette asymptote et donc du régime permanent, la difficulté varie très peu et se compense. Ainsi, l'espérance de la variation de la difficulté est égale à zéro.

$$\begin{aligned} E(d_n - d_{n-1}) = 0 &= E\left(\left\lfloor \frac{d_{n-1}}{2048} \right\rfloor \times \left(y - \left\lfloor \frac{\tau_n}{l} \right\rfloor \right) \right) \\ &= \left\lfloor \frac{d_{n-1}}{2048} \right\rfloor \times E\left(y - \left\lfloor \frac{\tau_n}{l} \right\rfloor \right) \end{aligned}$$

Le paramètre y permet de prendre en compte la durée de propagation des blocs. Si la durée de propagation est grande alors la probabilité d'apparitions d'oncle pendant la propagation du bloc est élevée. La boucle réagit en rallongeant la durée de minage des blocs pour éviter la formation de blocs concurrents.

Pour modéliser l'influence de ce paramètre y, nous allons définir un temps de propagation δ_t. Dès qu'un nœud i minera un bloc, il l'enverra à l'ensemble des nœuds du réseau en exactement

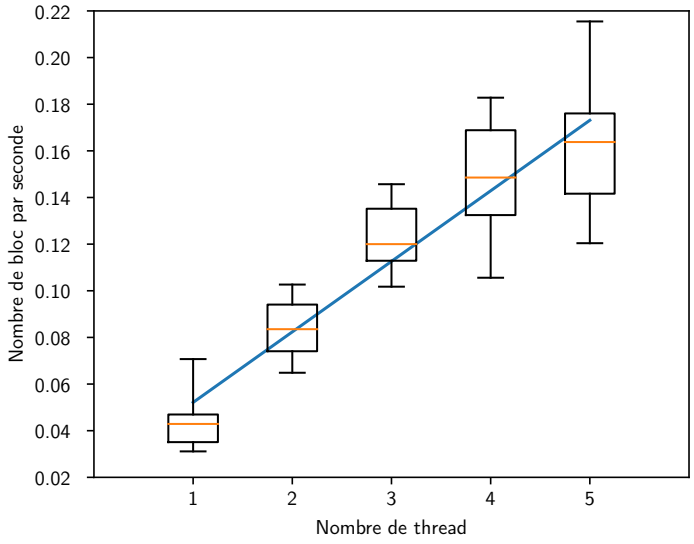


FIGURE 3.12 – Évolution de l'inverse du temps de minage moyen mesuré en fonction du nombre de *threads* sur une machine

δ_t secondes. De plus, on considère que le réseau est suffisamment grand pour négliger le taux de hachage du nœud i :

$$\#_i << \#_{tot} \tag{3.7}$$

Grâce à ces hypothèses, le nombre de blocs apparu dans le reste du réseau pendant cette durée δ_t suit une loi de Poisson de paramètre $\delta_t \cdot \lambda$. Par ailleurs, comme on a considéré λ constant, y et $\left\lfloor \frac{\tau_n}{l} \right\rfloor$ sont considérés comme indépendants. On peut donc définir :

$$\begin{aligned} 0 &= E\left(y - \left\lfloor \frac{\tau_n}{l} \right\rfloor\right) \\ &= E\left(y\right) - E\left(\left\lfloor \frac{\tau_n}{l} \right\rfloor\right) \end{aligned} \tag{3.8}$$

Le nombre d'oncles suivant une loi de Poisson P_P, on a :

$$E\left(y\right) = 1 \cdot P_P(X = 0) + 2 \cdot P_P(X > 0) = 2 - P_P(X = 0) = 2 - e^{-\lambda \delta_t} \tag{3.9}$$

De plus, comme la durée de minage suit une loi exponentielle P de paramètre λ :

$$\begin{aligned} E\left(\left\lfloor \frac{\tau_n}{l} \right\rfloor\right) &= \sum_k^\infty kP\left(kl \leq X < (k+1)\, l\right) = (1 - e^{-\lambda l}) \sum_k^\infty k e^{-\lambda k l} \\ &= \frac{e^{-\lambda l}}{1 - e^{-\lambda l}} \end{aligned} \tag{3.10}$$

Ainsi, en utilisant (3.8), (3.9) et (3.10), on obtient que le paramètre $\lambda = \frac{1}{\tau}$ doive vérifier :

$$2 = e^{-\frac{\delta_t}{\tau}} + 3e^{-\frac{l}{\tau}} - e^{-\frac{(\delta_t + l)}{\tau}} \tag{3.11}$$

La figure 3.13 fournit un ensemble de solutions approché pour τ en fonction d'un paramètre δ_t. Ainsi, dans le cas où les délais de propagation sont nuls, le délai de minage est égal à $12,98$s et peut monter jusqu'à $22,20$s quand le délai de propagation tend vers l'infini.

La chaîne principale d'Ethereum ayant un délai de minage de l'ordre de $13,10$s, nous évaluons le temps de propagation de l'ordre de 160ms. Ce temps est dans l'ordre de grandeur du temps de propagation mesuré sur la chaîne principale, qui est de l'ordre de 100ms pour 85% des nœuds (Kiffer et al., 2021).

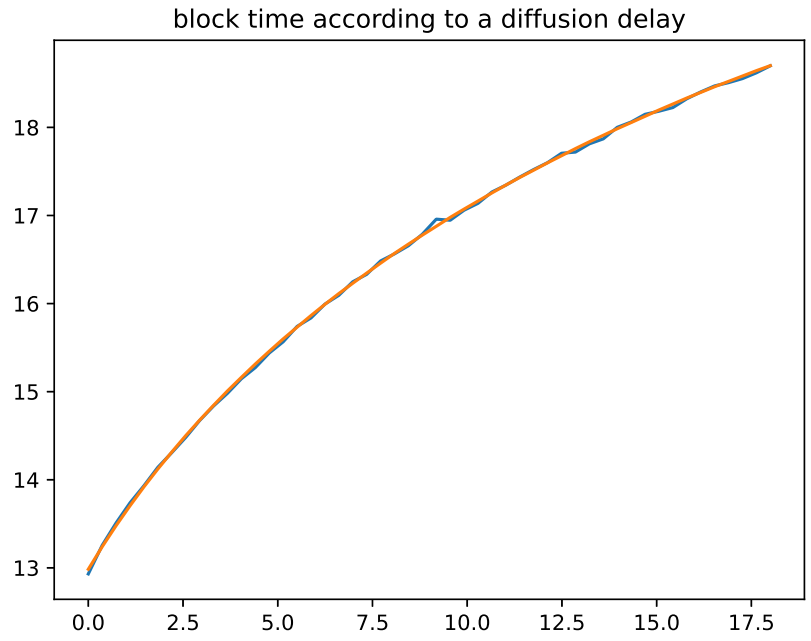


FIGURE 3.13 – Solution approchée du délai de minage τ de l'équation (3.11) en fonction d'un délai de propagation δ_t

3.5 Conclusion

L'étude des liens entre les différents constituants d'une blockchain montre qu'un concepteur d'application ne dispose que de très peu de leviers pour adapter les performances d'une blockchain dans le cas d'une preuve de travail de type Ethereum. Le seul véritable paramètre modifiable est la taille du bloc. Les performances matérielles et le nombre de nœuds sont des éléments qui peuvent être ignorés grâce à l'introduction de la difficulté qui vient asservir une durée entre les blocs relativement constante. De plus, la mesure de performance QoS la plus importante est le débit. En effet, au-delà d'un débit particulier que nous appelons *capacité*, la latence tend vers l'infini. Ce débit est également le débit maximal réalisable (Tx/s) à long terme, limite ce débit atteignable. Améliorer les performances revient donc à augmenter ce débit. Peu de leviers sont disponibles pour augmenter la capacité. Nous n'avons identifié que le débit de génération de bloc, la taille du bloc et la taille des transactions.

Outre la spécification de l'écosystème FarMarket, une méthodologie complète a été introduite pour aider les intégrateurs de solutions logicielles à mieux comprendre (et mesurer) les

performances de qualité de service qu'un écosystème de type FarMarket pourrait atteindre et supporter à long terme. Une attention particulière a été accordée à la *capacité* dans cette étude. Les analyses expérimentales présentées dans cet article devraient conduire à des discussions intéressantes concernant les aspects/interactions critiques à prendre en compte entre les paramètres liés à la blockchain et à l'infrastructure. À notre connaissance, il n'existe que peu de travaux de recherche qui examinent en profondeur de telles interactions et la manière dont ils influencent les performances globales (de bout en bout) en matière de qualité de service.

Dans le cas d'Ethereum, de nombreux liens relatifs aux capacités de calculs du réseau ont pu être ignorés grâce à l'évolution de la difficulté au cours du temps. Cette évolution se réalise grâce à une boucle de régulation permettant d'assurer une génération de bloc avec un débit constant. En se basant sur des processus de Poisson, ce débit moyen a pu être déterminé et ne dépend que de deux paramètres dans notre modèle, un temps de propagation et un paramètre "l" codé en dur dans le consensus. Ces deux valeurs s'ajoutent à la taille du bloc et la taille des transactions, pour former quatre paramètres personnalisables pour améliorer les performances d'une chaîne.

Par la suite, la boucle de régulation sur la difficulté est un formidable outil pour conserver une génération de bloc avec un débit constant. Le système fait varier la difficulté, et par conséquent la sécurité pour adapter ses performances. Peu de solutions existent dans d'autres consensus pour assurer ce débit constant, rendant les performances des chaînes associées sensibles à l'arrivée de nouveaux nœuds dans la chaîne (le consensus par vote étant un bon exemple). Une modification du consensus faisant varier la sécurité pour atteindre des performances constantes serait alors souhaitable, à l'instar d'Ethereum, mais dans le cadre d'un consensus BFT. Le chapitre suivant s'attachera ainsi à l'introduction de telle boucle d'asservissement.

4

Adaptation en ligne de la taille du pool de validateurs

Le consensus Practical Bizantine Fault Tolerance (PBFT) est un protocole de consensus largement adopté dans les blockchains privées et de consortium, la raison étant qu'il s'agit d'un protocole léger, peu gourmand en ressources informatiques et qu'il permet d'atteindre un débit de transaction élevé. La sécurité de ce consensus dépend du nombre de nœuds (alias validateurs) impliqués dans le consensus : plus le nombre de nœuds est élevé, plus le niveau de sécurité est élevé. Cependant, l'augmentation du nombre de validateurs s'accompagne d'une augmentation de la latence, et par conséquent d'une diminution du débit ou encore de la capacité, notion introduite au chapitre précédent.

La section précédente nous a prouvé l'existence de boucle de régulation permettant de conserver des performances constantes malgré l'arrivée de nouveaux nœuds dans le réseau. Le consensus présenté concernait un exemple de consensus basé sur des preuves, mais l'état de l'art nous a montré que ce genre de boucle existait sur les consensus à base de vote. En effet, les blockchains hybrides basés sur BFT vu en section 2.2.3 répondent à cette définition. En séparant validateurs et non-validateurs, ils permettent de conserver un noyau de nœuds réalisant un protocole BFT de taille constant, ayant par conséquent des performances constantes.

Cependant, l'état actuel de la littérature présente des blockchains avec un nombre de validateurs fixe. Notre hypothèse est alors qu'il puisse être intéressant d'imaginer et étudier une nouvelle approche visant à faire varier le nombre de validateurs dans le but d'avoir des performances s'adaptant à la demande. Par exemple, dans le cas où une blockchain serait peu sollicitée, le consensus disposerait de suffisamment de marge pour se permettre d'augmenter le nombre de validateurs, augmentant de fait la sécurité et amenant un modèle de cohérence plus fort.

La présente recherche propose un nouveau protocole appelé Self-Adaptive BlockchaIn coNsEnsus (Sabine) (cf. Leduc et al. (2022a, 2022b), qui tente de résoudre de manière optimale le compromis entre sécurité et débit. En d'autres termes, SABINE vise à adapter continuellement la taille du pool de validateurs avec pour objectif principal que le débit de sortie (c'est-à-dire le nombre de transactions validées à un moment donné) corresponde – *dans la mesure du possible* – au débit d'entrée (c'est-à-dire celui demandé par l'application). Sabine est évaluée et validé dans des contextes réels, dont les résultats montrent que Sabine a une erreur relative de $7,79\%$ entre le débit de transaction demandé et le débit de transaction engagé, comparé à $39,5\%$ pour une chaîne classique avec tous les nœuds participant au consensus, ce qui implique une chaîne plus réactive, avec moins de latence.

Ce chapitre se divise en quatre grandes étapes. Des tests ont été réalisés sur une blockchain

développée spécialement pour cette étude. Celle-ci a pour objectif de minimiser toutes les briques logicielles autres que le consensus pour pouvoir mesurer les performances associées au consensus. L'architecture et les plateformes supportant cette chaîne sont décrites en section 4.1. Par la suite, après avoir réalisé une analyse des impacts du nombre de validateurs sur le consensus, les objectifs de la boucle de régulation de Sabine sont décrits en section 4.2. Une fois ces objectifs définis, la constitution d'un modèle et les différentes étapes de l'algorithme de Sabine sont décrites en section 4.3. Puis, les impacts de Sabine sont évalués sur des chaînes en section 4.4 en faisant varier à la fois le débit soumis et le délai du réseau.

4.1 Description des bancs d'expérimentations

Afin d'évaluer les performances des différentes propositions de modification du protocole PBFT, ces modifications sont testées sur une blockchain implémentant ce consensus et lancées sur une plateforme expérimentale. Une multitude d'implémentations existe et une recherche sur Github avec le mot-clef "PBFT" propose près de 271 résultats en lien avec ce protocole. Parmi ces différentes propositions, on retrouve ainsi la version PBFT de Hyperledger Fabric et de Hyperledger Sawtooth (Hyperledger, 2022). Cependant, ces propositions s'inscrivent dans le cadre d'applications spécifiques. Ainsi, de nombreuses briques logicielles viennent complexifier la blockchain et perturber les performances, comme le chiffrement des messages ou la vérification des transactions. Dans le but d'évaluer le plus fidèlement le consensus PBFT et de ne pas prendre en compte les performances de ces autres briques par effet de bord, une blockchain a été spécialement conçue. Son architecture est présentée en section 4.1.1.

Cette blockchain est lancée sur deux plateformes différentes avec des performances propres, de manière à montrer que Sabine peut s'appliquer à toute plateforme. Ces plateformes sont décrites en section 4.1.2.

4.1.1 Structure de la chaîne

Afin de cibler précisément la couche de consensus dans le modèle 1.5, une nouvelle blockchain a été développée qui minimise les apports de toutes les autres couches. Le code de ce projet est disponible sur `https://github.com/inpprenable/Sabine`. Sa conception est détaillée dans cette section à travers la pile de conception.

Couche donnée

Le langage utilisé pour l'implémentation du programme est le langage Go (Google, 2012). Ce langage a été choisi pour sa facilité de parallélisation (notamment pour les services réseau), et la compilation en exécutable léger, facilement exportable vers tout type de support. De plus, il s'agit d'un langage de bas niveau ayant des performances très rapides.

Un grand soin a été donné pour optimiser le code, en adoptant des fonctions et structures ayant des performances en $O(1)$ ou en $O(logn)$ dans le pire des cas. Ce choix a été réalisé afin de minimiser l'impact de la gestion des messages et des blocs en arrière-plan, mais surtout de garder des performances égales au fur et à mesure de l'ajout de bloc dans la chaîne. La mesure des durées des étapes comme représentée sur la figure 4.6 montre que la plupart des étapes ne dépendent pas de la longueur de la chaîne.

Concernant les blocs, il a été défini qu'un bloc est constitué d'au maximum 5 transactions. Il s'agit d'une taille arbitraire suffisamment faible pour favoriser l'apparition de blocs, car l'objectif

des expérimentations reste l'analyse du consensus PBFT. Dès qu'un nœud devient Proposer et dispose d'un ensemble de transactions dans son pool de transaction non effectuée, il tente d'en ajouter un maximum dans un bloc et le propose au réseau.

À la fin de l'expérience, un signal est envoyé aux nœuds pour s'éteindre. À ce moment, les nœuds effectuent une sauvegarde de leurs chaînes dans un fichier au format JSON. Dans certaines expériences, une sauvegarde est réalisée périodiquement au cas où le nœud subirait une défaillance.

Couche réseau

Le maillage entre les nœuds adopte une topologie égale à un graphe complet comme décrite en section 1.4.2. Il n'y a donc pas de routage entre nœuds lors de la transmission d'un message. Chaque nœud possède donc une connexion TCP avec tous les autres nœuds du réseau et l'utilise pour transporter les messages. Ces connexions sont non chiffrées par économie de temps et de ressources.

Cette topologie complète a pu se réaliser grâce à un protocole de bootstrap décrit dans la figure 4.1a. Ainsi, lors de l'initialisation d'un nœud, celui-ci récupère la liste des nœuds sur un serveur particulier et s'y inscrit également. Puis, il réalise des connexions avec l'ensemble des autres membres de la liste.

Aucun protocole de récupération n'est prévu dans l'implémentation et les informations ne sont pas persistantes. Les nœuds ne sont pas supposés être défaillants. Ainsi, si un nœud manque un bloc ou subit une panne, il n'a pas de moyens de le récupérer, et donc de continuer le consensus. Une désynchronisation d'un bloc suffit à un nœud pour perdre la chaîne. Ainsi, une hypothèse que nous admettons par la suite est qu'aucun nœud ne peut défaillir. La récupération, bien qu'elle soit nécessaire pour obtenir une blockchain CFT, et donc BFT, est un mécanisme non traité dans cette étude.

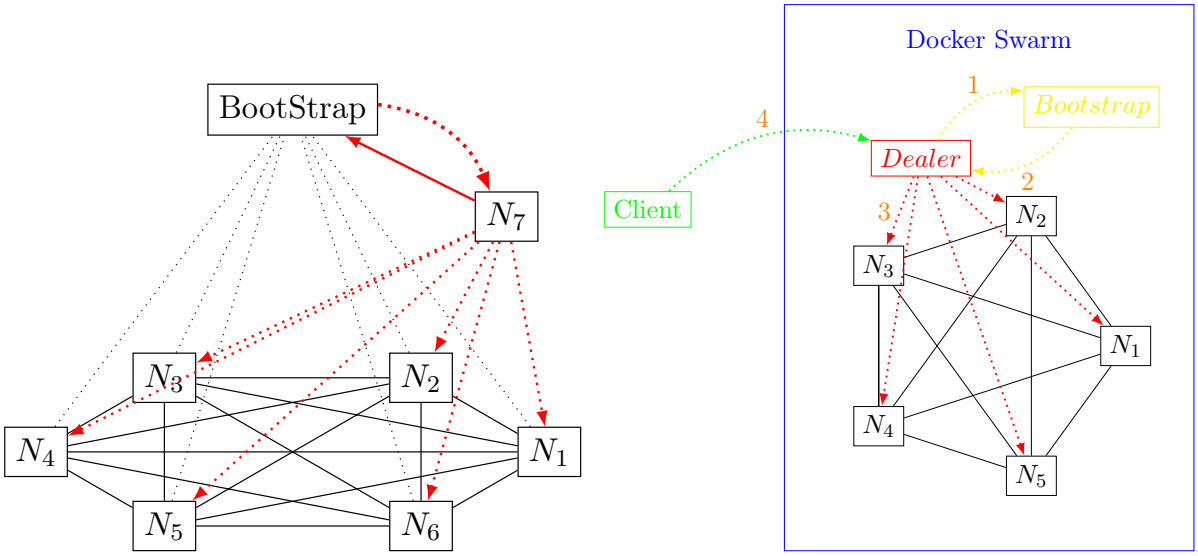
Concernant la diffusion des transactions, aucun protocole de commérage (ou bavardage) n'est mis en œuvre pour réduire le nombre de messages. Pour éviter que les nœuds ne perdent des ressources en redistribuant les propositions de transaction tout en s'assurant que chaque transaction est bien distribuée à chaque nœud, un serveur particulier nommé Dealer a été mis en place. Celui-ci récupère la liste des adresses des nœuds grâce au serveur de Bootstrap, puis le Client se connecte au serveur Dealer pour diffuser les propositions à l'ensemble du réseau. La figure 4.1b représente la connexion de ce serveur en orange, puis le cheminement d'une transaction en rouge.

Couche permission

Concernant les permissions, la blockchain est privée sans permission. Ainsi, seuls les membres autorisés peuvent participer et aucune distinction n'est faite entre les membres. L'identification et l'authentification des messages se réalisent grâce à des signatures asymétriques (cf. section 1.1.3). Afin d'éviter un échange des clefs, celles-ci sont fournies au démarrage aux différents nœuds.

Couche consensus

Description du consensus Sabine est testée dans une chaîne qui utilise une variante du consensus IBFT, qui est lui-même fortement inspiré du consensus PBFT. L'IBFT (*Istanbul Byzantine Fault Tolerance consensus*) est un algorithme de consensus tolérant aux fautes byzantines tolérant f processus défectueux sur n, où $n \geq 3f + 1$ (Moniz, 2020). Il a été développé comme alternative au consensus d'Ethereum, dans le cadre de l'EIP 650 d'Ethereum (Y.-T. Lin, 2017).



(a) Représentation de l'appareillage d'un nœud avec le bootstrap

(b) Représentation de la diffusion des transactions

FIGURE 4.1 – Divers protocoles utilisés par la chaîne

Comme PBFT, IBFT assure les propriétés de sûreté et de vivacité en réseau synchrone et uniquement la sûreté en réseau asynchrone.

Le mécanisme de vote pour le *commit* de bloc est proche du consensus PBFT original de Castro-Liskov décrit en section 1.4.4, et deux états ont été ajoutés pour les cas de non-validation *New Round Proposer*, correspondant à l'état d'un nœud au début d'une nouvelle epoch de bloc, dans le cas où ce nœud serait le Proposer. L'état *New Round NV*, correspond au cas où le nœud devient non-validateur (cf. 4.2.1) et en attente d'un bloc.

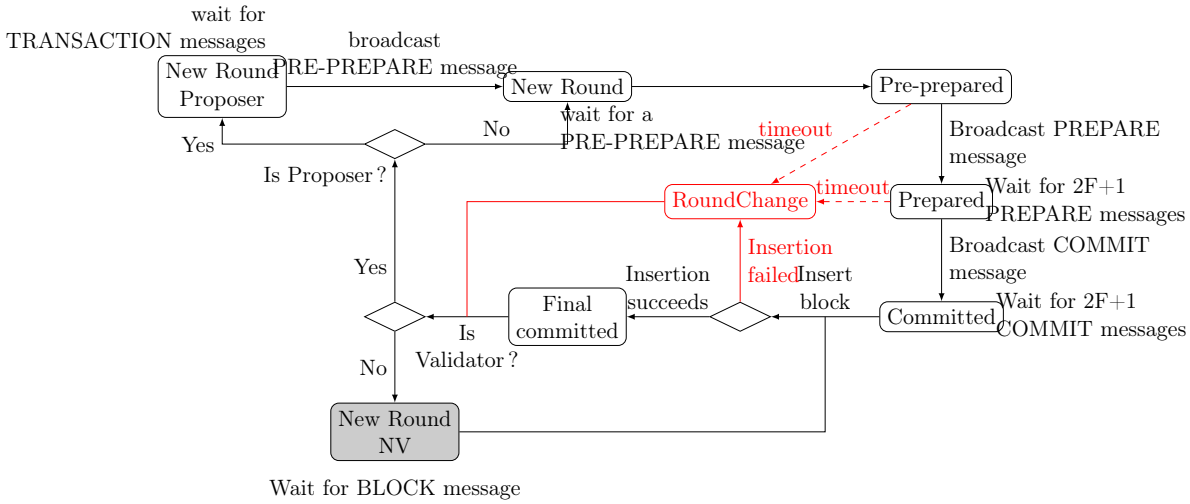


FIGURE 4.2 – Diagramme d'état de la version modifiée du consensus IBFT

Le diagramme d'état de la chaîne testée est représenté dans la figure 4.2. L'implémentation sous forme de machine à état permet de limiter l'émergence de blocage. L'étape *RoundChange* représente les cas où le proposant a échoué ou est suspecté d'être malveillant, mais elle n'est pas

implémentée, car l'étude ne se concentre pas sur les nœuds malveillants ou en panne.

Choix aléatoire du Proposer Un nouveau *Proposer* est voté à chaque nouveau *commit* d'un bloc. Dans le cas où le nouveau *Proposer* serait indisponible, la chaîne ne peut plus évoluer.

Le *Proposer* $H_{t+1}^{Proposer}$ du bloc $t+1$ est sélectionné en fonction du haché du bloc. L'identifiant de ce Proposer est calculé à partir de l'équation 4.1 en fonction du nombre de validateurs n_t du bloc t. Comme le haché est constitué d'une chaîne d'octets, l'identifiant du validateur ne peut excéder 256.

$$(H_{t+1}^{Proposer})_{id} = H_t^{\#}[0] \mod n_t \tag{4.1}$$

Le haché était une variable pseudoaléatoire, on peut considérer que chaque caractère forme un nombre aléatoire entre 0 et 255. On sélectionne alors ce premier caractère. L'utilisation du modulo dans l'équation 4.1 permet donc de générer un nombre pseudoaléatoire entre 0 et n_t. L'intérêt de cette méthode est que l'élection d'un *Proposer* et l'ajout d'un bloc sont confondus. Ainsi, à chaque *epoch*, il y a en même temps l'élection d'un nouveau *Proposer* (dont le mandat ne sera valable que pour 1 tour).

Couche contrat

L'utilisation de *smart contract* par la chaîne demande de mettre en place un environnement contrôlé pour pouvoir vérifier la bonne exécution des contrats sur toute plateforme. Ainsi, Ethereum met en place une machine virtuelle appelée EVM pour exécuter ces contrats. Pour éviter d'avoir à évaluer cet environnement, la chaîne n'exécute pas de *smart contracts*. Les transactions consistent simplement en des chaînes de caractères qui ne sont pas vérifiés par la chaîne.

Par ailleurs, l'historique des blocs est simplement conservé en mémoire vive, sans recours à une base de données externe.

Couche application

Dans le but de tester la chaîne, des transactions sont envoyées aux réseaux par le serveur "Dealer". Un serveur dédié appelé "Client" est créé et envoie les transactions avec un débit donné. Ce serveur dispose de ressources non limitées pour répondre au débit qui lui est soumis. Les transactions envoyées sont de simples chaînes de caractères contenant une nonce qui s'incrémente à chaque nouvelle transaction. Les transactions, et donc les blocs, ont donc toutes la même taille. Les nœuds participants à la blockchain ne réalisent pas d'interactions avec d'autres programmes.

4.1.2 Description des plateformes

Les expériences ont été effectuées sur deux bancs d'expériences. Les conditions expérimentales d'exécutions et les spécifications des différents bancs sont décrites dans cette section.

Exécutions sur conteneur

Afin de pouvoir contrôler finement l'environnement d'exécution de la chaîne, celle-ci s'exécute sur les différents nœuds sur des Dockers. Il s'agit de conteneur logiciel permettant de réaliser des processus isolés plus simples et plus légers que des machines virtuelles. L'un des avantages est de pouvoir contrôler les ressources en matière de processeur et de mémoire vive dédié. Ainsi,

dans nos expériences, plusieurs nœuds s'exécutent sur une même machine. La conteneurisation nous permet donc d'exécuter des nœuds isolés et dont les ressources ne sont pas en concurrence. De plus, comme plusieurs machines exécutent un ensemble de nœuds et que ces machines ont des spécifications identiques, on peut donc affirmer que les nœuds sont homogènes et disposent donc des mêmes capacités de calcul et de mémoire. Par ailleurs, l'orchestration des processus et le routage des messages sont réalisés par Docker Swarm.

Premier banc : Raspberry Pi

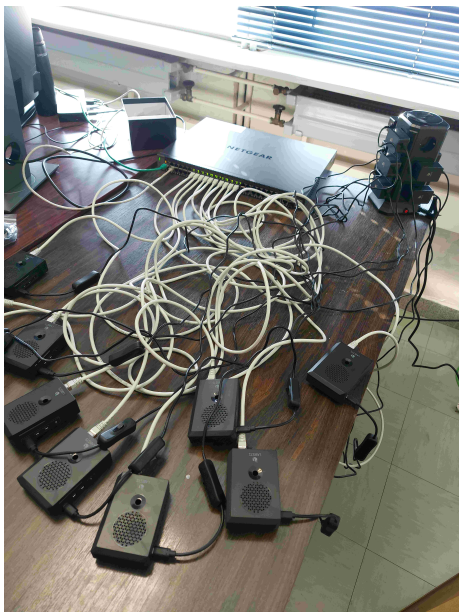


FIGURE 4.3 – Photo du banc d'expérience composé de Raspberry PI

Dans le premier banc d'expérience, les nœuds sont déployés sur un cluster de 10 Raspberry PI 4, représenté en figure 4.3. Ces derniers disposent d'un processeur SoC 64 bits à quadruple cœur A72 (ARM v8) et d'une mémoire vive de 4 Go de SDRAM LPDDR4. Un système d'exploitation Ubuntu 20.04 serveur y est installé. 50 nœuds sont déployés sur des dockers avec des ressources dédiées (0,5 CPU et jusqu'à 1 Go de Ram par nœud) pour considérer des nœuds indépendants sur une seule machine. Les Raspberry PI sont directement connectés en Ethernet avec un commutateur dédié afin de minimiser la latence du réseau. Les transactions sont générées et diffusées par un ordinateur externe sans limite de ressources afin de maintenir stable le débit des demandes de transactions.

Second banc : Grid 5000

Dans le second banc d'expérience, les nœuds sont déployés sur le Grid5000 (Balouek et al., 2013). Il s'agit d'un banc d'essai flexible et à grande échelle pour la recherche expérimentale dans tous les domaines de l'informatique. Il est composé de plusieurs machines physiques dans plusieurs universités de France. Dans notre banc d'expérience, les expérimentations se sont réalisé sur une grappe de 4 machines physiques composées de 2 Intel Xeon E5-2630 et 128 Go de mémoire vive. Environ 50 nœuds sont déployés sur des dockers sur chaque machine pour un total de 200 nœuds avec des ressources dédiées (0,3 CPU et jusqu'à 1 Go de Ram par nœud) pour considérer des

nœuds indépendants sur une seule machine. Les machines sont également directement connectées à un commutateur. Les transactions sont générées et diffusées par un "Client" sur une machine externe sans limite de ressources.

4.2 Objectif de la démarche

Comme étudié précédemment dans la section 3.4, la sécurité d'Ethereum repose sur une notion appelée difficulté et cette difficulté évolue en fonction du nombre de mineurs pour conserver des performances constantes. Dans le cas du consensus PBFT, un tel mécanisme n'existe pas. Ainsi, les performances diminuent avec l'arrivée de nouveaux membres dans le réseau. Le cas des consensus BFT hybrides décrits dans la section 2.2.3 montre une proposition pour conserver des performances constantes en sélectionnant un noyau de taille fixe de validateurs parmi les nœuds du réseau. La section 4.2.1 ci-dessous étudie les impacts du nombre de validateurs dans une blockchain et la difficulté pour trouver un nombre de validateurs idéal. Par la suite, la section 4.2.2 explicite notre proposition d'adaptation du consensus.

4.2.1 Impact du nombre de validateurs

Étude théorique du nombre de nœuds

Sécurité La sécurité de la chaîne est directement liée à la sécurité du consensus. Dans le cas des consensus BFT, le problème des généraux non signé y est traité, impliquant qu'un ensemble de nœuds réalisant ce type de consensus peut tolérer au plus un tiers de nœuds malveillants. Si l'on sépare ces nœuds en deux ensembles dont l'un est validateur, la sécurité du consensus dépend de la taille de l'ensemble réalisant le protocole. Celui-ci doit donc tolérer au plus un tiers de malveillants. Le nombre de malveillant f doit donc respecter l'inéquation (4.2), impliquant que le nombre de malveillant toléré augmente avec le nombre de validateurs n.

$$f < \frac{n}{3} \tag{4.2}$$

Nombre de messages Dans le cadre d'un consensus PBFT où tous les nœuds sont validateurs, le nombre de messages nécessaires pour ajouter un bloc est défini à l'équation (4.3). Comme énoncé précédemment, ce nombre de messages est quadratique avec le nombre de nœuds $\mathcal{N}$.

$$Nb_{mess} = (2\mathcal{N} + 1).(\mathcal{N} - 1) \tag{4.3}$$

Afin d'accélérer le consensus, une séparation entre validateurs et non-validateurs est possible comme pour les consensus BFT hybrides. La principale conséquence est que le nombre de messages est alors dépendant du nombre de validateurs. Cependant, les nœuds non-validateurs doivent toujours recevoir des messages pour mettre à jours leurs copies de la chaîne. Deux propositions sont possibles pour pouvoir les mettre à jour.

La première est que l'ensemble des messages intermédiaire du protocole PBFT, c'est-à-dire les messages *pre-prepare*, *prepare* et *commit*, soient renvoyés à l'ensemble des nœuds, afin que chacun exécute le protocole PBFT pour reconstituer la chaîne. Cette proposition a l'avantage de décentraliser entièrement sur les validateurs la responsabilité de l'envoi des messages et assure pour les non-validateurs la bonne exécution du protocole. Cependant, le nombre de messages est également proportionnel au nombre de validateurs et répond à la relation :

$$Nb_{mess} = \mathcal{N} - 1 + 2n.(\mathcal{N} - 1) \tag{4.4}$$

La seconde proposition est que les validateurs n'envoient les messages relatifs au consensus PBFT qu'entre eux et qu'à la suite de l'ajout d'un bloc, un des validateurs, le Proposer par exemple, envoie à l'ensemble des non-validateurs le bloc. Cette proposition diminue fortement le nombre de messages nécessaire dans le réseau qui est alors donné à l'équation (4.5), rendant le consensus quadratique avec le nombre de validateurs au lieu du nombre de nœuds.

$$Nb_{message} = \mathcal{N} - 1 + 2n \cdot (n - 1) \tag{4.5}$$

Cependant, elle crée également un tiers de confiance, puisque le consensus est proche d'une Preuve d'Autorité (PoA) pour sa part. De plus, pour que les non-validateurs puissent s'assurer de l'honnêteté du bloc reçu, une preuve de confiance doit être associée avec le bloc. Cette preuve peut consister en un agrégat de l'ensemble des signatures des messages de *commit* reçus par le nœud diffusant le bloc, prouvant donc sa bonne réalisation du consensus.

La comparaison du nombre de messages entre ces deux propositions est réalisée sur le graphique de la figure 4.4 pour un total de 200 nœuds. Dans le cas de nos expérimentations, nous allons employer la seconde proposition, avec le Proposer qui diffuse le bloc à l'ensemble des autres nœuds, car comme le prouvent les résultats de la figure, il s'agit de la proposition minimisant le nombre de messages. Cependant, nous n'associons pas avec ce bloc un agrégat des messages de *commit*.

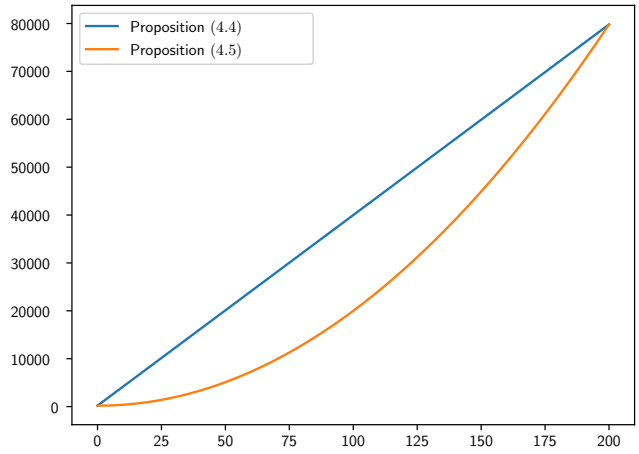


FIGURE 4.4 – Comparaison entre le nombre de messages avec et sans agrégat des messages intermédiaires, en fonction du nombre de validateurs pour 200 nœuds

Cohérence Cette séparation entre validateurs et non-validateurs amène également un changement de modèle de cohérence entre ces deux ensembles. En effet, les validateurs réalisent le consensus PBFT, qui impose une cohérence stricte dans le réseau. L'une des conséquences de cette cohérence stricte est qu'au moins $^2/_3$ des validateurs disposent d'une chaîne à jour avec le bloc courant. Ainsi, si un client quelconque réalise une lecture d'une donnée sur un des validateurs, il y a plus de 66% de chance que cette donnée soit à jour. Pour en être certain, le

client devrait réaliser une requête en lecture sur l'ensemble des validateurs et garder le choix majoritaire.

Dans le cas des non-validateurs, leur consensus est assimilable à une preuve d'autorité. Le modèle de cohérence est plus faible et est appelé cohérence éventuelle, impliquant qu'à partir d'un certain temps les mis à jours de la chaîne seront peut-être appliqués. Ainsi, aucune assurance n'est donnée sur la révision d'un des nœuds. Les messages transmit par le validateur dédié peuvent être en retard ou perdu sans que cela n'impacte ne consensus. Par conséquent, si un client quelconque réalise une lecture d'une donnée sur un non-validateur, il n'a aucune assurance de savoir si la valeur lue est à jour.

Les clients ne connaissent pas toujours en temps réel les validateurs sélectionnés puisque le protocole de sélection peut introduire de l'aléatoire comme les consensus hybrides basés sur BFT vus en section 2.2.3. Les nœuds que le client contacte ne sont alors pas forcément cohérents et leur base de données n'est pas nécessairement à jour. Par conséquent, si un client réalise une lecture d'une donnée sur un nœud quelconque du réseau, la probabilité que la valeur reçue soit à jour est égale à un taux de confiance $tx_{confiance}$ définie dans l'équation (4.6). Ce taux augmente linéairement avec le nombre de validateurs impliquant que plus le nombre de validateurs est élevé, plus la cohérence globale de la chaîne est forte et plus un nœud quelconque de la chaîne a de la chance d'être à jour.

$$tx_{confiance} = \frac{2}{3} \cdot \frac{n}{\mathcal{N}} \tag{4.6}$$

Étude pratique sur les plateformes

Afin d'illustrer les problématiques liées à la sélection du nombre de validateurs, une même demande a été testée sur les deux plateformes tout en faisant varier le nombre de validateurs selon trois configurations différentes :

— si l'on minimise le nombre de validateurs ;

— si l'on maximise le nombre de validateurs ;

— en connaissant la demande, on peut réaliser une estimation hors ligne du nombre de validateurs nécessaire pour répondre à la moyenne de la demande. On fixe alors directement la valeur à celle précalculée.

La demande consiste en un ensemble de transactions avec un débit donné. La réponse à cette demande, c'est-à-dire les transactions validées, est mesurée et le débit de validation des transactions est affiché sur les figures 4.11b et 4.11d. Le tableau 4.1 résume les résultats expérimentaux. La colonne *Écart Débit* est égale à l'erreur relative entre le débit des transactions proposées en entrée et le débit des transactions ajoutées en sortie. La colonne *Sécurité* correspond au nombre de malveillants moyen que la chaîne a pu tolérer pendant l'expérience et la colonne $tx_{confiance}$ la probabilité minimale moyenne qu'un des nœuds de la chaîne ait sa copie de blockchain à jour. Ces deux dernières valeurs sont calculées à partir du nombre de validateurs moyen durant l'expérience.

La figure 4.11a montre l'exemple de demande (de débit) soumis à la chaîne. Cette demande est conçue pour pouvoir être accomplie par un nombre de validateurs compris entre le nombre minimum et le nombre maximum possible. À la fin de chaque expérience, la demande est fixée à zéro afin de mettre en évidence la présence de transactions empilées dans la chaîne et de laisser à la chaîne le temps de commettre ces transactions. Les variations de la demande restent proches de la capacité du nombre maximal de validateurs puisqu'elle correspond à la zone la plus sensible au nombre de validateurs (voir figure 4.9b et 4.9c).

Plateforme	**Configuration**	**Écart Débit**	**Sécurité**	$tx_{confiance}$
PI	Min	$0,58\%$	1	$5,3\%$
	Max	$39,5\%$	16	$66,7\%$
	Offline	$24,1\%$	7	$29,3\%$
	Sabine	$7,79\%$	7	$28,4\%$
G5K	Min	$1,29\%$	1	$1,3\%$
	Max	$67,6\%$	66	$66,7\%$
	Offline	$25,6\%$	20	$20,3\%$
	Sabine	$4,45\%$	21	$21,4\%$

TABLEAU 4.1 – Résultats pour les mesures d'écart relatif entre la demande et le résultat (*Écart Débit*), le nombre de malveillant moyen tolérable et le taux de confiance moyen sur les deux plateformes en fonction de quatre configurations différentes

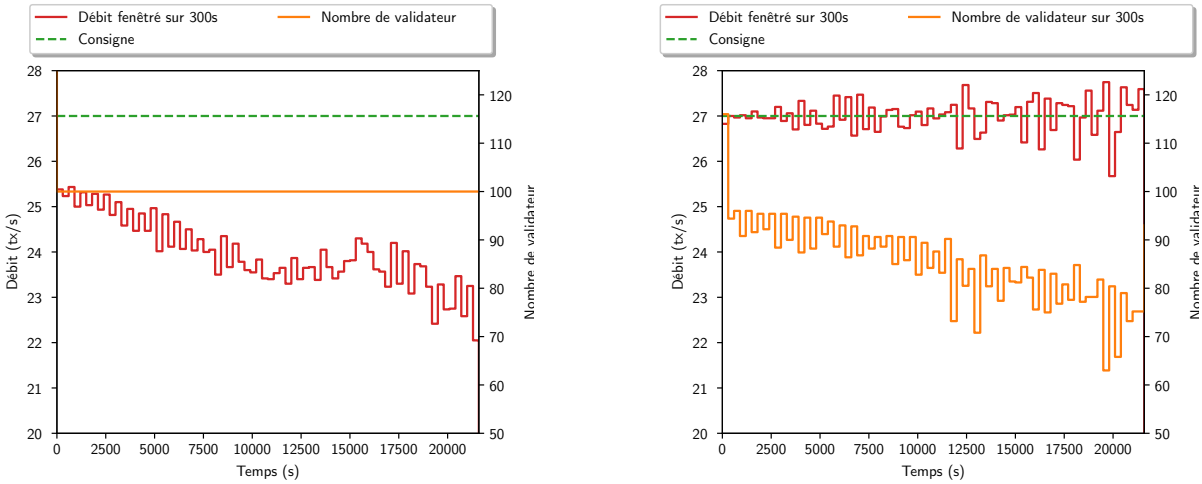
Les résultats des expériences montrent que si l'on prend trop peu de validateurs (4 validateurs, le minimum du protocole PBFT), on obtient une chaîne qui peut atteindre un débit élevé avec une faible erreur relative entre le débit demandé et le débit ajouté ($0,58\%$ estimé sur une fenêtre de $60s$ avec 4 validateurs sur la plateforme de PI sur la figure 4.11b et $1,29\%$ sur la Grid5000), mais, en raison des propriétés de la BFT, la chaîne ne tolère pas beaucoup de validateurs malveillants (seulement 1 dans les deux exemples) et que la probabilité qu'un nœud aléatoirement choisi soit à jour est faible.

À l'inverse, en fixant le nombre de validateurs au maximum (50 nœuds pour la plateforme de Raspberry Pi et 200 nœuds pour la Grid5000), la tolérance aux validateurs malveillants augmente fortement (16 et 66 respectivement) et une cohérence stricte est assurée pour tous les nœuds, impliquant qu'au moins deux tiers des nœuds soient cohérents. Cependant, les performances en ce qui concerne le débit chutent sensiblement, la demande ne sera pas satisfaite immédiatement, ce qui implique une latence pour satisfaire la demande. L'erreur relative est élevée ($39,5\%$ correspondant à un écart moyen de $19,1$ tx/s de différence en moyenne pour la plateforme de Pi et $67,6\%$ soit $26,6$ tx/s de différence moyenne pour la Grid5000) avant la fin de la demande (à $t = 3\,000\ s$).

Enfin, la connaissance du réseau permet de calculer le nombre de validateurs de manière hors ligne pour satisfaire la demande moyenne, mais elle nécessite la connaissance de la demande future et ne s'adapte pas à la variation du débit demandé, ce qui implique une erreur importante entre l'entrée et la sortie. Dans nos exemples, la demande 4.11a peut être atteinte avec 22 de validateurs pour la plateforme de Pi et 61 validateurs pour la Grid5000, mais l'erreur relative d'une chaîne avec ces nombres de validateurs reste élevée puisqu'elle est égale à $24,1\%$ et $25,6\%$ respectivement.

Impact du temps de l'expérience

En étudiant notre implémentation du consensus PBFT sur des expériences relativement longues, nous nous sommes aperçus que les performances diminuaient au cours du temps. Ainsi, sur une expérience de 6 heures avec un débit d'entré constant représenté en figure 4.5a, le débit de sorti a diminué d'environ 3 tx/s. Cette perte peut paraître minime, mais si elle persiste, les performances de la blockchain s'effondrent au bout de quelques jours. Or, la blockchain est supposée être une solution pérenne.



(a) Sans boucle de régulation

(b) Avec boucle de régulation

FIGURE 4.5 – Expérience menée pendant 6 heures sur le Grid5000 avec 200 nœuds, sans délai avec un débit fixe

Afin d'identifier l'origine de cette baisse de performance, le délai de chaque étape du consensus a été mesuré sur une chaîne de 20 nœuds soumise à une demande fixe supérieure à sa capacité pendant 4000 secondes. Ces délais sont représentés sur la figure 4.6.

Les étapes *Pre-Prepared*, *Final Committed* et *New Round Proposer* sont des étapes de transitions constituées d'un ensemble d'opérations presque fixe réalisé par le nœud. La mesure du délai nécessaire pour exécuter ces étapes, respectivement sur les figures 4.6a 4.6c et 4.6e, montre ainsi des délais faibles et constants. Les étapes *Prepared* et *Committed* attentent 2/3 de N messages dans leurs étapes et l'étape *New Round* attend un message du Proposer. Les étapes *Prepared* et *Committed* ont un profil, sur les figures 4.6b et 4.6d, relativement similaire. Le délai est plus élevé que les étapes précédentes, ce qui s'explique par la nécessité de se synchroniser avec les autres nœuds. L'étape *New Round* cependant est différente sur la figure 4.6f. Si une grande majorité des délais est relativement constante, une minorité est majorée par un délai proportionnel à la durée de l'expérience.

Ainsi, le temps moyen de chaque étape reste stable, sauf dans le cas de l'étape *New Round* qui croît linéairement. Cette étape semble responsable de l'augmentation d'une durée de création d'un bloc plus importante, ce qui induit une baisse du débit que l'on peut retrouver sur la figure 4.5a. Cette constatation rejoint les travaux de Kolasa et al. (Kolasa et al., 2020) qui montrait que l'étape la plus importante et venant imposer son rythme au temps de création d'un bloc était l'attente du message *Pre-Prepare* envoyé par le Proposer, correspondant ici à l'étape *New Round*.

Notre hypothèse est que dans le cadre d'un débit soumis supérieur à la capacité, les nœuds doivent gérer énormément de messages, favorisant l'apparition de nœuds avec des états en retard par rapport à la majorité. Cependant, comme le consensus de sélection dans notre implémentation est aléatoire et le Proposer change à chaque bloc, il arrive que les nœuds les plus en retard soient élu Proposer. Afin de jouer leurs rôles, ils doivent rattraper leurs retards dans la lecture des messages et l'exécution du protocole PBFT. Ce temps de rattrapage correspond à la majoration du délai de l'étape *New-Round* présent dans la figure 4.6f (le nœud attend alors le message *pre-prepare* du Proposer en retard). Ce retard serait donc lié directement au consensus utilisé dans

notre blockchain.

4.2.2 Commande en boucle fermée

Toutes les solutions blockchain présentées précédemment reposent sur l'idée qu'un petit groupe de validateurs est plus rapide qu'un grand groupe, mais l'état actuel des choses ne met pas en évidence une blockchain avec une taille variable du groupe de validateurs. Comme nous l'avons expliqué précédemment, les performances d'une blockchain BFT sont intrinsèquement liées à la taille du groupe de validateurs. À notre connaissance, aucune étude n'a jamais proposé un protocole de consensus de blockchain permettant l'autoadaptation de la taille du pool dans le temps afin de répondre au débit d'entrée.

Comme il a déjà été mentionné dans la section précédente, plus le nombre de validateurs est élevé, plus la chaîne est sécurisée (cf. équation (4.2)) et cohérente (cf. eq (4.6)). Cependant, cela a une conséquence directe sur le débit de sortie de la chaîne, qui diminue de manière quadratique avec le nombre de validateurs (cf. eq (4.5)), ce qui conduit à un problème de compromis entre la sécurité et le débit. Les blockchains répondent aujourd'hui à ce compromis en imposant un nombre de validateurs fixe, par exemple 125 validateurs pour la blockchain Tendermint (Buchman et al., 2018). Comme montré précédemment, l'estimation de ce nombre de validateurs idéal nécessite la connaissance de la future demande, mais n'est pas idéale pour autant. C'est parce qu'il s'agit d'une réponse statique à un problème dynamique.

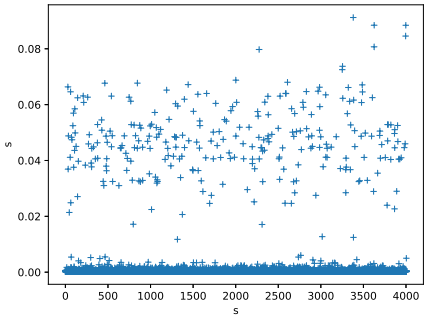
Afin de répondre dynamiquement à ce compromis, le framework Sabine (*Self-Adaptive BlockchaIn coNsEnsus*) est proposée et exploite une boucle de régulation pour résoudre ce compromis en sécurité et performance. Ce problème de compromis peut être exprimé sous la forme d'une contrainte à résoudre, comme formalisé dans (4.7) selon l'énoncé de la table 4.2.

Var.	Description
n	Le nombre de valideurs dans la chaîne
N_{limit}	Nombre minimum de validateurs
$\mathcal{N}$	Le nombre de noeuds
d_{req}	Débit des transactions demandée
d_{commit}	Débit des transactions ajoutés
lag	Délai global (incluant le délai du réseau, le délai du matériel...)
M	Modèle obtenu par ML, donnant le débit maximal de la transaction engagée en fonction du nombre de validateurs et du délai.
$BlockSize$	Nombre maximal de transactions par bloc
b	Le débit du bloc engagé
$\mu(d, n_{val})$	Retourne la latence maximale telle que le modèle M restreint au nombre de validateurs n offre une capacité égale à d
$\varphi(d, lag)$	Retourne le nombre maximal de validateurs tel que le modèle M restreint au temps de latence lag fournisse une capacité égale à d

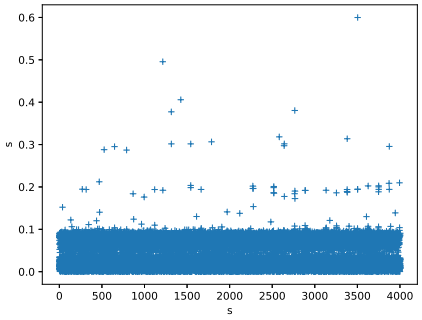
TABLEAU 4.2 – Notations

$$\underset{\mathcal{N}}{\operatorname{argmax}} \left(n^t \mid d_{commit}(n^t, lag^t) = d^t_{req}\right) \; s.t.\; n > N_{limit} \tag{4.7}$$

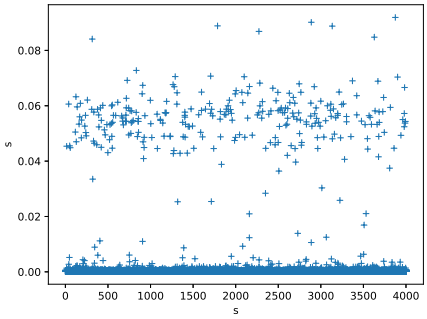
Sabine ajuste le nombre de validateurs de la chaîne n^t en temps réel t afin que la *Capacité*, c'est-à-dire le débit maximal que la chaîne peut atteindre, soit proche du débit demandé récent,



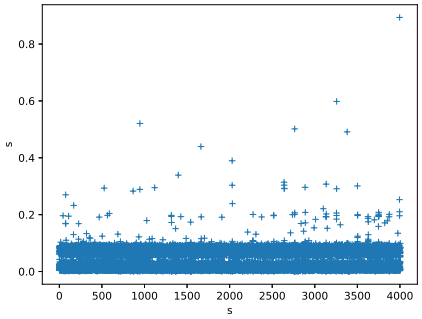
(a) Étape *Pre-Prepared*



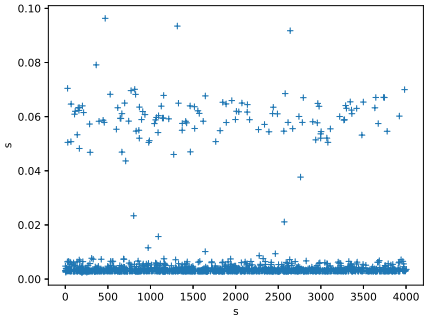
(b) Étape *Prepared*



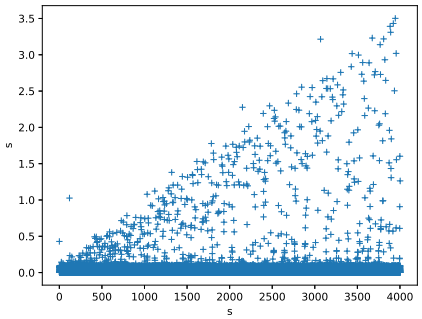
(c) Étape *Final Committed*



(d) Étape *Committed*



(e) Étape *New Round Proposer*



(f) Étape *New Round*

FIGURE 4.6 – Temps nécessaire pour accomplir chaque étape pour un nœud dans une chaîne soumise à une demande fixe supérieure à sa capacité pendant 4000 secondes

ce qui implique que le débit ajouté d_{commit} soit égal au débit demandé d_{req}, quel que soit le retard *lag* présent dans le réseau. Afin d'éviter une baisse drastique du nombre de validateurs en cas de forte demande, une limite de sécurité N_{limit} ajustée par les administrateurs est ajoutée et vient empêcher que le nombre de validateurs soit inférieur à cette limite et en définitive qu'une exigence de sécurité soit satisfaite.

De plus, des ralentissements peuvent apparaître au sein du réseau à cause de la variation de la bande passante ou à cause des variations des performances du matériel (concurrence au sein du processeur ...). Par ailleurs, comme expliqué en section 4.2.1, le consensus peut favoriser l'émergence d'un retard diminuant les performances. En conséquence, Sabine prendra en compte ce retard pour adapter le nombre de validateurs en compensant la baisse de performances.

4.3 Fonctionnement de Sabine

En raison de la spécificité de chaque réseau où Sabine peut être appliquée, il n'est pas possible d'identifier un modèle mathématique liant la capacité, les délais et le nombre de validateurs. En conséquence, Sabine utilise l'apprentissage automatique pour relier ces valeurs et contrôler le nombre de validateurs. Les principes et l'exécution de Sabine suivent les étapes présentées dans la figure 4.7. Trois étapes principales émergent.

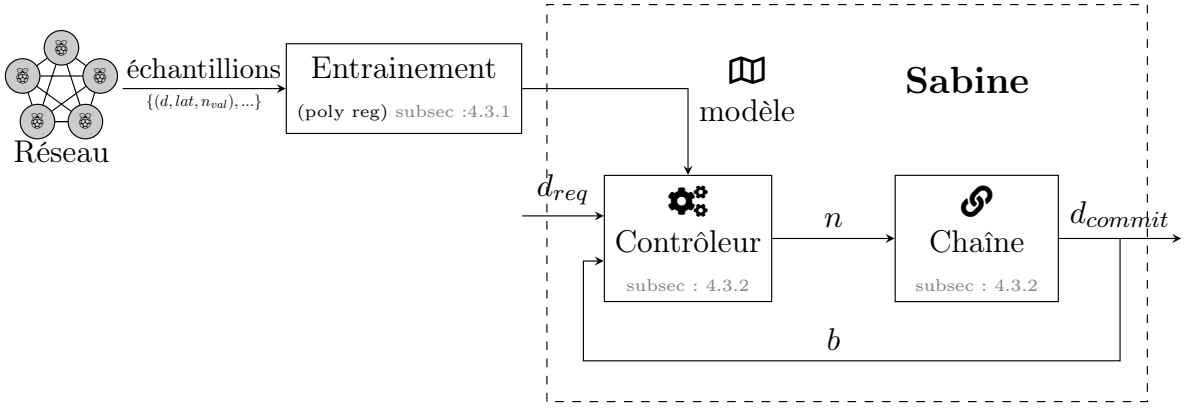


FIGURE 4.7 – Phases de Sabine

Tout d'abord, une phase d'entraînement, détaillée dans la section 4.3.1, est nécessaire pour construire un modèle spécifique au réseau, reliant la capacité, le délai et le nombre de validateurs. Ce modèle est détaillé dans la section 4.3.2 et est utilisé par le contrôleur de Sabine pour estimer le nombre idéal de validateurs à considérer dans la chaîne par le mécanisme détaillé dans la section 4.3.2.

4.3.1 Phase d'entraînement

Sabine vise à adapter la configuration de la chaîne pour maximiser le débit en fonction d'un modèle estimé lors d'une phase d'apprentissage. Ainsi, la première étape de Sabine est de collecter des échantillons du débit maximal du réseau pour construire le modèle. Cette valeur particulière du débit est appelée *Capacité* et est directement liée au temps nécessaire pour construire et ajouter un bloc à la chaîne. Pour mesurer cette capacité, la chaîne est déployée sur le réseau et l'on mesure le débit du nombre de transactions ajouté à la chaîne en réponse à un débit

demandé fixe de transaction. Ce débit demandé est déterminé dans une analyse primaire pour être légèrement supérieur au débit de réponse. En effet, comme a pu le montrer Han et al. (Han et al., 2020), une trop forte demande soumise à une chaîne diminue les performances de cette chaîne, car celle-ci perd des ressources pour gérer la gestion de ces soumissions.

On fait varier deux paramètres : le nombre de validateurs de la blockchain et le délai (réseau). En supposant qu'il n'y a pas d'autre trafic dans le réseau, le délai représente la variation de temps due à la variation de la bande passante, des performances du matériel et du retard induit par le consensus. Il est simulé dans notre chaîne par les nœuds avec une attente avant d'envoyer un message aux autres nœuds de la chaîne. Ce retard peut être considéré comme la cause fondamentale de la dérive du système par rapport au modèle initial estimé. L'estimation de cette dérive est donc nécessaire pour adapter le système en cas de perturbations.

Par conséquent, chaque échantillon consiste en un triplet $(capa, lag, n)$. Si un ensemble de triplets est suffisamment grand, il est utilisé comme un ensemble de données d'apprentissage pour construire un modèle qui associe le nombre de validateurs, le retard et la capacité de la chaîne en utilisant l'apprentissage automatique. Cet apprentissage est basé sur une régression polynomiale détaillée dans 4.8. Son fonctionnement est le suivant : une première fonction définie empiriquement (la fonction *invert*) est d'abord appliquée aux données, puis une transformation polynomiale est appliquée et enfin, une régression ridge (régression linéaire avec une contrainte quadratique) est réalisée. Les hyperparamètres sont obtenus par *hyperparameter tuning*.

```
def invert(x, deca=100):
    return 1 / (deca + x)


polynomial_regression = make_pipeline(
    FunctionTransformer(invert),
    PolynomialFeatures(degree=14),
    StandardScaler(),
    RidgeCV(
        alphas=[0.001, 0.01, 0.1, 1, 10,
                100, 1000]
    )
)
```

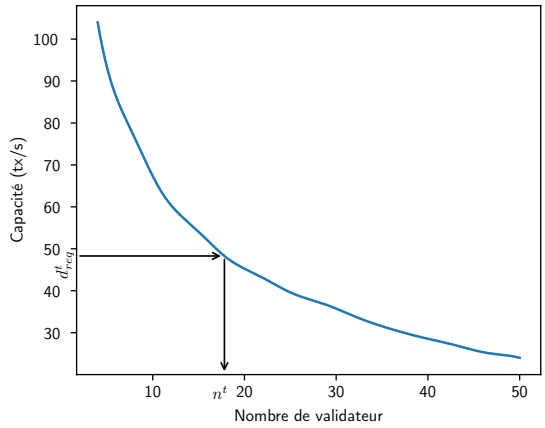
FIGURE 4.8 – Paramètres du modèle avec Sklearn

Le modèle étant dépendant de la plateforme, chaque plateforme possède son modèle associé. Ainsi le modèle de la plateforme de Pi est représenté sur la figure 4.9b et celui de l'expérience de la Grid5000 est représenté sur la figure 4.9c.

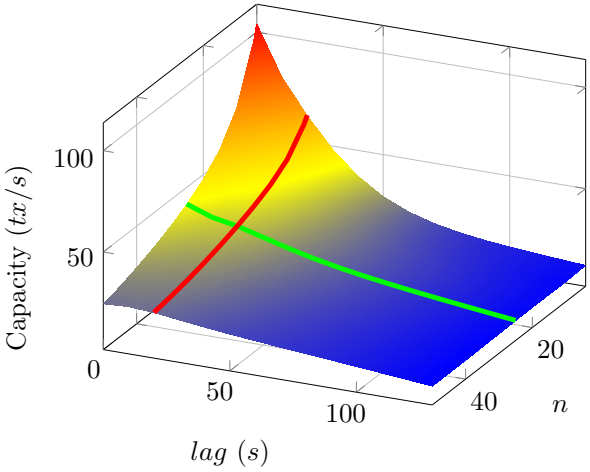
Le modèle obtenu M par *Machine Learning* fournit un débit principalement décroissant $M(n, lag)$ en fonction du nombre de nœuds n et du retard lag, et un traitement post-filtre est appliqué pour affirmer la diminution. Dans ce post-filtre, pour un couple (n, lag), si une Capacité supérieure existe pour un n ou un δ plus grand, alors la Capacité à (n, lag) prend cette valeur.

$$\frac{\partial M}{\partial lag}(n, lag) \leq 0 \tag{4.8}$$

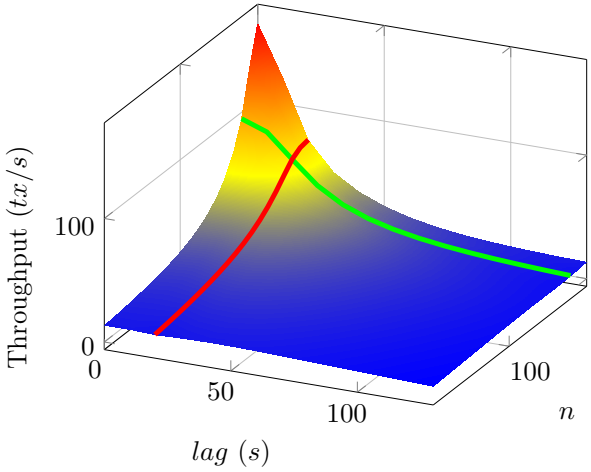
$$\frac{\partial M}{\partial n}(n, lag) \leq 0 \tag{4.9}$$



(a) Modèle sur le réseau de PI, sans prise en compte des délais



(b) Modèle sur le réseau de PI



(c) Modèle sur le Grid5000

FIGURE 4.9 – Modèles estimés par *Machine Learning*

4.3.2 Phase opératoire

Contrôleur

Sur la base de ce modèle, l'algorithme de Sabine résout la contrainte 4.7. L'idée est de maximiser la sécurité en augmentant le nombre de validateurs tant que le débit est suffisant, c'est-à-dire que le débit de la transaction engagée est égal au débit de la transaction demandée, sous la condition d'un niveau de sécurité minimum, équivalent à un nombre minimum de validateurs. Sabine résout cette contrainte en mesurant le débit de la transaction engagée et le débit de la transaction demandée, et en comparant ces variables avec le modèle précédent.

Sans délai Pour une compréhension plus facile, nous allons ignorer dans un premier temps le délai.

Dans ce cas, le modèle est simplifié et est similaire à celui représenté dans la figure 4.9a. Il s'agit d'un modèle correspondant à la plateforme de Pi. Comme on peut s'y attendre, la capacité

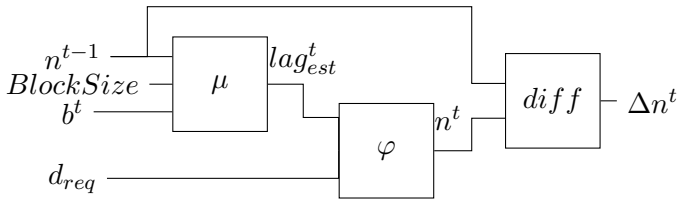


FIGURE 4.10 – Calcul de Sabine expliqué avec un diagramme, chaque bloc représente une fonction détaillée dans la section 4.3.2

diminue avec le nombre de nœuds. Plus précisément, le modèle forme une bijection décroissante entre la capacité et le nombre de validateurs. La fonction inverse σ détaillée dans 4.10 peut être définie.

$$\sigma(d) = n | M(n, lag = 0) = d \tag{4.10}$$

L'estimation du nombre de validateurs est simple. En effet, l'utilisation de la blockchain est optimale lorsque le débit est égal à la capacité, puisque tous les blocs sont remplis au maximum. Donc le nombre de validateurs est réglé de telle manière que la réponse de la chaîne soit sa capacité et égale au débit en requête d^t_{req}. En utilisant la fonction σ, le nombre de validateurs n^t est estimé par l'équation 4.11. Ce cheminement est représenté par les flèches sur la figure 4.9a.

$$n^t = \sigma(d^t_{req}) \tag{4.11}$$

Avec délai Dans un environnement réaliste, un retard peut apparaître en raison de la latence du réseau ou de la variation des performances du matériel et réduire les performances de la chaîne. Dans ce cas, le modèle se complexifie. Grâce aux propriétés (4.8) et (4.9), les fonctions inverses (4.12) et (4.9) peuvent être définies :

$$\mu(d, n) = max(\{lag | M(n, lag) = d\}) \tag{4.12}$$

$$\varphi(d, lag) = max(\{n | M(n, lag) = d\}) \tag{4.13}$$

Afin d'estimer un nombre idéal de validateurs dans cet environnement, Sabine estime le délai en se basant sur le délai simulé du modèle obtenu par *Machine Learning*. Ce calcul suit le schéma de la figure 4.10.

Dans une première étape, le retard est supposé responsable de la perte de performance de la chaîne. Connaissant le nombre réel de validateurs n^{t-1}, le retard lag_{est} est estimé en prenant le retard simulé qui rejoint la capacité réelle de la chaîne (ligne verte dans la figure 4.9b et 4.9c). Le débit d'ajout de transaction n'est pas une bonne métrique pour estimer la capacité réelle, car, en fonction du débit soumis, tous les blocs peuvent ne pas être remplis de transactions, ce qui est le cas lors d'une baisse de débit soumis. Ce que l'on cherche ici à mesurer n'est pas l'aptitude de la chaîne à ajouter des transactions, mais son aptitude à réaliser le consensus avec un nombre de nœuds précis. Par conséquent, le débit d'ajout de bloc est plus pertinent, en supposant que le débit de transaction demandé est suffisamment important pour produire des blocs continus, ce qui se produit si le débit demandé est supérieur à la capacité divisée par la taille du bloc. Ainsi, la capacité est estimée en multipliant le débit d'engagement de bloc b avec le nombre maximum de transactions dans un bloc $BlockSize$.

$$lag^t_{est} = \mu(b^t \cdot BlockSize, n^{t-1}) \tag{4.14}$$

Ensuite, le modèle est restreint au délai estimé lag_{est}, formant une bijection décroissante entre le débit maximal et le nombre de validateurs (ligne rouge dans la figure 4.9b et 4.9c). On se retrouve alors dans un exemple proche de la situation sans délai. Le nombre idéal de validateurs est obtenu en comparant le débit de la transaction demandée d_{req} à cette bijection. En cas d'incertitude, le nombre inférieur de validateurs est choisi. Avec cette configuration, la chaîne est capable d'ajouter des transactions avec un débit égal au débit demandé, répondant à la contrainte 4.7, sous la condition que le nombre estimé de validateurs soit supérieur à une limite de sécurité détaillée par les administrateurs.

$$n^t = \varphi(d^t_{req}, lag^t_{est}) \tag{4.15}$$

Action sur la chaîne

Sabine est lancée dans chaque nœud d'une chaîne pour que le système reste décentralisé. Après la fin d'un minuteur sans changement de validateur, le Proposer de la blockchain applique l'algorithme Sabine pour déterminer le nouveau nombre idéal de validateurs, ce qui signifie que Sabine est appelée périodiquement par un nœud du réseau. La période d'échantillonnage est fixée dans nos conditions expérimentales à 60s. Une fois ce nombre idéal de validateurs déterminé, une transaction spéciale est insérée en priorité dans la chaîne pour transmettre à tous les nœuds la mise à jour du nombre de validateurs et les modifications sont appliquées après l'engagement de la transaction. L'utilisation d'une transaction dans une chaîne BFT pour transmettre la mise à jour du validateur assure la cohérence de la modification pour les blocs suivants.

Un Proposer malveillant pourra donner un ordre pour diminuer fortement le nombre de validateurs, affaiblissant le consensus. Pour éviter ce cas, lors de la phase *Prepared*, les validateurs exécutent chacun leurs instances de Sabine et continuent le consensus si la proposition est en accord avec le résultat.

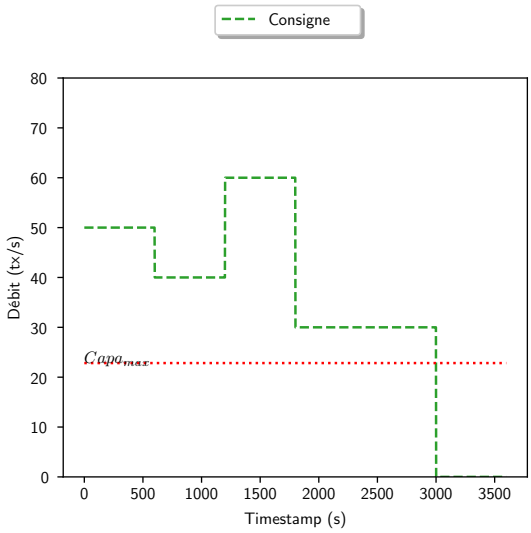
Le mécanisme de sélection des nouveaux validateurs dépend de la chaîne et tous les mécanismes listés dans le tableau 2.3 peuvent être adaptés. Dans notre exemple de chaîne, aucun mécanisme particulier n'est néanmoins implémenté, les nœuds sont ordonnés sur une liste et un pivot sépare les validateurs des non-validateurs.

4.4 Experimentation

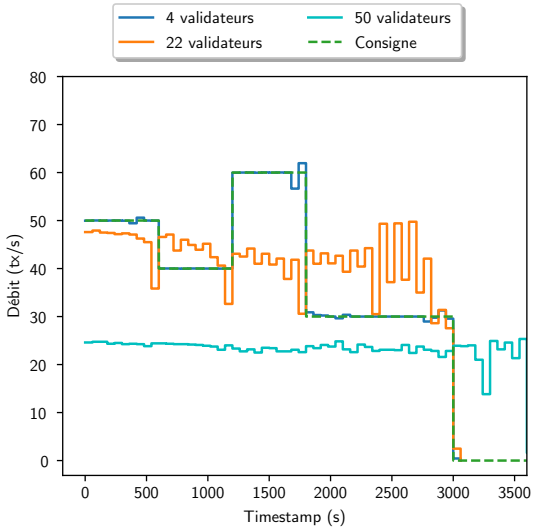
Sabine a été implémentée et adaptée à la chaîne PBFT décrite en section 4.1.1 et déployée sur deux plateformes différentes décrites en section 4.1.2 : une plateforme de Pi et le Grid5000.

4.4.1 Modèle

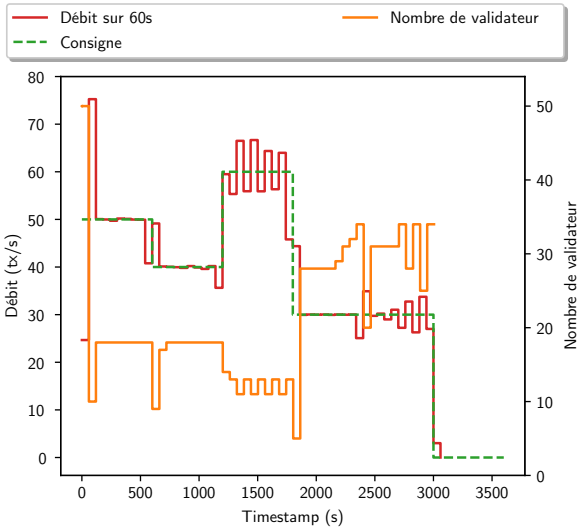
La figure 4.9b représente le modèle obtenu par *Machine Learning* sur le banc de 10 Raspberry Pi, avec une erreur absolue moyenne de $0,465$tx/s. La figure 4.9c représente le modèle obtenu par *Machine Learning* sur la chaîne évoluant sur la Grid5000, avec une erreur absolue moyenne de $0,961$tx/s. Comme prévu, dans les deux cas, la *Capacité*, c'est-à-dire le débit maximal engagé, diminue avec le nombre de validateurs et le délai simulé. Cette étude montre que le contrôle par l'algorithme de Sabine est intéressant si le débit souhaité est compris entre les capacités aux limites du nombre de validateurs. S'il est inférieur, le débit est engagé sans besoin d'adaptation, supérieur, le débit engagé est limité par la capacité au minimum de validateur (4 dans notre étude, le minimum des consensus BFT).



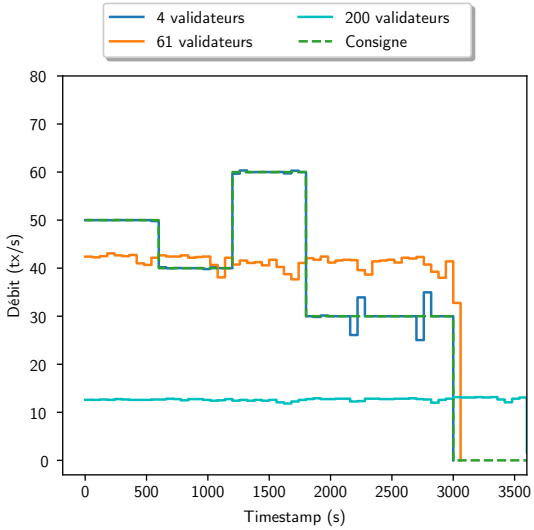
(a) Demande utilisée pour évaluer les chaînes



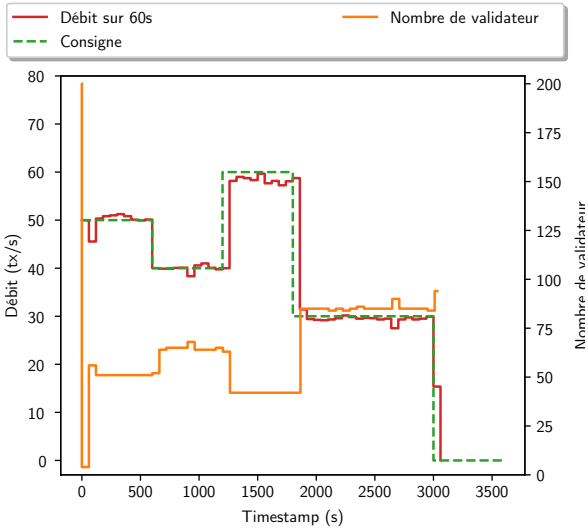
(b) Réponses à la demande 4.11a avec un nombre fixe de validateurs (PI)



(c) Débit ajouté et nombre de validateurs de Sabine à la suite de la demande 4.11a (PI)

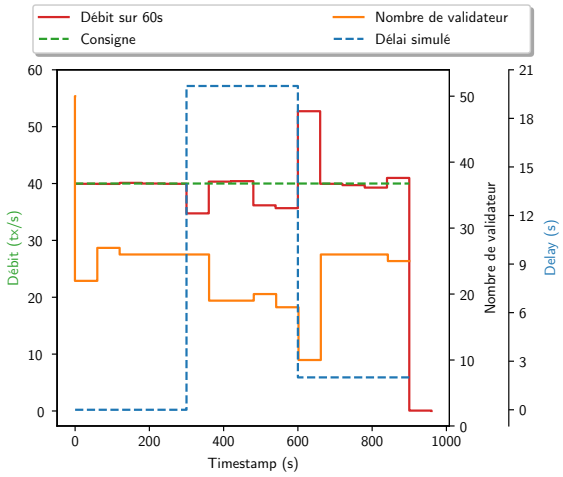


(d) Réponses à la demande 4.11a avec un nombre fixe de validateurs (G5K)

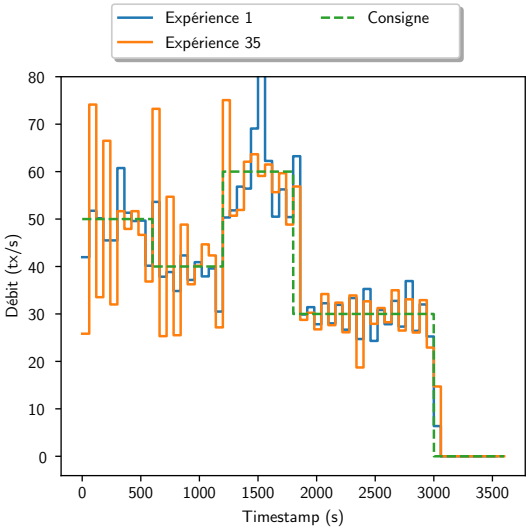


(e) Débit ajouté et nombre de validateurs de Sabine à la suite de la demande 4.11a (G5K)

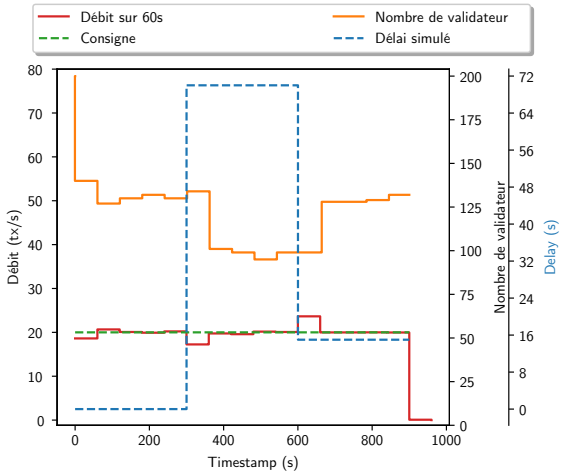
FIGURE 4.11 – Comparaison du débit ajouté pour une demande commune pour les chaînes avec et sans Sabine (PI et G5K)



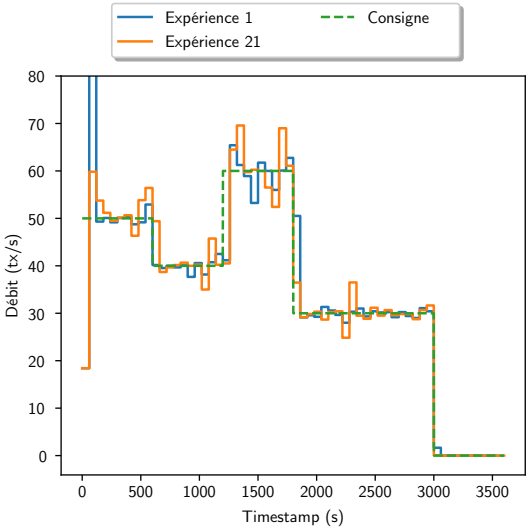
(a) Réponse de la chaîne avec Sabine à une variation de délai avec un débit demandé fixe (PI)



(b) 2 chaînes avec Sabine, avec le même modèle et la même demande, mais avec un retard aléatoire et indépendant (PI)



(c) Réponse de la chaîne avec Sabine à une variation de délai avec un débit demandé fixe (G5K)



(d) 2 chaînes avec Sabine, avec le même modèle et la même demande, mais avec un retard aléatoire et indépendant (G5K)

FIGURE 4.12 – Réponses de Sabine aux variations de délai

4.4.2 Consigne dynamique

Avec ces modèles, les chaînes peuvent s'adapter à une demande comme celle utilisée précédemment et représentée dans la figure 4.11a, avec la réponse représentée dans la figure 4.11c pour les Pi et la figure 4.11e pour le Grid5000. Ici, le nombre de validateurs est adapté en ligne au débit de la transaction entrante. Le temps nécessaire pour estimer un nombre idéal de nœuds est égal à $60s$, ce qui correspond à une occurrence du contrôle de Sabine.

En termes de résultats, le débit de transaction engagé est d'environ $7,79\%$ par rapport au débit demandé (correspondant à une erreur absolue de $1,71$ transactions par seconde) et avec un nombre moyen de $21,3$ validateurs.

Les résultats sont ajoutés au tableau 4.1 et montrent que la sortie de la chaîne est proche de l'entrée. Ainsi, dans le cas de la plateforme de Pi, la différence moyenne entre le débit de transaction proposée et celui ajouté est de $1,71$ transactions, correspondant à une erreur relative absolue de $7,79\%$ avec un nombre moyen de validateurs de $21,3$, lui permettant donc de tolérer en moyenne 7 malveillants. Quant à la chaîne sur le Grid5000, le débit de transaction ajouté varie d'environ $7,79\%$ du débit demandé, ce qui correspondant à une erreur absolue de $1,71$ transactions. Le nombre moyen de validateurs est de $21,3$.

Les résultats montrent que Sabine est capable d'adapter la chaîne en fonction d'une demande variable en modifiant le nombre de validateurs. Les performances des chaînes testées sont proches de performances demandées. Il est à noter que le nombre de validateurs mesuré sur les chaînes correspond au nombre de validateurs moyens sur la durée de l'expérience. Du point de vue de la sécurité, il est plus intéressant de regarder le nombre de validateurs minimal, car il s'agit du moment où la sécurité de la chaîne est la plus faible. Cependant, ce nombre de validateurs minimal dépend de la demande et est minoré par le minimum de sécurité évoqué dans l'équation (4.7) et défini par les administrateurs.

4.4.3 Adaptation à un délai

La prise en compte d'un délai arbitraire permet d'adapter la chaîne en cas d'apparition d'un délai inconnu. Pour illustrer cette propriété, un retard variable est introduit durant l'exécution d'une chaîne. Cette chaîne est soumise à un débit de transactions proposées constant. Ainsi, s'il n'y avait pas de contrôle, l'augmentation du délai diminue le débit d'ajout des transactions, et la réduction du délai doit le ramener plus proche du débit proposé.

Les figures 4.12a et 4.12c montrent les réponses des chaînes avec Sabine pour un débit de demande de transaction fixe avec une variation du délai. L'erreur relative entre la demande et le débit engagé dans ce scénario est maintenant égale à $4,87\%$ en moyenne pour la plateforme de Pi et $3,27\%$ pour le Grid5000. Les expériences montrent que le contrôle adapte le pool de validateurs en diminuant leur nombre afin d'accélérer le débit engagé et de compenser le ralentissement dû à l'augmentation du délai et inversement, la taille du pool de validateur augmente lorsque le délai diminue, car le consensus dispose de davantage de marge pour pouvoir introduire de nouveaux validateurs.

Dans un second temps, l'expérience de 6 heures réalisée en section 4.2.1 a été réitérée en appliquant Sabine. Les résultats sont disponibles sur la figure 4.5b et montre que Sabine a compensé la diminution des performances due au consensus et diminuant au cours du temps le nombre de validateurs. Ainsi, le débit d'ajout de transaction est égal au débit soumis tout au long de l'expérience, et pour compenser la perte en 6 heures, le nombre de validateurs a diminué d'une dizaine de validateurs.

4.4.4 Consigne et délais variables

Nous avons vu que Sabine permet d'adapter le nombre de validateurs aux variations de demande et de délais indépendamment, le framework est capable de réaliser son adaptation quand ces deux paramètres évoluent. Ainsi, la même demande représentée sur la figure 4.11a a été soumise aux deux chaînes, mais cette fois, un délai aléatoire est introduit. Ce délai est de l'ordre de $15s$ et change tous les $150s$ en moyenne. Afin d'avoir des mesures plus précises, les expériences ont été répétées un grand nombre de fois, avec des délais différents entre chaque expérience. Deux de ces expériences pour chaque plateforme sont représentées dans la figure 4.12. Toutes les expériences ne sont pas représentées pour des raisons de visibilité.

Dans le cas de la plateforme de Raspberry Pi, les résultats sont représentés sur la figure 4.12b. 35 chaînes associées à Sabine ont été déployées et soumises à la demande de référence avec un retard aléatoire. Les résultats montrent que le débit d'ajout de transaction est proche de la demande et que les diminutions dues au retard sont rapidement compensées. En effet, l'erreur relative entre la demande et le débit d'ajout est égale en moyenne à $17,45\%$, avec un écart-type de $0,05\%$. Cette compensation du retard a été gérée par une réduction du nombre de validateurs à $18,2$ en moyenne, avec un écart-type de $3,0$ entre les expériences.

Dans le cas de la chaîne sur le Grid5000, les résultats sont représentés sur la figure 4.12d. 21 chaînes ont été déployées. L'erreur relative entre la demande et le débit d'ajout est de $7,21\%$ avec un écart-type de $0,71\%$. Le nombre de validateurs moyen a été amené à $70,9$, avec un écart-type de $6,52$.

4.5 Conclusion

Nous avons réalisé le premier contrôle sur le nombre de validateurs dans une chaîne BFT afin d'adapter la capacité de la chaîne à son besoin, en fonction du débit demandé tout en gardant une limite de sécurité. Dans ce chapitre, nous avons proposé un nouveau protocole Self-Adaptive BlockchaIn coNsEnsus (Sabine) et nous avons montré qu'une chaîne BFT utilisant Sabine peut augmenter sa sécurité pendant une période de faible demande en augmentant le nombre de validateurs. De plus, grâce à cette augmentation, la cohérence au sein du réseau est renforcée et la confiance envers les valeurs lues est améliorée. Des expérimentations réelles et des tests de performance montrent que Sabine a augmenté de manière significative la satisfaction des taux de débit de la demande tout en couvrant les problèmes de sécurité.

Les travaux futurs peuvent envisager d'inclure d'autres critères tels que l'efficacité énergétique. Le protocole BFT est économe en énergie, mais l'énergie dépensée dans le protocole augmente avec le nombre de nœuds impliqués dans le consensus. En prenant en compte l'énergie pour définir le modèle comme une nouvelle dimension, Sabine pourrait adapter le nombre de validateurs afin de respecter une limite énergétique.

Dans notre étude, Sabine a été utilisée sur une chaîne qui utilise un protocole PBFT classique, mais le contrôle peut être adapté à des protocoles BFT plus efficaces (pour lesquels les performances sont également limitées par le nombre de validateurs). Il est également envisageable d'exploiter Sabine sur des consensus à base de preuve en adaptant le framework. Par exemple, dans le cadre d'Etereum, Sabine peut être utilisée pour faire varier, non pas le nombre de mineurs, mais le paramètre l régissant la définition de la difficulté dans l'équation (3.11). Cependant, cette adaptation poserait des questions de sécurité sur la confiance d'un changement sur cette valeur, contrairement aux consensus BFT où le changement du nombre de validateurs fait suite à une décision commune.

Le problème de la sélection du nouveau validateur n'est pas traité jusqu'ici. La chaîne testée

utilisait un simple pivot pour séparer validateurs et non-validateurs. Mais Sabine peut être adaptée sur des chaînes qui résolvent le problème de la sélection du validateur comme les consensus hybrides basés sur BFT. Ces consensus résolvent le problème des validateurs à choisir, Sabine, combien. Des propositions efficaces et dynamiques pour sélectionner les validateurs sont donc explicitées dans le chapitre suivant.

5

Solutions pour la sélection des validateurs

Dans la description du framework Sabine, la sélection des validateurs a délibérément laissé libre. Des algorithmes comme ceux prenant place dans le cadre de consensus hybride basé sur BFT et présentés en section 2.2.3 peuvent être intégrés pour réaliser cette sélection. Pour rappel, ces différents algorithmes proposés utilisent un premier consensus basé sur une preuve afin de sélectionner les nouveaux validateurs de la chaîne. Plus précisément, ce premier consensus permet une sélection aléatoire des validateurs protégés des attaques Sybil grâce à la ressource associée au consensus, apportant de nombreux avantages dans la sécurité du consensus. Ainsi, il est plus difficile à un attaquant de contrôler la chaîne, puisqu'il ne peut connaître à l'avance les nœuds qu'il doit chercher à contrôler et qui seront sélectionnés pour sélectionner le consensus. De plus, une blockchain supposant une sélection aléatoire suivant une distribution uniforme s'assure d'une décentralisation optimale, puisqu'à terme, les validateurs sont sélectionnés parmi les nœuds dans la même proportion. Plus précisément, dans les consensus comme PoW ou PoS, cette proportion est pondérée avec la quantité de ressources nécessaire à la preuve possédée.

Ainsi, la sélection aléatoire est une force pour un consensus. Cependant, le revers de la médaille de cette sélection implique qu'il est impossible de réaliser une sélection plus complexe des validateurs. En effet, dans l'exemple du consensus PBFT, sélectionner des nœuds proches ou avec de meilleures performances réseau peut permettre d'accélérer les différentes étapes du consensus, impliquant un débit d'exécutions des transactions plus élevé. Pour cette raison, ce chapitre propose de réduire les objectifs de sécurités de la chaîne en abandonnant la sélection aléatoire des validateurs, dans le but d'exploiter une sélection plus astucieuse visant à améliorer les performances. Cette idée de sélection a déjà été introduite par d'autres groupes de recherche comme M. Liu et al. (M. Liu et al., 2019).

La première section de ce chapitre définit les problématiques auquel doit répondre une telle sélection. En effet, bien que la sécurité est réduite, des garde-fous doivent quand même être mis en place. Deux approches sont décrites dans les sections 5.2 et 5.3. La première utilise un système de réputation, souvent mis en place dans le cadre d'infrastructure blockchain, la seconde tente d'exploiter la proximité et de sélectionner les nœuds les plus proches. Parmi ces deux approches, un protocole sera défini et testé. Les résultats de ce protocole sont détaillés en section 5.4.3. La mise en place de cette sélection au sein de la blockchain et une méthode pour rendre cette sélection dynamique en fonction des changements du réseau sont développées en section 5.5.1.

5.1 Présentation des problématiques associées à la sélection

5.1.1 Sécurité

Comment explicité précédemment en introduction, la sélection aléatoire rend plus difficile pour un attaquant la prise de contrôle, puisqu'il doit s'assurer que les nœuds qu'il contrôle soient sélectionnés comme validateurs. Ainsi, pour un attaquant possédant f nœuds dans un réseau nécessitant v validateurs et composé de n nœuds, en supposant que $3f + 1 > v$ et $f \leq v$, la probabilité que cet attaquant viennent perturber le consensus avec tous ses nœuds est de $\frac{\binom{n}{v-f}}{\binom{n}{v}}$. Cette probabilité diminue donc fortement la taille du réseau, assurant donc qu'un réseau plus large est plus sécurisé.

Cependant, cette probabilité n'est pas nulle et il suffit d'une seule prise de possession du consensus pour que le malveillant puisse insérer des transactions corrompues dans la chaîne. Au final, la sécurité du consensus ne dépend que de la taille du pool de validateurs, question abordée dans le chapitre précédent.

5.1.2 Efficacité

Notre approche consiste donc à sélectionner les validateurs les plus efficaces. Les validateurs réalisent toujours un protocole BFT proche du protocole PBFT dans notre approche. La vitesse d'exécution de ce protocole dépend de nombreux paramètres matériels, comme la vitesse d'exécution du processeur qui impacte la construction des blocs ou la vitesse de vérification des signatures par exemple. Il a cependant été montré par Kolasa et al. (2020) que le facteur le plus déterminant dans ce protocole est le temps de transmission des blocs. La sélection des validateurs a donc tout intérêt à se réaliser parmi les nœuds ayant un meilleur débit réseau par exemple, ou ceux diminuant le délai de réponse avec leurs voisins.

Cette problématique peut être modélisée pour un graphe fini et pondéré avec les valeurs de débit ou de délai. Nous supposerons ce graphe complet, signifiant que chaque nœud est capable de communiquer avec un autre. La problématique énoncée revient donc à trouver une sous-clique de taille donnée, minimisant (pour le délai de réponse) ou maximisant (pour le débit) les poids associés au graphe. Dans nos évaluations, nous choisissons la mesure du délai de réponse comme métrique mesurant un sous-graphe efficace.

5.1.3 Dynamisme

Afin que tous les nœuds disposent d'une même connaissance du réseau, nous supposerons que les nœuds partagent une matrice d'adjacence résumant le graphe complet qui représente le réseau. Le réseau n'étant pas figé, les valeurs de la matrice d'adjacence sont amenées à évoluer. De plus, la sélection est amenée à cohabiter avec Sabine, impliquant une possible modification du nombre de validateurs au cours du temps. L'algorithme de sélection doit donc pouvoir s'adapter aux changements de performances du réseau et au changement du nombre de validateurs. Le partage et la modification au cours du temps de la matrice d'adjacence sont détaillés en section 5.5.1.

5.1.4 Décentralisation

Le risque d'un algorithme de sélection est de trouver un point fixe ou un motif périodique sur un sous-groupe impliquant que seul un sous-groupe de nœuds soit majoritairement ou exclusivement sélectionné comme validateurs. En effet, la présence de ce groupe implique une centralisation

des décisions sur ce sous-groupe, en formant un tiers de confiance. Typiquement, les solutions optimales aux problèmes de cliques sont fixes. La blockchain étant un système décentralisé, il est donc important de faire participer tous les nœuds et éviter la formation de ce tiers de confiance. Ainsi, une sélection idéale devrait donner comme taux de participation au consensus, c'est à dire le nombre de fois qu'un nœud a été sélectionné pour réaliser un consensus sur le nombre de consensus total, une valeur proche de tx_{part}^{opt} définit selon l'équation (5.1) où n est le nombre de validateurs et $\mathcal{N}$ le nombre de nœuds.

$$tx_{part}^{opt} = \frac{n}{\mathcal{N}} \tag{5.1}$$

Afin de mesurer ce degré de décentralisation, divers outils sont possibles. La solution la plus naturelle revient à déterminer l'écart-type de taux de participation de l'ensemble des nœuds. Ainsi, un écart-type élevé indiquera l'existence d'une grande différence de participation entre des nœuds du réseau. Cette valeur est à comparer avec l'écart-type du pire cas possible σ_p, celui obtenu quand le pool de validateur ne change jamais. Cet écart-type dépend du nombre de nœuds et du nombre de validateurs et est déterminé dans l'équation (5.2). La connaissance de cette valeur nous permet de déterminer l'écart par rapport à cette valeur avec l'écart-type relatif σ_r dans l'équation (5.3). L'objectif est donc de diminuer cet écart-type relatif.

$$\sigma_p(n, \mathcal{N}) = \frac{1}{2} \cdot \sqrt{1 - \left(2 \cdot \frac{n}{\mathcal{N}} - 1\right)^2} \tag{5.2}$$

$$\sigma_r(x, n, \mathcal{N}) = \frac{\sigma(x)}{\sigma_p(n, \mathcal{N})} \tag{5.3}$$

Ce degré de décentralisation a été également été étudié par M. Liu et al. (M. Liu et al., 2019) et a été associé à l'indice de Gini. L'indice de Gini, dont le calcul pour un vecteur x de taille n est explicité dans l'équation (5.4). Il s'agit d'un nombre variant de 0 à 1, où 0 signifie l'égalité parfaite et 1, qui ne peut être atteint, signifierait une inégalité parfaite. Ainsi, il faut également obtenir l'indice de Gini le plus bas possible pour améliorer la décentralisation.

$$G(x) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} |x_i - x_j|}{2n^2 \overline{x}} \tag{5.4}$$

Cet indice de Gini a tendance à diminuer naturellement avec l'augmentation du nombre de validateurs, puisque cela implique une plus grande distribution des décisions. L'indice de Gini est ainsi limité par une valeur G_p dépendant du nombre de validateurs n et du nombre de nœuds $\mathcal{N}$, qui est défini dans la relation (5.5).

$$G_p(n, \mathcal{N}) = 1 - \frac{n}{\mathcal{N}} \tag{5.5}$$

5.1.5 Efficience

L'outil de sélection est destiné à être déployé sur tous les nœuds du réseau de la blockchain, afin que ceux-ci exécutent également l'algorithme de sélection pour vérifier la conformité des décisions et garder le système décentralisé. Comme la blockchain doit pouvoir être déployée sur des objets IoT possédants peu de ressources et que l'exécution ne doit pas trop surcharger les processeurs et faire concurrence au consensus, la sélection des nouveaux validateurs doit être rapide et peu gourmande en ressource. En supposant que le réseau doit pouvoir être déployé à très grande échelle, le calcul doit être rapide et efficace, idéalement avec un temps d'exécution en $O(1)$ ou $O(n)$ avec n le nombre de nœuds.

5.2 Système par réputation

La première proposition exploite une utilisation classique de blockchain. Elle se base en effet sur un système par réputation (Bellini et al., 2020). Les systèmes distribués de gestion de la réputation (*DRMS*) sont couramment exploités dans les systèmes distribués, notamment les partages pair-à-pair et permettent d'évaluer de façon relativement fiable les ressources proposées par les différents membres d'un réseau décentralisé, sans avoir à mettre en place un tiers de confiance.

L'origine de notre approche provient de Zeng et al. (Zeng et al., 2022). Dans cette proposition, les auteurs utilisent une application de vote et de réputation mise en place à l'aide de *smart contracts* sur une blockchain pour sélectionner les meilleurs contrôleurs dans le cadre d'un réseau multicontrôleur SDN. L'idée principale de cette sélection est de définir une réputation globale à chacun des nœuds du réseau puis de choisir le sous-ensemble avec la meilleure réputation. Cette réputation globale se construit pour chaque nœud avec les réputations locales des autres nœuds, c'est-à-dire la réputation que donne chaque nœud à ces voisins. Pour prévenir les nœuds avec une mauvaise réputation de mentir sur leurs réputations locales en vue de se favoriser lors de la sélection, le calcul de la réputation globale intègre une pondération. Ainsi, la réputation globale pour un nœud i est égale à la moyenne des réputations locales des autres nœuds pour ce nœud i, pondérée par la réputation des autres nœuds de l'epoch précédente. Le détail des calculs est décrit par la suite.

Notre première suggestion consiste à exploiter le même système de calcul de réputation. Dans la proposition de Zeng et al., un contrôleur améliore sa réputation si les autres nœuds sont en accord avec ses décisions. Nous proposons donc d'adopter un concept semblable, mais en prenant en plus en compte le temps de réponse des nœuds. C'est-à-dire que la notation favorise les nœuds votant pour des transactions non corrompues et dont le temps de réponse vers les autres nœuds est faible. Pour ce dernier point, nous favorisons les nœuds dont les votes ont été décisifs pour arriver au seuil des $^2/_3$.

Ainsi, un sous-groupe de nœuds répondant le plus rapidement avec des décisions en majeure partie soutenues devrait émerger et être de préférence choisi par la sélection. Ce système a l'avantage de résister également au changement du nombre de validateurs et de la variation des délais de réponse.

5.2.1 Calcul d'une réputation locale

Dans notre approche, la réputation locale d'un nœud i pour un nœud j est définie selon l'équation (5.6).

$$R_{l_{ij}} = \frac{\alpha_{ij} + 1}{\alpha_{ij} + \beta_{ij} + 2} \tag{5.6}$$

Où α_{ij} est le nombre de fois où le nœud i a été en accord avec la décision d'ajout de nœud de j et a eu besoin du nœud j pour aller à l'étape suivante, et $\beta_{ij} = N - \alpha_{ij}$ (avec N le nombre d'élections où i et j sont intervenu ensemble), le nombre de fois où le nœud i n'a pas eu besoin du nœud j pour aller à l'étape suivante. Plus clairement, lors de la décision d'ajout d'un bloc, si i est d'accord pour ajouter ce bloc, qu'il reçoit un message *prepare* de la part de j donnant son accord favorable pour l'ajout de ce bloc et que ce message fait partie des $^2/_3$ de messages nécessaire pour passer à l'étape *commit*, alors i incrémente son terme α_{ij}, dans le cas contraire, et si j est validateurs, il incrémente le terme β_{ij}.

Pour s'assurer de l'exécution identique du calcul des réputations globales dans l'étape suivante, il est nécessaire que le partage des réputations locales soit cohérent. Il est également nécessaire de s'assurer que ces valeurs ne puissent pas être modifiées ou mal estampillées au bénéfice d'un attaquant. Pour cette raison, Zeng et al. propose de diffuser ces réputations locales à l'aide de *smart contract*.

Pour éviter d'envoyer un trop grand nombre de transactions du simple fait des mises à jour des réputations locales et donc de faire concurrence aux réputations courantes, nous proposons que les nœuds mettent à jour le *smart contract* dans le cas où ils deviennent *Proposer*. Ce rôle étant dans notre implémentation de blockchain choisi au hasard, si l'on suppose que l'on a statistiquement assez de changement de Proposer entre deux sélections pour que chaque validateur puisse être Proposer, alors on peut considérer que chaque réputation locale est mise à jour au moins une fois entre deux sélections.

5.2.2 Calcul de la réputation globale

La sélection des validateurs se fait selon une période fixe. La première étape de cette sélection est de recalculer les réputations globales de tout nœud i. Il s'agit de la moyenne des réputations locales données par les autres validateurs, pondérés par leurs réputations globales relatives du calcul précédent :

$$R_{g_i} = \sum_{k=1,\neq i}^{n} w_k R_{l_{ki}} \tag{5.7}$$

w_k est donc la réputation globale relative du nœud k. Cette valeur représente son influence dans le réseau dans le calcul précédent des réputations globales et se calcule selon l'équation (5.8).

$$w_k = \frac{R_{g_k}}{\sum_{j=1}^{n} R_{g_j}} \tag{5.8}$$

Par la suite, afin d'imposer un changement des nœuds et de permettre une mise à jour régulière des réputations locales et globales, la réputation globale est multipliée par un terme croissant avec le nombre de périodes de sélection, afin d'augmenter artificiellement la probabilité des nœuds non-validateurs à être sélectionné. La réputation globale finale est donc calculée selon l'équation (5.9).

$$R_{g_{it}} = R_{g_{i0}} e^{-\lambda_i t} = \left(\sum_{k=1,\neq i}^{n} w_k R_{l_{ki}} \right) e^{-\lambda_i t} \tag{5.9}$$

La variable t s'incrémente à chaque période de sélection si le nœud i n'est pas sélectionné comme validateurs. À l'inverse, si le nœud est sélectionné comme validateur, la variable t prend la valeur 0. La variable λ_i est une variable réelle. Dans le cadre de la proposition de Zeng et al., λ_i est proportionnel au nombre de nœuds. $\lambda = \mu * \mathcal{N}$. Plusieurs valeurs de cette variable μ ont été testées.

5.2.3 Validation

Simulation

La sélection par répartition peut être intégrée dans une blockchain et être testée sur une grande plateforme, cependant il est plus simple et plus rapide de simuler le réseau. En effet, une simple matrice d'adjacence permet à ce stade de fournir assez d'éléments pour réaliser l'exécution

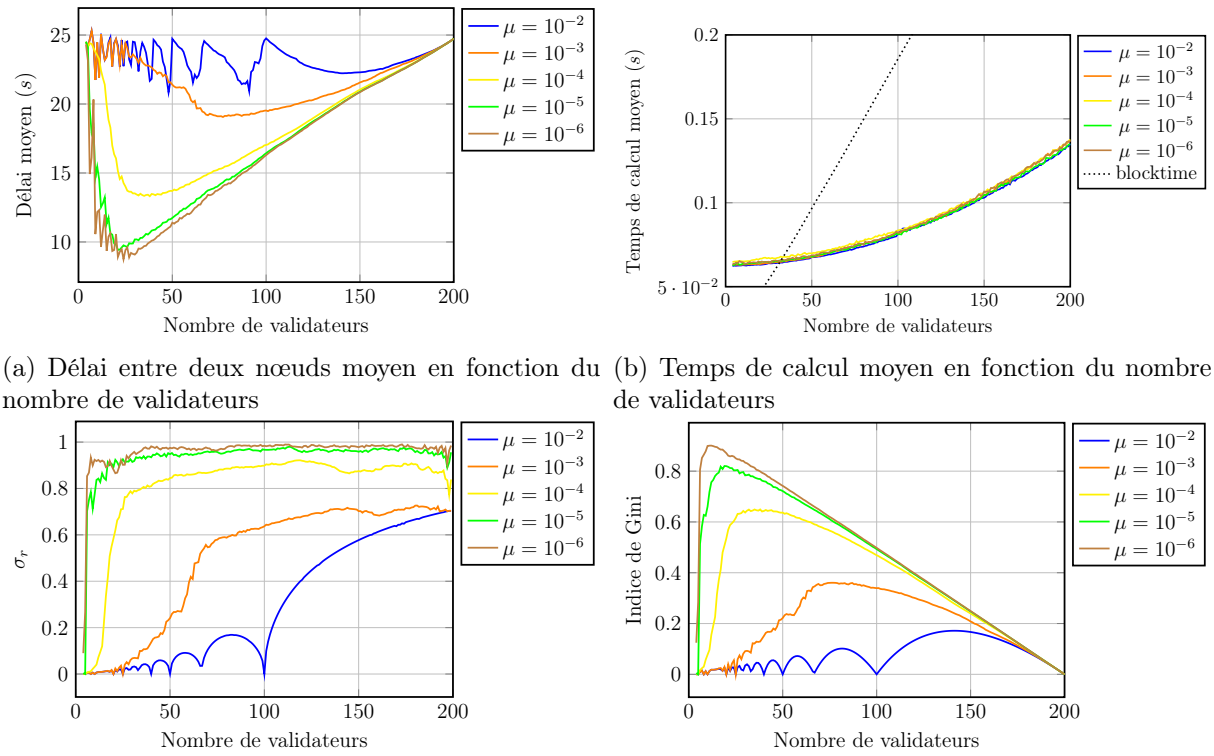
de l'algorithme. Pour nos simulations, la métrique utilisée pour remplir les valeurs de la matrice est le temps de transmission d'un message (RTT).

En effet, si le temps de réponse d'un nœud dans le cadre de la sélection par réputation permet de prendre en compte le temps de traitement des requêtes par le voisin, ce délai est approximable par le temps de transmission d'un message. Afin donc de simuler un réseau, 200 nœuds ont été projetés aléatoirement sur un plan et la distance entre les nœuds est assimilée au temps de transmission d'un message. Dès lors, pour chaque itération de la sélection, la simulation sélectionne pour chaque validateur les $^2/_3$ validateurs les plus proches et augmenter leurs réputations.

Les résultats de ces simulations sont disponibles dans la figure 5.1. Dans le cadre de ces simulations, une unique matrice d'adjacence est définie avec 200 nœuds dans un espace de taille 50×50. Le nombre de validateurs est fait varier du minimum (4) au maximum (200) et une simulation est lancé pour chaque nombre de validateurs. De plus, le paramètre μ étant défini arbitrairement, celui-ci prendra différentes valeurs au cours des simulations.

Lors d'une simulation, les nœuds sont initialement choisis arbitrairement, puis 50 itérations de l'algorithme de sélections sont effectuées pour arriver dans une situation proche du régime permanent. Ensuite, 400 itérations sont effectuées et des mesures sont effectuées à chaque itération.

Résultats



(a) Délai entre deux nœuds moyen en fonction du nombre de validateurs

(b) Temps de calcul moyen en fonction du nombre de validateurs

(c) Écart-type relatif du taux de participation en fonction du nombre de validateurs

(d) Indice de Gini sur la participation en fonction du nombre de validateurs

FIGURE 5.1 – Mesures moyennes diverses sur 400 itérations en fonction du nombre de validateurs sur une chaîne composé de 200 nœuds

Les résultats des simulations sont présentés sur la figure 5.1 à travers des métriques définies

grâce aux discussions des problématiques évoquées dans la section 5.1.

La figure 5.1a présente le délai moyen entre nœuds. Il s'agit de la principale mesure, l'objectif est ici de le minimiser. Les résultats montrent qu'en diminuant le paramètre μ, le délai moyen diminue jusqu'à s'écraser sur une pente croissante au-delà d'environ 40 validateurs, qui s'avère être le délai minimum d'après la section 5.3.2. Si le paramètre μ est supérieur à 10^{-3}, alors le temps moyen s'approche d'une sélection aléatoire (palier à 24s).

La figure 5.1b présente le temps de calcul moyen de la sélection. On constate que, pour un nombre de nœuds donné, le temps de calcul augmente avec le nombre de validateurs selon une courbe polynomiale ou exponentielle, et est indépendant du facteur μ. Des recherches plus poussées permettraient de déterminer si le nombre de nœuds impacte ce temps de calcul, ce qui influencerait le passage à l'échelle de la sélection. Dans tous les cas, le temps de calcul d'une itération est faible (< 150ms). Si l'on compare avec une estimation de la durée de minage d'un bloc dans notre chaîne sans latence sur le Grid5000, obtenue grâce au modèle obtenu en figure 4.9c ($blocktime = \frac{blocksize}{Capacit}$) et représenté en pointillé, et que l'on suppose que les performances de notre simulation sont identiques à ceux d'une implémentation, alors l'exécution de la sélection par réputation peut avoir un impact minime sur les performances de la chaîne, particulièrement dans le cas où le nombre de validateurs est inférieur à 50. Dans ce cas, le délai d'ajout d'un bloc ponctuel peut être doublé, ce qui ne devrait pas perturber les performances globales de la chaîne.

Les figures 5.1c et 5.1d mesurent respectivement l'écart-type relatif du taux de participation et l'indice de Gini de la participation. Cette première mesure la dispersion de la participation des nœuds par rapport au pire cas et la seconde la répartition de la participation. Ces deux mesures doivent être les plus basses possibles. Les résultats expérimentaux montrent que l'indice de Gini est très élevé et qu'il diminue avec l'augmentation du nombre de validateurs. Cependant, cette diminution est issue uniquement par l'augmentation du nombre de validateurs, l'indice de Gini tend vers la limite maximale G_p décrite dans l'équation (5.5). Par ailleurs, les simulations montrent que ces deux indicateurs diminuent quand le paramètre μ augmente, indiquant un système mieux décentralisé. Les valeurs atteignent le maximum possible lorsque μ devient très faible, quand il devient bien inférieur à 10^{-3}.

Au regard des figures 5.1a, 5.1c et 5.1d, il apparaît que le facteur μ amène des comportements différents autour de la valeur 10^{-3}. Ce facteur contrôle l'ajout de nouveaux validateurs dans le protocole grâce à l'ajout de la fonction exponentielle dans l'équation (5.9). Si μ est grand et bien supérieur à 10^{-3}, alors le terme $e^{-\lambda_i t}$ devient prédominant dans le calcul de $R_{g_{it}}$, l'insertion de nouveaux validateurs à la place de ceux ayant meilleure réputation est majoritaire, amenant le protocole à préférer changer de validateurs plutôt que de sélectionner les meilleurs. Les validateurs sont donc changés, selon un motif périodique, amenant la forme particulière de la limite inférieure de l'écart-type et de l'indice de Gini. Dans le cas où μ serait très faible et bien inférieur à 10^{-3}, la sélection des meilleurs validateurs est préférentielle. Le protocole tend à sélectionner la clique de validateurs optimale, avec quelques variations, car μ est non nul. Le taux de participation de ces nœuds devient donc très fort, diminuant la décentralisation.

5.2.4 Discussions

Le protocole proposé est efficace. Pour une valeur μ suffisamment faible, il tend vers une minimisation de la distance au-delà d'une trentaine de validateurs. Cette valeur de μ ne doit pas pourtant être nulle, pour éviter la formation d'un point fixe, possiblement avant l'optimal, et permettre de tester occasionnellement d'autres nœuds.

Si les simulations effectuées se sont basées sur le RTT comme métrique de mesure, dans des conditions de production, le protocole peut évaluer les performances réelles des nœuds. En effet, le temps de réception d'un *commit* est une mesure complexe dépendant de la vitesse d'exécution, de la bande passante, de la connexité... des nœuds adjacents.

Néanmoins, ce protocole dispose de nombreux défauts. Pour commencer, une valeur idéale pour le paramètre μ est difficilement identifiable et peut dépendre du réseau. Ainsi, ce paramètre est à ré-évaluer en cas d'ajout de nouveaux nœuds ou modifications du réseau.

Par la suite, le paramètre μ n'est pas le seul élément non dynamique. Au fil du temps, les réputations locales tendent vers une valeur finie. Cependant, une perturbation du réseau peut modifier cette limite et le temps nécessaire pour que les réputations locales tendent à nouveau vers cette nouvelle limite pour être conséquentes. Pire, un malveillant pourrait attendre suffisamment longtemps que sa réputation ait convergé pour que son attaque n'implique qu'une très faible variation de sa réputation locale et donc globale. Il serait donc intéressant de ne calculer les réputations locales que sur une fenêtre fixe de blocs (et donc de temps) pour éviter ces scénarii.

Pour finir, le plus grand défaut de ce protocole est qu'il ne répond pas aux critères de décentralisations. Par construction, le protocole tend vers un sous-groupe de validateurs, formant un tiers de confiance. Un compromis existe en permettant la décentralisation grâce au paramètre μ, mais les simulations montrent que l'on a une perte de performance importante avant d'avant une décentralisation satisfaisante.

5.3 Exploitation de la proximité

La recherche de performances dans un système distribué est source de nombreuses recherches. L'une des méthodes privilégiées pour améliorer un tel système est de choisir les éléments du réseau ayant une tache à accomplir les plus proches possibles.

Par exemple, dans le cas où deux utilisateurs u et v communiquent au travers d'un serveur S, obtenu via un algorithme distribué, l'application CRUX (Basescu et al., 2018) permet que ce serveur S soit sélectionné au plus près des utilisateurs pour améliorer les performances. Le framework possède une *carte* des liens entre chaque nœud, avec une distance pour chaque lien (le RTT), élément similaire à notre matrice d'adjacence.

Cette recherche de proximité est applicable dans notre sélection des validateurs. L'idée est donc de sélectionner le sous-graphe optimisant une distance, puisque l'on suppose que des nœuds plus proches peuvent communiquer plus vite et accélérer le protocole. Cette métrique peut-être égale au nombre de sauts ou la bande passante par exemple, mais le RTT est privilégié, car il permet de s'abstraire de la couche physique.

Dans la suite de cette section, des méthodes sont évoquées pour sélectionner les cliques optimales minimisant les distances entre les nœuds.

5.3.1 Sous-graphe optimal

À partir de la connaissance du réseau et des distances entre les nœuds, il est possible de former un graphe pondéré par la distance. Ce graphe est non orienté et complet, car l'on suppose que l'on peut s'abstraire du réseau par un réseau pair-à-pair. Puis l'idée est de sélectionner un sous-graphe de taille donnée minimisant cette métrique. Il s'agit d'un problème de partitionnement en clique, qui est un ensemble de problèmes NP-complet (Cerioli et al., 2008). L'objectif est de déterminer une clique de taille n dans un graphe complet de taille $\mathcal{N}$, minimisant la somme des poids des arrêtes, comme il est précisé dans le système de l'équation (5.10).

$$\begin{aligned} \min \ & g(z) \\ \text{s.t.} \ & \sum_{j=1}^{\mathcal{N}} z_j = n \\ & z = (z_1, .., z_{\mathcal{N}}) \in \mathbb{B}^{\mathcal{N}} \\ & g(z) = \sum_{j=1}^{\mathcal{N}} z_j \left(\sum_{i=j}^{\mathcal{N}} z_i \, d_{ij} \right) \end{aligned} \tag{5.10}$$

De prime abord, définir un algorithme de sélection par clique minimale optimal revient à résoudre le système (5.10). L'algorithme décrit sur la figure 5.2 permet de trouver les solutions de ce système en parcourant en profondeur l'ensemble des nœuds pour définir toutes les cliques possibles. Des optimisations de parcours ont été réalisées.

L'algorithme utilisé pour rechercher ce sous-graphe est écrit sur la figure 5.2. Cet algorithme est NP complet impliquant de grands délais de calculs pour la résolution de grands problèmes. Ainsi, divers tests pour déterminer 10 nœuds parmi 100 prennent 240 secondes. Il semble difficile d'utiliser cet algorithme pour déterminer rapidement une soixante de validateurs parmi 200.

Dans le cas où le nombre de validateurs à trouver est supérieur à $\frac{\mathcal{N}}{2}$, une variation de cet algorithme plus rapide peut être utilisée, consistant à définir la clique de poids maximale de taille $\mathcal{N} - n$, puis à la retrancher à l'ensemble des nœuds (l'algorithme 5.2 est plus efficace pour un nombre de validateurs proche de 0 que de $\mathcal{N}$).

5.3.2 Sous-graphe approché

Comme il s'agit d'un problème NP, chercher la solution optimale impliquerait une grande puissance de calculs et de longs délais. En revanche, des solutions approchées peuvent être trouvées. Un premier exemple de détermination de solution approchée est l'algorithme présenté par M. Liu et al. (2019) et présenté en section 2.2.3. Cet exemple utilise des réseaux de neurones dans le cadre du *Deep Reinforcement Learning* pour gérer les caractéristiques dynamiques et de grandes dimensions des scénarii. Le DRL montre sa supériorité dans le traitement des problèmes dynamiques et complexes problèmes dynamiques et complexes. Il s'agit donc d'une piste sérieuse pour permettre de réaliser une sélection selon nos critères et qui répond de façon dynamique aux changements du réseau.

Cette piste ayant déjà été explorés et validée par d'autres, d'autres exemples d'approximations de solution optimales ont été cherchés. Ainsi, la résolution du système a été approchée grâce à des algorithmes génétiques (Sastry et al., 2005). Il s'agit de méthodes de recherche fondées sur les principes de la sélection naturelle et de la génétique. Une population d'individus est créée de manière aléatoire, certains possédants caractéristiques répondant aux problématiques posées. Puis, une nouvelle population est créée en faisant des copies des individus les plus performants et en supprimant les moins performants. Au fil des itérations, une population répondant au système apparaît.

Le système (5.10) a donc été implémenté et résolu avec la librairie python Pymoo (Blank & Deb, 2020). La matrice générée en section 5.2.3 basé sur 200 nœuds sur un plan de taille 50×50 a été réutilisée. L'estimation du délai moyen en fonction du nombre de validateurs est disponible sur la figure 5.3. Le programme implémenté a également facilement pu être modifié pour déterminer la pire clique possible, donnant le délai moyen le plus élevé possible.

```
import numpy as np
import math


def parcours_rec(weight_max: float, validators: list,
                 non_validators: list, size_objective: int, current_weight: float,
                 matrix_weight: np.ndarray, graph_opt: list) -> tuple:
    if len(validators) == size_objective:
        return current_weight, validators
    non_validators_searched = []
    i=0
    for node in non_validators:
        i+=1
        if len(validators) + len(non_validator) - i < size_objective:
            return weight_max, graph_opt
        additive_weight = additive_weight_from_node(validators, node, matrix_weight)
        if current_weight + additive_weight <= weight_max:
            new_non_validator = non_validators[i:]
            weight_max, graph_opt = parcours_rec(weight_max,
                                        validators + [node],
                                        new_non_validator,
                                        size_objective, current_weight + additive_weight,
                                        matrix_weight, graph_opt)

    return weight_max, graph_opt


def parcours(nb_val: int, matrix: np.ndarray):
    nb_node = len(matrix)
    assert nb_val <= nb_node
    list_validators = list(np.arange(nb_node))
    return parcours_rec(math.inf, [], list_validators, nb_val, 0, matrix, [])
```

FIGURE 5.2 – Algorithme de recherche pour un sous-graphe

Les solutions approchées obtenues par cette approche sont uniques, ce qui implique un degré de décentralisation minimale. Ainsi, les valeurs d'écart-type relatif du taux de participation et l'indice de Gini de cette participation sont respectivement égaux aux pires cas σ_p et G_p. M. Liu et al. résout astucieusement cette problématique en intégrant une valeur minimale de l'indice de Gini dans le système, forçant le solveur à changer régulièrement les nœuds. Une telle solution aurait pu être appliquée pour améliorer la décentralisation, mais le temps nécessaire pour trouver une solution approchée satisfaisant (5.10) est de l'ordre de la dizaine de secondes. Une proposition utilisant les algorithmes génétiques n'est pas exploitable dans le cadre d'une blockchain, car elle réduirait au plus haut point les performances.

En revanche, les solutions approchées du pire et du meilleur délai sont utiles pour comparer les autres approches, car elles définissent des bornes supérieures et inférieures pour les délais moyens. À titre de comparaisons, le délai moyen obtenu avec une sélection aléatoire sur 1000 itérations a été représenté sur la figure 5.3. Ainsi, un algorithme de sélection efficace devrait avoir son délai moyen se rapprocher de la courbe bleue. Si sa sélection n'est pas astucieuse et tend vers de l'aléatoire, le délai moyen devrait se rapprocher de la courbe marron. Enfin, si la sélection possède une courbe du délai moyen qui se rapproche de la courbe orange, la sélection favorise les validateurs les plus éloignés.

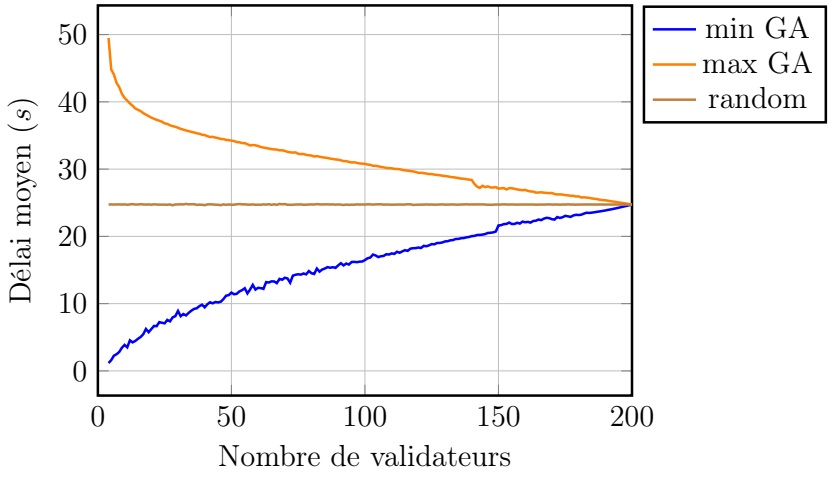


FIGURE 5.3 – Délai de la meilleure et de la pire approximation de la clique optimale par algorithme génétique, associé avec une estimation du délai pour une sélection aléatoire

5.4 Sélection centrique

Finalement, un autre protocole de sélection a été défini. Celui-ci a pour objectif de minimiser le délai tout en assurant une décentralisation importante. Comme ce protocole est amené à fonctionner avec Sabine, il doit aisément permettre un changement du nombre de validateurs.

L'idée de ce protocole est de définir un petit groupe de validateurs proche, puis de déplacer ce groupe en sélectionnant les voisins du groupe précédent comme nouveau validateur. Un mécanisme de poids rapproche virtuellement les nœuds peu choisis et éloigne ceux choisis régulièrement. Cette nouvelle distance est appelée *distance compensée*. Cependant, chaque nouvelle sélection choisit toujours les nœuds les plus proches du centre de l'ancien groupe de validateurs.

Dans cette proposition, l'idée est de définir comme point central parmi les nœuds le nœud qui minimise sa somme des distances avec les autres nœuds. Ce nœud est donc le plus "proche" des autres. Puis, il s'agit de sélectionner les n validateurs qui lui sont le plus proche. Dès lors, le mécanisme est répété, en suivant un ensemble de règles afin de permettre une variation du choix des validateurs.

5.4.1 Distance compensée

Établissement de la distance

A chaque itération t, un nœud j est choisi et est appelé "centre". À partir de la matrice d'adjacence $(d_{ki})_{k,i\leq\mathcal{N}}$ composée des RTT entre les différents nœuds et partagées entre eux, une nouvelle distance d'_{ij} est établie entre le centre j et l'ensemble des autres nœuds i suivant l'équation (5.11). Grâce à un paramètre $\kappa_{i,t}$ décrit dans l'équation (5.12) $\forall t > 2$ et qui représente la participation du nœud i dans l'ensemble des consensus, un nœud s'éloigne ou se rapproche du centre en fonction de sa participation.

$$d'_{ij} = d_{ij} \times e^{\kappa_{i,t}/\gamma} \tag{5.11}$$

$$\kappa_{i,t} = \begin{cases} \kappa_{i,t-1} + \frac{1}{n}, & \text{si } i \text{ était validateur} \\ \kappa_{i,t-1} - \frac{1}{\mathcal{N}-n}, & \text{sinon} \end{cases} \tag{5.12}$$

En effet, si un nœud i participe peu au consensus, son terme $\kappa_{i,t}$ sera positif. γ étant un terme négatif, la distance compensée d'_{ij} sera inférieure à d_{ij}. Le nœud i se rapproche donc virtuellement du centre. Comme l'algorithme sélectionne les nœuds les plus proches du centre avec la distance compensée, le nœud a davantage de chance d'être choisi comme validateur aux prochaines itérations. À l'inverse, si le nœud participe souvent, son terme $\kappa_{i,t}$ devient négatif, le nœud s'éloigne virtuellement et sera moins choisi dans les étapes suivantes. Ce phénomène a pour conséquence que la distribution des validateurs tend à être homogène et presque optimale.

Le paramètre κ

Les variations de κ sont choisies pour pouvoir se compenser au cours des itérations, afin qu'en moyenne, les distances compensées soient proches des distances réelles, pour ne pas perdre les proximités des nœuds au sein du réseau au cours des itérations. Ainsi, si l'on suppose que la distribution est optimale, ce qui implique qu'un nœud i a une probabilité d'être sélectionné $p = \frac{n}{\mathcal{N}}$, alors l'espérance de $\kappa_{i,t}$ est nulle, impliquant que le terme exponentiel est en moyenne égale à 1.

$$E\left[\kappa_i\right] = \sum_{t=0}^{\infty} \left(p \cdot \frac{1}{n} - (1-p) \cdot \frac{1}{\mathcal{N}-n} \right) = 0 \tag{5.13}$$

L'espérance reste également nulle en cas de changement du nombre de validateurs dans de futures itérations, car la variation de κ s'adapte au nombre de validateurs. De plus, comme les distances compensées sont proches en moyenne de la distance réelle et qu'aucune modification pérenne n'est effectuée, car les poids ajoutés à chaque tour se compensent, le protocole s'adapte très bien aux modifications de la matrice d'adjacence, rendant le protocole dynamique.

Le paramètre γ

Enfin, le terme γ est un terme positif permettant d'ajuster la force de la recherche de nouveaux validateurs. Sa valeur permet en effet de contrôler le diamètre du pool de validateur et sa capacité à changer de validateurs à chaque itération. En effet, une valeur de γ faible a tendance à rapprocher plus fortement les nœuds peu souvent sélectionnés et éloigner ceux souvent sélectionnés (le terme $e^{\kappa_{i,t}/\gamma}$ sera très éloigné de 1). Le diamètre du pool de validateur sera élevé, augmentant donc la distance entre les nœuds et donc le délai moyen. En revanche, comme les nœuds sollicités seront plus éloignés, le nouveau centre pourra être plus éloigné de l'ancien, les prochains validateurs seront donc plus éloignés des anciens, ce qui améliore la décentralisation.

5.4.2 Fonctionnement du protocole

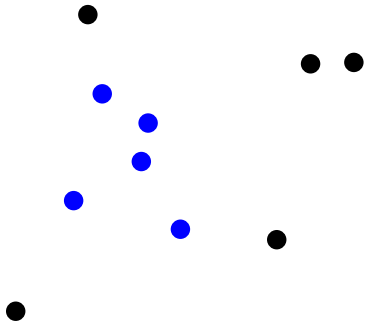
Les étapes du protocole sont illustrées avec la figure 5.4. Dans ce protocole, les nœuds partagent une matrice d'adjacence contenant les délais de transmissions entre chaque nœud. L'exécution d'une sélection se déroule en quatre étapes effectuées dans cet ordre :

0. situation initiale. Les anciens validateurs (en bleu) sont séparés des anciens non-validateurs (en noir sur la figure 5.4a ;
1. le validateur minimisant la distance avec tous les autres validateurs est sélectionné. Ce nœud est appelé “centre”. Il s'agit du nœud jaune sur la figure 5.4b ;
2. les distances compensées sont calculées grâce à la définition de l'équation (5.11). Les nœuds ayant été plus souvent validateurs s'éloignent (flèche rouge), ceux l'ayant été moins se

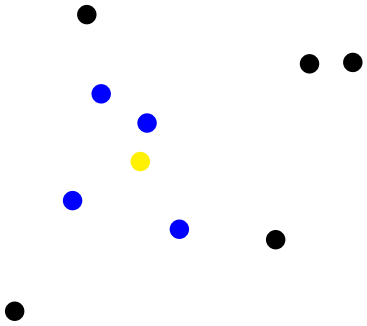
rapprochent. Leurs nouvelles positions sont sur les pointes des flèches dans la figure 5.4c. Il est à noter que les flèches ne sont pas de tailles égales, et que la valeur de ce déplacement dépend de l'historique entier de chaque nœud ;

3. les n nœuds les plus proches du "centre" selon cette distance sont sélectionnés comme validateur. Il s'agit des nœuds contenu dans le cercle noir sur la figure 5.4d et identifié par le cercle bleu ;
4. la valeur κ de tous les nœuds est mise à jour. Elle est augmentée de $\frac{1}{n}$ si le nœud est sélectionné comme validateurs, et diminué de $\frac{1}{\mathcal{N}-n}$ dans le cas contraire.

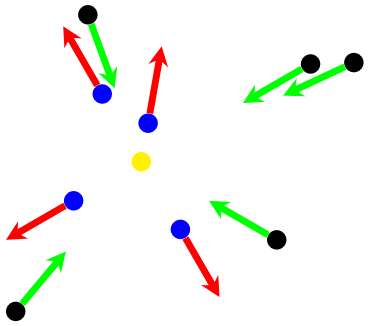
Le protocole se répète ainsi à chaque itération.



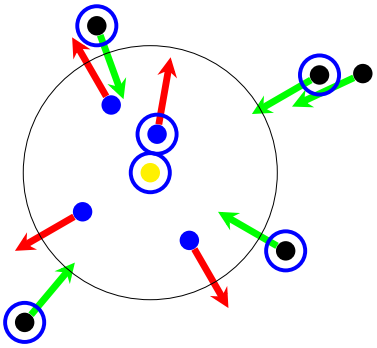
(a) Situation initiale, les anciens validateurs sont en bleu



(b) Le "centre" est sélectionné, en jaune



(c) La distance compensée est calculée, les anciens validateurs s'éloignent, les autres se rapprochent



(d) Les nœuds les plus proches sont sélectionnés comme validateurs, ceux cerclés de bleu

FIGURE 5.4 – Représentation de la sélection des validateurs, décrits dans 5.4.2

5.4.3 Validation

À l'instar de la sélection par réputation, la sélection centrique a été implémentée et testée sur un simulateur pour déterminer les performances et comparer les deux approches.

Simulation

Le simulateur utilisé est relativement proche de celui exploité lors des simulations en section 5.2.3 si ce n'est que l'algorithme de sélection est différent. La même matrice d'adjacence

composée de 200 nœuds projetés aléatoirement sur un plan de dimension arbitraire 50×50 a été utilisé afin de pouvoir comparer les deux approches. Ce simulateur est disponible sur `https://github.com/inpprenable/centric-selection`.

Les mêmes métriques ont également été mesurées. Comme le protocole est un peu plus rapide, davantage d'itérations ont été effectuées. Ainsi, pour chaque expérience, 200 itérations ont été effectué afin d'atteindre un régime proche du régime permanent, puis 2000 itérations ont été effectués en mesurant à chaque fois le délai moyen, la participation des nœuds et le temps de calcul.

Résultats

La figure 5.5 représente 6 itérations de l'algorithme de sélection centrique. On peut ainsi voir que le centre représenté en couleur jaune change de nœud au cours des itérations, créant un déplacement d'une zone de validateurs. Cette zone parcourt l'ensemble des nœuds au cours du temps, impliquant que chaque nœud devient validateur dans les mêmes proportions. La vitesse de déplacement et le diamètre de cette zone sont impactés par le paramètre γ.

La figure 5.6 rassemble les mesures des différentes métriques en fonction du nombre de validateurs, et de valeurs du paramètre γ.

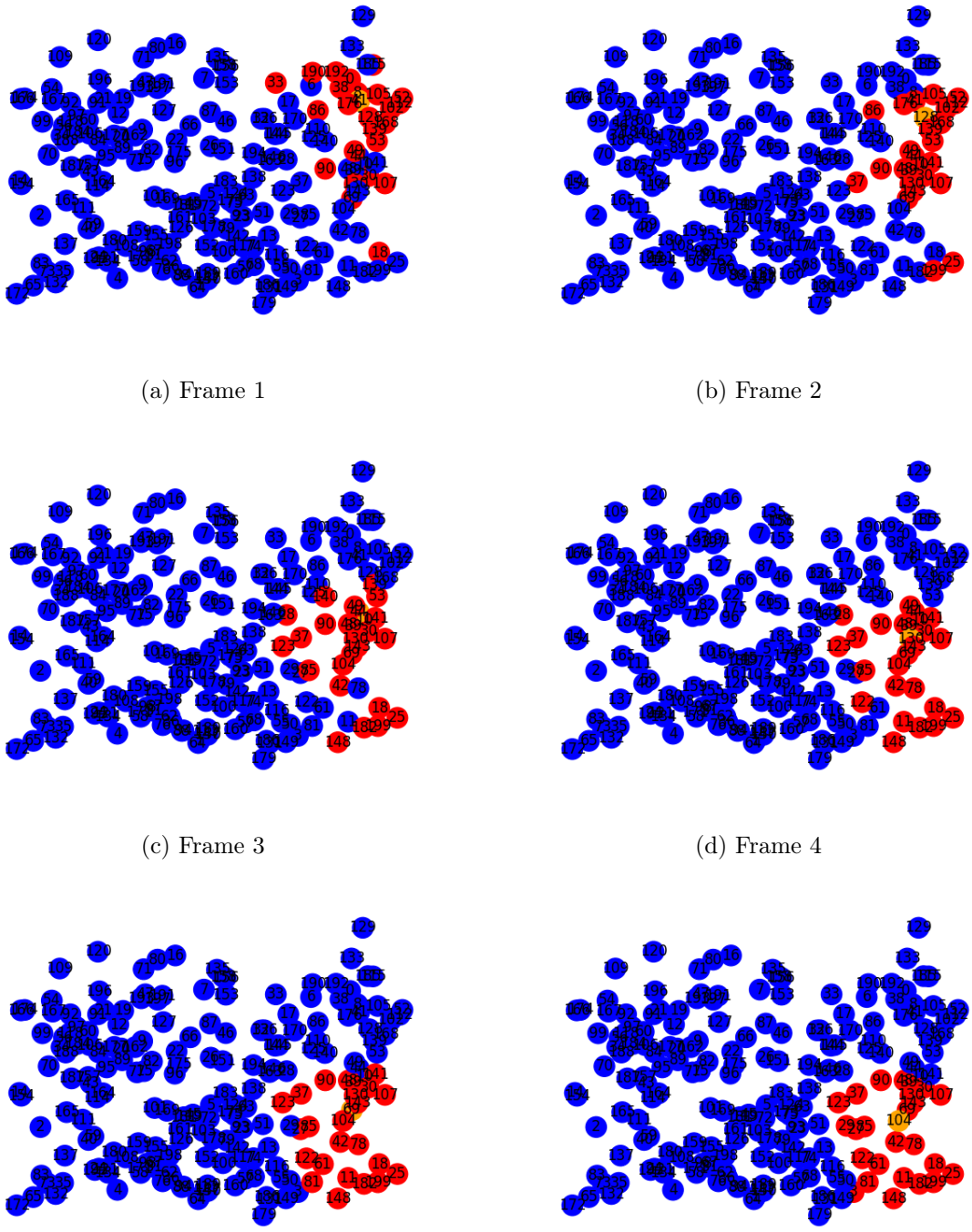
Plus précisément, la figure 5.6a représente le délai moyen de transmission entre deux nœuds. Les courbes de ce délai sont très proches de la courbe minimale estimée par algorithme génétique dans la figure 5.3 et éloigné de la sélection aléatoire pour tout nombre de validateurs. La proposition de sélection centrique est donc très efficace. On remarque également que plus le paramètre γ est élevé, plus la courbe est proche des délais optimums. Cette remarque confirme ce qui avait été théorisé dans la section 5.4.1 concernant ce paramètre. Ainsi, un paramètre élevé implique une "zone" de validateurs avec un diamètre réduit, ce qui implique des validateurs proches.

La figure 5.6b présente le temps nécessaire pour calculer une itération de la sélection. Comme la sélection par réputation, le temps de calcul ne dépend pas de son paramètre. Cependant, le temps de calcul d'une itération est 6 à 30 fois plus rapide que la sélection par réputation pour un nombre de validateurs équivalent. Ce temps de calcul est ainsi bien inférieur au temps nécessaire pour ajouter un bloc, impliquant donc que cette sélection aura peu d'impact sur le temps d'ajout de bloc dans une blockchain.

Ensuite, les figures 5.6c et 5.6d mesurent respectivement l'écart-type relatif du taux de participation des nœuds et l'indice de Gini de cette participation. Les valeurs mesurées sur cette sélection sont très faibles, sauf dans le cas $\gamma = 100$. Pour des valeurs de $\gamma \leq 1$, ces deux indices ont des valeurs proches de 0. Cette sélection fournit donc une excellente décentralisation, ce qui n'est guère étonnant, car la "zone" autour du centre se déplace sur l'ensemble des nœuds. Le cas $\gamma = 100$ est intéressant, car les courbes sont proches des courbes des pires cas σ_p et G_p. Le paramètre doit alors être si fort qu'il ne rend presque nul l'argument de l'exponentiel dans l'équation (5.11).

Enfin, pour des nombres de validateurs très faibles, de l'ordre de 4 ou 5 validateurs, les performances chutent fortement. Le débit et les métriques de décentralisations augmentent. Cependant, ces comportements correspondent à des cas où le nombre de validateurs est très faible et où des problématiques de sécurités déconseilleraient d'accéder.

En résumé, la sélection centrique est une sélection donnant des performances efficaces avec un délai moyen proche de l'optimum et fournissant un degré de décentralisation très élevé. De plus, le calcul de cette sélection est efficient et ne devrait pas perturber les performances du



(a) Frame 1

(b) Frame 2

(c) Frame 3

(d) Frame 4

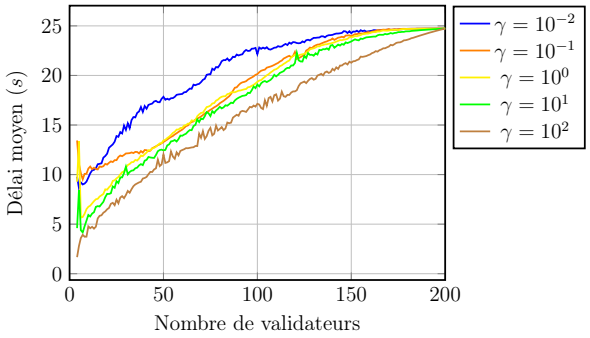
(e) Frame 5

(f) Frame 6

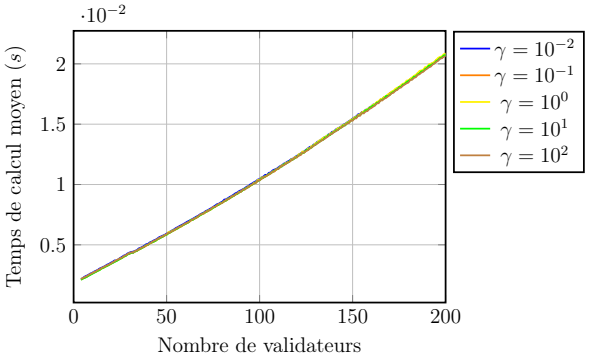
Les nœuds non-validateurs sont représenté en couleur jaune, les validateurs en couleur rouge, le centre (qui est également validateur) est en couleur jaune
On peut constater que les validateurs se regroupent en zone. À chaque itération, de nouveaux validateurs sont sélectionnés en replacement d'ancien validateur. La zone et le centre se déplacent légèrement vers le bas au fil des itérations.

FIGURE 5.5 – Exemple de 6 itérations de la sélection centrique, avec 200 nœuds

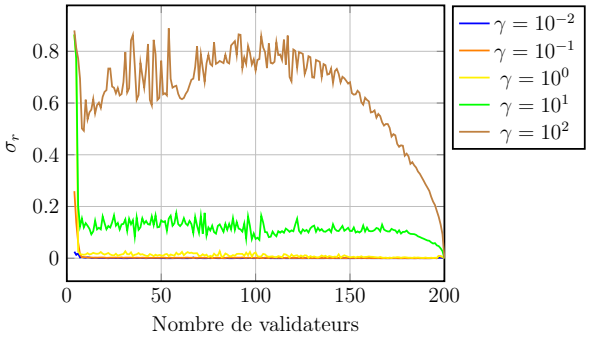
consensus. Dans l'exemple de notre réseau de taille 50×50, une valeur $\gamma = 1$ est un excellent compromis donnant des délais faibles et une excellente décentralisation.



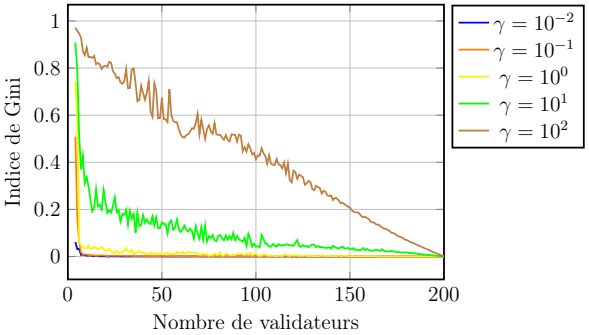
(a) Délai entre deux nœuds moyen en fonction du nombre de validateurs



(b) Temps de calcul moyen en fonction du nombre de validateurs



(c) Écart-type relatif du taux de participation en fonction du nombre de validateurs



(d) Indice de Gini sur la participation en fonction du nombre de validateurs

FIGURE 5.6 – Mesures moyennes diverses sur 2000 itérations en fonction du nombre de validateurs sur une chaîne composé de 200 nœuds

5.5 Perspectives

Les différents protocoles présentés ont vu leurs performances évaluées grâce à des simulations. Si ces sélections sont développées et intégrées dans des blockchains, des problématiques d'interactions avec les différents éléments constitutifs de la blockchain apparaissent. Le problème d'intégration principale de cette sélection est ainsi l'établissement de la matrice d'adjacence, pour rendre le système dynamique. Ensuite, des interactions sont possibles entre la sélection et le framework Sabine décrit dans la section précédente.

5.5.1 Dynamique de la sélection

Dans le cadre des simulations, une matrice d'adjacence a été fournie aux protocoles. Cependant, dans le cadre d'une intégration dans une blockchain, cette matrice est à établir pour réaliser les sélections. Ce point pourtant essentiel n'est pas abordé dans M. Liu et al. qui est une des principales références de cette section.

Mesure du RTT

La sélection par réputation définie en section 5.2 ne nécessite pas de mesure d'une distance. Les valeurs associées au temps de transmission des messages et la performance des voisins sont sous-entendues par les paramètres α et β partagé par les nœuds.

La sélection centrique nécessite quant à elle cette matrice pour déterminer ses distances compensées. La façon la plus simple de les établir est de réaliser régulièrement pour chaque nœud une requête ICMP vers chacun de ses voisins, puis de mesurer le temps nécessaire pour avoir la réponse. Ainsi, le nombre minimum de requêtes type ICMP est donc de n^2 requêtes pour que chaque nœud connaisse sa distance par rapport à ces voisins. Ce nombre peut donc être très important en cas de grands réseaux et cette estimation doit être répétée régulièrement.

Une autre distance peut être utilisée en lieu et place du RTT. Durant le consensus PBFT, après avoir reçu un message *pre-prepare*, les nœuds envoient en *broadcast* un message *prepare*, puis reçoivent de la part de ces nœuds un message *commit*. Le délai entre l'émission de ce message *prepare* et la réception du message *commit* peut possiblement être exploité comme nouvelle distance et éviter l'émission de requête ICMP. Cette piste amène d'autre problématique. Ainsi, dû aux problèmes de synchronisation, ce délai peut être possiblement négatif.

Cohérence de la matrice d'adjacence

La sélection centrique pourrait fonctionner sans partage d'une matrice d'adjacence unique. Dans ce cas, chaque nœud devant orchestrer la sélection la réalise avec sa version de la matrice d'adjacence. Ce paradigme amène des problématiques de sécurité. En effet, la sélection n'est plus vérifiable. Qu'elle doit alors être le comportement des nœuds en cas de désaccord ? Pire, un nœud malveillant pourrait profiter de ces différences pour planifier une attaque.

Afin d'avoir une solution vérifiable et sécurisée, elle doit donc être déterministe, ce qui passe par l'établissement d'une matrice d'adjacence unique et cohérente. De plus, pour prendre en compte les modifications du réseau, elle doit être dynamique.

La proposition de Zeng et al. (Zeng et al., 2022) exploite un *smart contract* au sein de la chaîne pour partager les valeurs de α et β. Une proposition similaire peut être réalisée pour établir une matrice d'adjacence. Les nœuds peuvent alors mettre régulièrement à jour la matrice en envoyant des transactions spéciales, dans le cas où ils deviennent Proposer par exemple.

Des mécanismes de réputations peuvent aussi être ajoutés pour certifier la confiance en une valeur proposée par l'un des nœuds.

5.5.2 Interactions avec Sabine

À terme, Sabine et la sélection des validateurs doivent fonctionner de concert. En même temps que le Proposer propose une nouvelle valeur de la taille du pool de validateur, il propose un nouvel ensemble de validateurs. Les autres validateurs acceptent ou non ces propositions en accord avec leurs instances de Sabine et de la sélection.

Ces deux algorithmes peuvent amener de meilleure performance en travaillant de concert. Une critique possible à la sélection est qu'elle rend prévisible les futurs validateurs pour améliorer les performances et réduire les délais. Sauf qu'associé avec Sabine, ce gain de performance peut également se traduire par une augmentation du nombre de validateurs. En effet, avec des débits réduits, Sabine dispose de plus de marge pour augmenter le nombre de validateurs.

Supposons que le modèle de la figure 4.9c corresponde aux performances sur le plan utilisé dans les simulations. Sur une chaîne de 200 nœuds, avec 50 validateurs sélectionnés aléatoirement, la capacité est de $44tx/s$, car un délai de $25s$ est mesuré. En appliquant la sélection centrique

par exemple, ce délai tombe à $13s$. Sabine peut alors augmenter le nombre de validateurs à 59 pour une capacité égale, rendant la chaîne plus résistante et plus cohérente.

5.6 Conclusion

En conclusion, grâce à des algorithmes génétiques, la détermination des cliques minimales d'une taille donnée a été approximée sur n'importe quel réseau alors qu'il s'agit d'un problème NP-complet, permettant donc de déterminer le gain maximal possible d'une sélection astucieuse des validateurs par rapport à une sélection aléatoire.

Deux protocoles de sélection des validateurs ont ensuite été proposés. Ces deux protocoles permettent de sélectionner les validateurs parmi les différents nœuds afin de réduire le temps de transfert moyen entre deux validateurs. Le premier protocole, la sélection par réputation, permet de sélectionner les validateurs sur plusieurs critères résumés dans un temps de réponse. Ce protocole tend ainsi à privilégier un unique sous-groupe de validateurs, amenant des problématiques de décentralisation.

Le second protocole, la sélection centrique, privilégie la sélection de validateurs proches de la précédente itération, mais propose une variation régulière de l'ensemble des nœuds, ce qui a pour conséquence de fortement augmenter le degré de décentralisation. Ce protocole nécessite la mise en place d'une matrice d'adjacence à travers un *smart contract* à définir. D'autres protocoles peuvent être encore définis, en exploitant par exemple les DRL.

Associé avec Sabine, ces différentes sélections permettent d'améliorer (théoriquement) la sécurité et la cohérence d'une blockchain en augmentant le nombre de validateurs. La mise en place d'une plateforme expérimentale exploitant ces sélections aurait à confronter les mesures aux simulations et établir ainsi l'ensemble des gains et limites de nos propositions.

Conclusions et perspectives

En conclusion, la blockchain vise à définir un consensus pour définir une seule version du registre à travers le réseau. Deux types de consensus coexistent, l'un basé sur la construction d'une preuve individuelle, l'autre sur une décision collective grâce à un ensemble de votes. Cette première famille de consensus peut se déployer facilement à grande échelle, contrairement à la seconde qui impose un modèle de cohérence bien plus strict.

Plusieurs autres choix de conception composent la blockchain sans toutefois revêtir la même importance. Des interactions entre ces choix existent et sont à prendre en compte lors de la définition de la chaîne. Chacun des choix de conception, du consensus à l'application, impacte les performances de la blockchain. Afin de critiquer ces choix, des bancs d'expérimentation sont développés dans la littérature scientifique. Les deux métriques principalement utilisées sont ainsi le débit et la latence. Ces différentes évaluations ont pour la plupart été menées dans l'objectif de définir un meilleur consensus unique, et par conséquent, de montrer la suprématie d'une des deux familles sur une autre. Cependant, les performances des blockchains sont relatives à des contextes d'exécution tellement différents que cette question perd de son sens.

Nous avons étudié les interactions entre les paramètres liés à la blockchain et à l'infrastructure, et la manière dont ils influencent les performances globales. Un modèle du consensus Ethereum a été proposé pour résoudre cette question. Quatre paramètres ont ainsi été identifiés : la taille du bloc, la taille des transactions, un temps de propagation et un paramètre codé en dur dans le consensus. Pour une application donnée, la taille du bloc est donc le seul paramètre modifiable.

Au travers d'une application décentralisée appelée "FarMarket" que nous avons développée, une méthodologie complète a été introduite pour aider les intégrateurs de solutions logicielles à mieux comprendre (et mesurer) les performances de qualité de service qu'un écosystème pourrait atteindre et supporter à long terme. L'influence de la taille du bloc a été montrée et nous avons mis en évidence un débit particulier appelé *capacité*. Cette capacité correspond au débit maximal atteignable à long terme. Le débit de transaction ajouté ne peut excéder ce débit et si le débit de transaction proposé est supérieur à la capacité, la latence va tendre vers l'infini. Améliorer les performances revient donc à augmenter ce débit.

Une fois les performances d'une blockchain mesurées avec différents critères, les goulots d'étranglement responsables des ralentissements et perte de débit d'ajout des blocs peuvent être localisés. Dès lors des solutions peuvent être mises en place pour augmenter la capacité et les performances. Des propositions existent dans la littérature pour améliorer les performances. Les services associés à la blockchain peuvent être optimisés ou la blockchain peut autoriser le *sharding*, accélérant fortement le débit. Dans le cas des consensus BFT, la proposition la plus souvent utilisée est de séparer les nœuds en un ensemble de validateurs réalisant le protocole et les autres. Des protocoles existent pour sélectionner de manière sécurisée les validateurs, mais la question du nombre de validateurs n'a jamais été traitée, alors qu'elle est déterminante. Généralement, ce paramètre est laissé fixe.

À cette fin, nous proposons le protocole Sabine (Self-Adaptive BlockchaIn coNsEnsus) pour

adapter le nombre de validateurs de telle manière que la capacité de la chaîne est proche du débit demandé. Sabine base ses décisions sur un modèle obtenu par apprentissage automatique, intègre un nombre minimal de nœuds fixé par l'administrateur, et est capable de prendre en compte des délais de différentes natures apparaissant dans le réseau. Ce protocole a été évalué sur deux plateformes différentes, les performances montrant que ce protocole a augmenté de manière significative la satisfaction des taux de débit de la demande tout en couvrant, voire améliorant, les problèmes de sécurité et de cohérences.

La littérature a explicité des propositions dans le but d'accélérer le consensus en sélectionnant les validateurs les plus performants et les plus proches. Là où Sabine détermine combien de validateurs sont nécessaires, cette sélection choisit les meilleurs. Les solutions trouvées dans la littérature ne satisfont toutefois pas les impératifs de décentralisation ou de ressources de calcul pour l'établissement d'une blockchain (notamment dans l'IoT). Deux propositions de mécanismes de sélection ont ainsi été avancées. Le premier utilise un système de réputation pour privilégier un unique sous-groupe de validateurs avec des temps de réponse plus faible. Le second privilégie la sélection de validateurs proches, mais impose un changement régulier de validateurs. Ces deux protocoles ont été évalués et leurs efficacités ont été démontrées, avec un net avantage du second protocole vis-à-vis des contraintes de décentralisation. Associés avec Sabine, nous avons montré que ces sélections permettraient une augmentation de la sécurité dans une blockchain et renforceraient la cohérence du système.

Cette thèse soulève principalement la problématique de la sélection des validateurs dans le cadre du consensus BFT. La proposition présentée avec Sabine au cours du chapitre 4 pourrait être améliorée de différentes façons. Le protocole présenté cherche à réaliser un compromis entre sécurité et débit de transaction ajouté grâce à son modèle liant nombre de validateurs et capacité. De nouvelles dimensions peuvent être ajoutées à ce modèle telles que le coût énergétique associé au réseau et au matériel. On peut supposer raisonnablement qu'un réseau avec un nombre de validateurs plus faible sera plus économe en énergie et que l'énergie est un paramètre à faire diminuer. Dans le cas actuel, Sabine cherche à maximiser le nombre de validateurs. La prise en compte de l'énergie viserait à définir une borne maximale à ce nombre de validateurs. Le nombre de dimensions du modèle n'est pas limité. Le système pourrait également prendre en compte la bande passante disponible, pour laisser de la marge réseau aux équipements IoT.

D'autres améliorations peuvent également être apportées à Sabine. Sa grande limitation est sa dépendance vis-à-vis d'un modèle du réseau. La détermination de ce modèle est conséquente et dure plusieurs jours afin de pouvoir définir suffisamment d'échantillons pour réaliser l'apprentissage automatique, sans être transférable à un autre environnement. Des recherches plus poussées permettraient de supprimer la dépendance à ce modèle.

Ensuite, la principale critique de Sabine est le risque qu'un attaquant exploite le protocole pour diminuer le nombre de validateurs et ainsi facilite une attaque sur le réseau. Cet attaquant pourrait par exemple réaliser pendant un temps très court une grande demande d'ajout de transaction à la chaîne, implique une baisse du nombre de transactions. Pour éviter ce genre d'attaque, la baisse du nombre de validateurs pourrait être contrôlée pour rendre cette attaque plus longue à mettre en place et l'origine des transactions pourrait être contrôlée, afin par exemple de limiter la demande par utilisateur.

Ces différentes propositions de protection ne détournent pas le fait que la sécurité minimale de la chaîne est assurée par la limite définie par l'administrateur. Ainsi, le protocole Sabine serait plus intéressant à intégrer dans le cas de blockchain exploitant le *sharding* décrit en section 2.2.2 où le protocole pourrait être exploité pour déterminer dynamiquement la taille des *shards* en fonction de la demande, et peut être si davantage de recherche sont effectués, le nombre de

shards.

Sabine pourrait être étendue sur d'autres consensus, comme la preuve de travail. En effet, le modèle présenté en section 3.4.4 montre que le paramètre l défini permet de contrôler la durée de minage du bloc et donc le débit d'ajout de transaction. Un protocole semblable à Sabine contrôlant cette fois ce paramètre au lieu d'un nombre de validateurs pourrait être exploité pour adapter le temps de minage au débit soumis. Une telle version de Sabine devrait répondre à d'autres problématiques, par exemple comment assurer que l'ordre vienne d'une décision collective et non d'un individu unique potentiellement malveillant.

Enfin, la sélection des validateurs définie en chapitre 5 n'a été que simulée. Les performances des deux protocoles proposés doivent encore être évaluées, par exemple dans les deux plateformes définies dans la section 4.1.2. Par la suite, cette sélection pourra être utilisée en complément de Sabine pour augmenter la sécurité comme décrit en section 5.5.2. Cette interaction n'est que théorisée et nécessite d'être étudié avec la mise en place d'une plateforme expérimentale.

Publications

Article en revue internationale

— Guilain Leduc, Sylvain Kubler & Jean-Philippe Georges. Innovative blockchain-based farming marketplace and smart contract performance evaluation. *Journal of Cleaner Production*, 306 :127055, 2021. ISSN 0959-6526. doi:10.1016/j.jclepro.2021.127055. URL `https://www.sciencedirect.com/science/article/pii/S0959652621012749`.

Communications dans des conférences internationales avec actes

— Guilain Leduc, Sylvain Kubler & Jean-Philippe Georges. Adapting the number of validator according the need in a bft chain with sabine. *IFAC-PapersOnLine*, 55(8) :64–70, 2022a. ISSN 2405-8963. doi:10.1016/j.ifacol.2022.08.011. URL `https://www.sciencedirect.com/science/article/pii/S2405896322010849`. 6th IFAC Symposium on Telematics Applications TA 2022.

— Guilain Leduc, Sylvain Kubler & Jean-Philippe Georges. Sabine : Self-Adaptive BlockchaIn coNsEnsus. In *9th International Conference on Future Internet of Things and Cloud, FiCloud 2022*, Rome, Italy, August 2022b. URL `https://hal.archives-ouvertes.fr/hal-03777391`.

A

Annexes

A.1 Autres protocoles basés sur des votes

A.1.1 Algorithme CFT : Paxos

Non-trivialité

La condition de non-trivialité (*Non-triviality*) implique que seules des valeurs proposées peuvent être apprises. Dans le cadre des hypothèses de Paxos, il n'existe pas d'utilisateurs malveillants dont le but est de tromper les propositions des membres honnêtes. De plus, si les nœuds sont défaillants, ils cessent d'envoyer des messages et n'envoient pas de messages erronés. Ces deux hypothèses simplifient la condition de *non-trivialité*.

Les algorithmes Paxos sont une famille d'algorithmes qui ont été définis par Leslie Lamport (Lamport, 1998) à travers la métaphore d'un système de consensus législatif utilisé par un parlement sur l'île de Paxos en Grèce. La problématique de ces algorithmes est de trouver un consensus dans le cadre d'un réseau de nœuds faillible, c'est-à-dire que les nœuds du réseau peuvent avoir des pannes, c'est ce que l'on appelle la *CFT* (*Crash Fault-Tolerance*). Ces algorithmes garantissent la cohérence des données, mais ne peuvent garantir la progression (trouver un consensus). L'algorithme suppose en revanche que les moyens de communication sont infaillibles, c'est-à-dire que tout message émis arrive à son destinataire de manière intègre.

Problématique

La problématique est donc de définir un réseau répliquant des informations tolérant aux fautes, c'est-à-dire que l'on suppose que n'importe quel nœud du système peut défaillir. Comme chaque membre du réseau possède une copie de la base de données, les informations partagées entre les membres du réseau sont les modifications de la base de données.

On peut définir également un ensemble de conditions que doit remplir cet algorithme (Lamport, 2005, 2006). Le système doit donc garantir :

— La cohérence des données ;

— La non-trivialité ;

— La stabilité ;

— La *liveness*.

L'algorithme

Rôles Paxos définit plusieurs rôles pour les différents acteurs prenant place. Un processus peut jouer plusieurs de ces rôles au même instant.

— Client Les *Clients* envoient des requêtes au *Leader*, comme la modification d'une variable. Ils attendent un acquittement de la réussite de la modification, ou une réponse à la requête dans le cas nécessaire.

— Acceptor Les agents sont l'un des trois agents participants au consensus de l'algorithme. Ce sont les agents disposant d'une copie de l'état du système. Ils se regroupent en *Quorum* et décident ensemble de la validité d'une modification. Ainsi, lorsqu'une modification est proposée, elle n'est appliquée que si une portion assez grande des *Acceptors* l'accepte. Pour la suite, nous définirons $Q \subset \mathbb{N}$ l'ensemble des ID des Acceptors d'un quorum.

— Proposer Le *Proposer* est l'agent qui retransmet la requête aux *Acceptors*. C'est l'agent chargé de coordonner les différentes étapes de l'algorithme pour que les *Acceptors* acceptent une modification. Plusieurs *Proposers* existent au sein du réseau, cependant strictement un seul est nécessaire pour effectuer une modification de la base de données. Les *Proposers* partagent souvent le rôle avec les clients.

— Learner Le *Learner* est le troisième agent. Il centralise les lectures des variables et renvoyer la réponse au *Client*. Son rôle dépend également de l'interprétation de la condition de monotonicité. Il S'agit souvent d'un *Acceptor*.

— Leader Le *Leader* est un *Proposer* particulier qui est l'agent de référence durant une modification des données. Il ne peut exister qu'un seul *Leader* au cours de l'exécution du protocole Paxos pour que le système progresse. Si deux exécutions ont lieu en même temps, avec deux *Leaders* différents, le système peut être bloqué. Cependant, la cohérence et l'intégrité des données sont préservées.

Étapes Un algorithme Paxos se déroule en 6 étapes différentes (Lamport, 2001). La première et la dernière consistent en l'interaction du réseau d'agent avec les clients extérieurs. L'algorithme permet d'assurer que le système reste cohérent si l'une des étapes ou l'un des nœuds ne peut être effectué à cause d'un défaut d'un des éléments.

1. Request Un Client effectue une *request* sur la base de données distribuée. Il envoie cette requête à un nœud Proposer pour qu'il la transmette à l'ensemble des Acceptors constituant le Quorum.
2. Prepare Un Proposer sélectionne un numéro de proposition n et envoie une requête *prepare* avec ce nombre n à une majorité d'Acceptors.
3. Promise Les Acceptors recevant cette requête et dont le numéro n est plus grand que toutes les requêtes *prepare* auxquelles ils ont répondus, ils envoient une promesse à l'aide d'une requête *promise* au Proposer assurant qu'ils ne répondraient pas aux requêtes *prepare* avec un numéro de proposition inférieure à n. Cette promesse contient également, si elle existe, la proposition que l'Acceptor a acceptée et dont le numéro de proposition est le plus élevé (forcément inférieur à n), avec ce numéro de proposition.
4. Accept Si le Proposer reçoit plus d'une majorité de promesses *promise* de la part du quorum d'Acceptors, il envoie alors à chacun de ces Acceptors une proposition numérotée n avec une valeur v' estampillée avec la valeur n dans une requête *accept*. Cette valeur v' correspond à la proposition avec le numéro le plus élevé parmi les promesses reçues. Si aucune promesse ne comprenait de telle proposition, v' correspondra à la valeur v issue de la requête effectuée

par le Client. Comme plusieurs propositions de modification peuvent coexister lors de la phase une, cette sélection de v' permet de sélectionner la dernière proposition envoyée. La définition mathématique de v' est donnée ci-dessous.
Soit Q' l'ensemble des id des Acceptors ayant répondu, $v' = Max_{propo}(\{(v_i, n_i)\}_{i \in Q'}, v)$ où

$$\forall Q' \subset Q, Max_{propo}(\{(v_i, n_i)\}_{i \in Q'}, v) = \begin{cases} v, & \text{if } \{v_i\}_{i \in Q'} = \emptyset \\ v_k / k \in Q' \wedge max(\{n_i\}_{i \in Q'}) = n_k, & \text{otherwise} \end{cases}$$

5. Accepted Si un Acceptor reçoit un *accept*, il accepte la proposition s'il n'a pas fait une promesse à la suite d'une requête *prepare* avec un numéro de proposition supérieure à n. Il renvoie alors une confirmation de modification au Proposer avec une requête *accepted*. Il envoie également une requête aux Learners pour les notifier d'une modification des données.
6. Response Une fois que les Learners sont notifiés des modifications, ils peuvent donner la réponse à la requête au Client via une *response*.

Les étapes *Prepare* et *Promise* correspondent à une étape où le *Proposer* s'informe de l'état du système et informe qu'une modification a lieu. Un mécanisme assimilable à un sémaphore a lieu avec le numéro de proposition n. Les étapes *Accept* et *Accepted* correspondent à une seconde phase où le réseau modifie les données. Ces différentes étapes sont représentées sur la figure A.1.

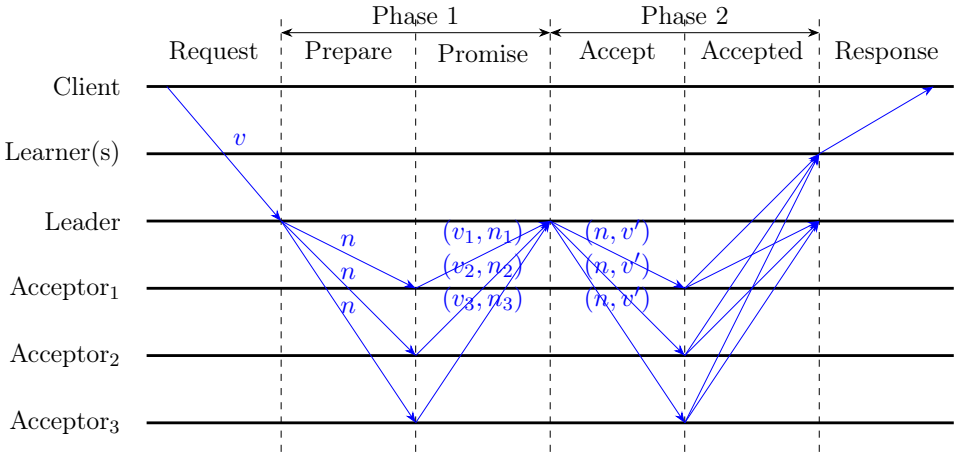


FIGURE A.1 – Diagramme représentant le flux de message dans un cas général qui a réussi (Lamport, 2001) avec $v' = Max_{propo}(\{(v_i, n_i)\}_{i \in \{1,2,3\}}, v)$.

Accès à la base

La modification de la base de données se réalise à travers des requêtes en écriture (**Update**), qui déclenche une élection via l'algorithme représenté figure A.1 afin d'assurer la cohérence de la réplication de la modification sur tous les nœuds du réseau.

Les lectures de la base de données, les **Read**, se décomposent en deux types. D'une part, il y a les *fast read*, pour *lecture rapide*, qui sont des lectures rapides auprès d'une des réplications de la base d'un Acceptor. Cette requête est rapide, mais ne reflète pas forcément les récents changements de la base de données, par exemple dans le cas où le nœud Acceptor questionné est défaillant ou qu'il n'a pas encore reçu de requête *Accepted* associée à la modification de cette valeur.

Pour s'assurer que la valeur demandée est cohérente avec l'entièreté de la base de données, il est préférable de réaliser un *slow read*. Cette requête suit également le même protocole décrit par la figure A.1 afin d'assurer la cohérence de la réponse.

Une solution simple en cas de conflits entre deux clients avec deux valeurs différentes est de comparer les numéros de propositions associés à leurs *slow read*. Les requêtes *Slow Read* pouvant être coûteuses et afin de ne pas associer une valeur à un numéro de proposition, les *Learners* ont été introduits. Dans cette configuration, la base de données est divisée en domaines et chacun de ces domaines est associé à un unique *Learner*, qui possède l'état actuel de la base de données. Ce *Learner* est mis à jour des modifications de son domaine par les *Acceptors* comme montré dans l'étape *Accepted* de la figure A.1. Le *Learner* est également choisi par le quorum pour une période donnée et renouvelable, afin de pouvoir le changer en cas de défaillance. Ce système de *Learners* a cependant de nombreux défauts. L'accès à la base est centralisé sur un seul nœud, qui peut défaillir et dont l'état ne pas toujours être cohérent avec la base de données. De plus, la division en domaine est problématique. En effet, pour qu'une modification respecte la condition de monotonicité, si cette modification concerne plusieurs domaines, alors ces domaines doivent être gérés par un même *Learner*.

A.1.2 Consensus de Hyperledger Fabric

Principes

Une des principales critiques de ces systèmes est que les performances sont limitées par l'exécution séquentielle des transactions qui impose un ordonnancement. De plus, les opérations doivent être déterministes pour arriver à un consensus, ce qui implique une plateforme lourde, utilisant un unique langage rarement standard. Afin d'accélérer le processus de décision, *Hyperledger Fabric* (Androulaki et al., 2018 ; Foundation, 2020) n'adopte pas la même architecture *order - execute*. En effet, les étapes *Pre-Prepare*, *Prepare* et *Prepared* dans le cadre d'un consensus PBFT ne servent qu'à ordonner les transactions proposées avant d'être exécutées et validées par l'étape *Commit*. Hyperledger remet donc en cause l'ordonnancement des modifications par l'emploi d'une architecture *execute - ordering - validation*.

1. *Execute* : les transactions sont d'abord exécutées et validées localement ;
2. *Ordering* : les transactions sont ensuite ordonnées et diffusées à travers un protocole de consensus ;
3. *Validation* : Les transactions sont ensuite validées par des applications spécifiques de confiances.

Ainsi, a contrario d'un consensus de blockchain classique qui cherche à ordonner les *entrées* d'une transaction, *Hyperledger Fabric* ordonne les *sorties* combinées avec les dépendances des états sollicités.

La logique d'exécution se fait à travers des smart contracté appelé dans le cadre de Hyperledger Fabric des *chaincodes*. La politique d'approbation des *chaincodes* peut être paramétré par les administrateurs.

Les nœuds se décomposent entre trois types :

— *Les Clients* : ce sont eux qui proposent et diffusent les propositions de transactions ;

— *les Pairs* : Ils exécutent et valident les propositions de transactions. Ils possèdent chacun une copie de la blockchain. Seuls les pairs dédiés à une transaction (endosseurs, *endorsers*), comme spécifiés dans la politique d'approbation, exécutent et valident ces transactions ;

— *les Nœuds du Service d'Ordonnancement (OSN)* ou *Orderer* : ce sont des nœuds qui forment collectivement le *service d'ordonnancement*. Ils ignorent l'état de la blockchain et ordonnancent les transactions validées.

Phase d'exécution

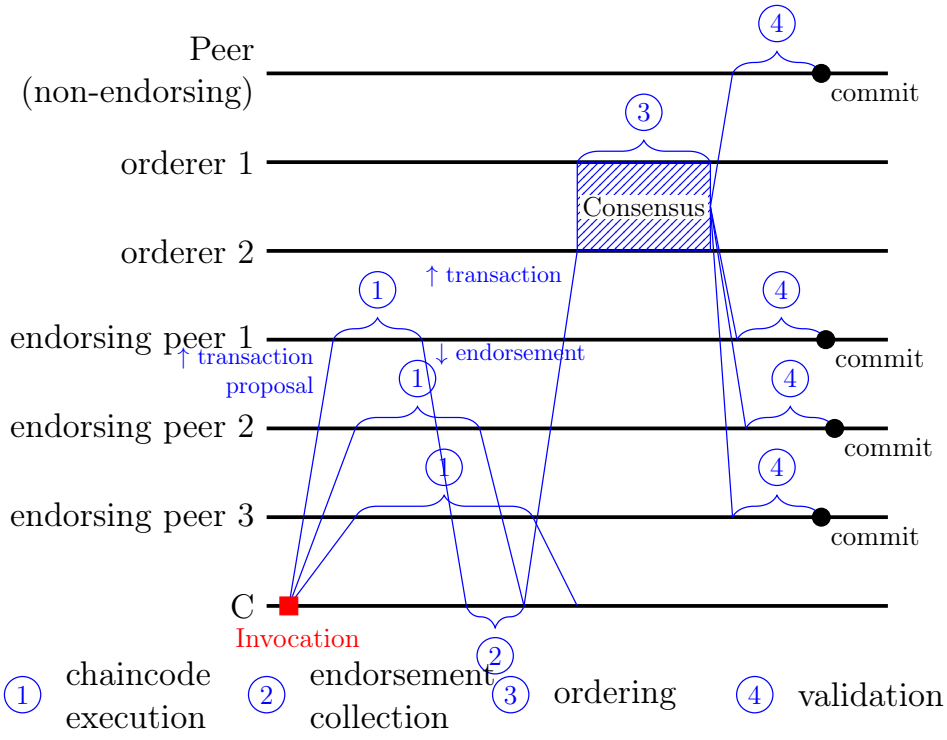


FIGURE A.2 – Diagramme représentant le flux de message dans un cas général qui a réussi (Androulaki et al., 2018).

Execute Les Clients forment, signent et envoient les propositions de transactions (*transaction proposal*) composées de :

— l'identité du Client émetteur de la proposition ;

— l'opération à exécuter (le code, appelé également *transaction payload*) ;

— les paramètres utilisés ;

— un identifiant pour le *chaincode*.

Les Clients envoient ensuite ces propositions à un ou plusieurs endosseurs. Ces endosseurs sont définis par la politique d'approbation définie par les administrateurs. L'endosseur simule alors et exécute la proposition dans un conteneur, isolé des autres processus, à partir de l'état de sa blockchain locale. Les endosseurs construisent alors un message appelé *endorsement*, composé :

— les résultats d'exécutions composés des :

readset : les dépendances de la transaction. Il s'agit des clefs lues durant la simulation ;

writeset : l'ensemble des états mis à jour par la simulation ;

— un numéro de transaction ;

— l'identifiant de l'endosseur ;

— sa signature.

Order Une fois que le Client a reçu suffisamment d'*endorsement* avec des résultats d'exécution identique pour satisfaire la politique d'approbation, il forme la *transaction*. Cette transaction est ensuite envoyée au service d'ordonnancement. Cette transaction est composée de :

— la *payload* de la transaction avec les paramètres utilisés ;

— les métadonnées, dont les résultats d'exécutions et les identifiants du client ;

— un ensemble d'*endorsement*, pour prouver la phase précédente.

Le service d'ordonnancement ordonne l'ensemble des transactions qui lui ont été soumises selon un ordre total, ce qui constitue le consensus. En aucun cas il ne valide ou exécute les transactions. Il peut ainsi ordonner une transaction défaillante. Cette architecture permet de séparer le consensus de l'exécution et la validation. Comme le consensus est actuellement séparé, il est facilement modifiable, ainsi Hyperledger Fabric peut théoriquement supporter des consensus de type CFT (cf. A.1.1) ou des consensus BFT (cf. 1.4.4). Actuellement, Hyperledger supporte trois types de consensus, dont aucun n'est BFT (Hors, 2019). Il supporte ainsi :

— *Raft* (Ongaro & Ousterhout, 2014). Il s'agit du dernier consensus recommandé. Raft est un consensus CFT suivant un modèle *leader and follower* où les nœuds leader sont élus et ses décisions répliquées aux followers ;

— *Kafka* (G. Wang et al., 2015). Kafka est un consensus CFT également basé sur un modèle *leader and follower*. Il est aujourd'hui déprécié ;

— *Solo*. Dans ce consensus, il n'y a qu'un seul nœud d'ordonnancement. Il s'agit d'un consensus utilisé durant les tests.

Puis, le service d'ordonnancement construit des lots avec ces transactions, ce sont les blocs de la blockchain. Les blocs seront ensuite liés par cryptographie avec un système de hachage pour assurer l'ordonnancement des blocs, puis diffusés à l'ensemble des *pairs*. Cette diffusion peut être effectuée par un *gossip service*, un service de diffusion, qui peut se baser par exemple sur des algorithmes P2P.

Validate Une fois que les pairs ont récupéré un bloc, directement auprès du service d'ordonnancement ou auprès d'un protocole de diffusion, le bloc récupéré entre dans une phase de validation composée de trois étapes successives :

— la validation de la politique d'approbation des transactions ;

— une vérification des conflits de lecture-écriture. Pour chaque transaction contenue dans le bloc, la version des objets lue doit être égale avec la version de ces objets dans le registre local du nœud.

— la phase de mise à jour du ledger. Le bloc est ajouté à la blockchain du nœud local et son état est mis à jour. Les writeset sont alors appliqués et les variables associées sont mises à jour.

Dans le cas où une transaction ne satisfait pas les deux premières étapes, elle est définie comme invalide et ses effets sont ignorés dans la troisième phase.

A.2 État de l'Art des solutions agricoles basées sur la blockchain

La figure A.3 fournit une vue d'ensemble d'une chaîne alimentaire traditionnelle, y compris les contrats qui sont généralement établis entre les parties concernées (Bumblauskas et al., 2020 ; Feng et al., 2020 ; Kamilaris et al., 2019). Ces contrats sont les suivants :

— *F2D (Farmer-to-Deliver)* et *I2D (Industry-to-Deliver) :* les termes du contrat concernant, entre autres, les environnements agricoles ou les aliments transformés, l'origine des variétés de médicaments et des aliments transformés, la fertilisation et les exigences de distribution des produits (par exemple, la chaîne du froid) ;

— *D2F (Deliver-to-Farmers), D2I (Deliver-to-Industry) et D2R (Deliver-to-Farmers)* : termes du contrat concernant la distribution des produits, notamment l'entreposage de la distribution, la livraison, le destinataire prévu du produit (détaillant ou industrie) ;

— *R2D (Retailer-to-Customer : conditions contractuelles concernant les délais de vente, le prix et la qualité.* : conditions contractuelles concernant les délais de vente, le prix et la qualité.

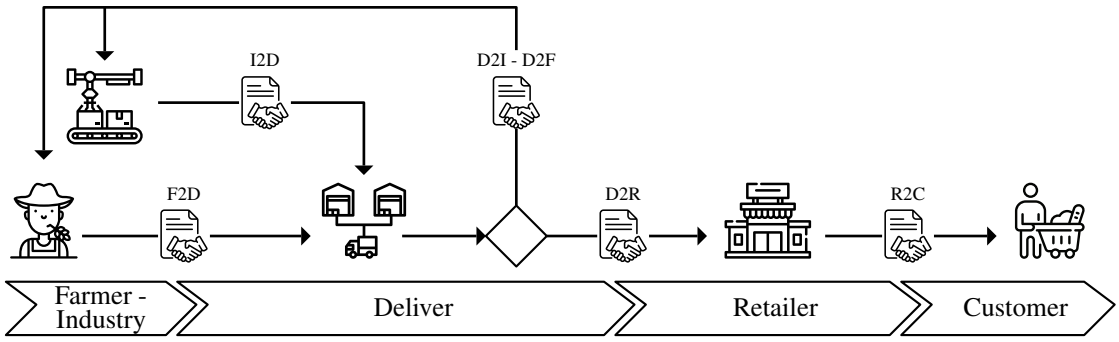


FIGURE A.3 – Chaîne de transport de produits agricoles traditionnelle

Dans le tableau A.1, nous passons en revue et classons les études de pointe qui envisagent et éventuellement mettent en œuvre la technologie blockchain à des fins d'agriculture intelligente. Les articles sont classés en fonction de cinq critères :

1. *Objectif :* nous indiquons pourquoi la blockchain est utilisée dans l'étude (par exemple, pour la traçabilité, le suivi, le commerce). Même si la traçabilité et le suivi sont parfois utilisés de manière interchangeable, une différence peut être faite. Dans un système de traçage, le flux d'informations remonte la chaîne d'approvisionnement (des consommateurs aux fournisseurs), alors que le suivi suit les informations vers l'avant (de la source aux utilisateurs finaux) ; (Laux & Hurburgh, 2010) ;
2. *Prise en charge et accent sur les smart contracts :* nous indiquons si l'étude fait usage de contrat(s) intelligent(s), et si c'est le cas, nous indiquons (i) si ces contrats sont formalisés dans le document correspondant ("F" et "N/F" dans la table A.1 étant les abréviations respectives de "Formalisé" et "Non-Formalisé") et (ii) quelles parties de la chaîne sont impliquées sur la base de la taxonomie des contrats introduite précédemment : F2D/I2D, D2F/D2I, D2R ou R2C ;

3. *Plateforme :* nous indiquons si l'étude a envisagé/utilisé une technologie/plateforme de chaîne de blocs "prête à l'emploi" tel qu'Ethereum ou Hyperledger ;
4. *Performance :* nous indiquons si l'étude a réalisé et détaillé toute évaluation de performance concernant les solutions proposées, que ce soit en termes de temps d'exécution, de latence du réseau, de débit, de sécurité ou d'autres facteurs.

Tout d'abord, on peut observer que toutes les études rapportées ont été publiées au cours des trois dernières années, ce qui confirme l'attention croissante portée à la blockchain dans le secteur agricole. En outre, la majorité des études (23 sur les 27 rapportés dans le tableau A.1) emploient la blockchain à des fins de traçabilité et/ou de suivi des aliments. Les quatre autres études utilisent la blockchain pour automatiser les contrats d'emploi temporaire entre les agriculteurs et les entrepreneurs de main-d'œuvre (Pinna & Ibba, 2019) et permettre l'échange de ressources agricoles entre les agriculteurs, les livreurs et les détaillants (Bore et al., 2020 ; Leng et al., 2018 ; Mao et al., 2019).

Deuxièmement, pratiquement toutes les études rapportées exploitent les capacités des contrats intelligents pour atteindre les objectifs susmentionnés (c'est-à-dire pour répondre aux exigences de traçabilité, de suivi et d'échange) ; 17 des 27 études se concentrent sur les interactions de marché entre les agriculteurs/industries, les livreurs et les détaillants (c'est-à-dire F-I2D, D2F-I, R2C). Sur ces 17 études, 12 étendent les installations de traçabilité, de suivi ou d'échange à l'ensemble du cycle de vie des aliments (c'est-à-dire qu'elles couvrent les interactions R2C). Il convient de noter que les études rapportées ne suivent pas nécessairement les mêmes caractéristiques du système alimentaire. En effet, certaines études telles que (Bumblauskas et al., 2020 ; Hang et al., 2020 ; J. Lin et al., 2018 ; Shyamala Devi et al., 2019 ; Surasak et al., 2019) suivent les informations environnementales de fond d'un aliment en utilisant des dispositifs de type capteur (par exemple, quantité de pesticides utilisés, évolution de la température), tandis que d'autres études suivent d'autres informations sur la chaîne d'approvisionnement telle que (i) les détails des incidents tout au long du processus de récolte (Iqbal & Butt, 2020), (ii) l'empreinte carbone au niveau de la production et du transport des aliments (Shakhbulatov et al., 2019), et (iii) l'évolution de la qualité des aliments (Carbone et al., 2018 ; George et al., 2019).

Troisièmement, si l'on examine quelles technologies blockchain ont été prises en compte dans les études rapportées (voir la colonne "Plateforme" dans le tableau A.1), Ethereum et Hyperledger Fabric sont les solutions les plus largement adoptées (les deux étant cités dans cinq études). Cela n'est pas surprenant, car elles sont toutes deux leaders en termes de parts de marché (50% des projets mis en œuvre étant hébergés sur ces plateformes) (Udokwu et al., 2018).

Enfin, on peut observer dans le tableau A.1 (voir la colonne "Performance") que moins de la moitié des études examinées ont effectué des évaluations expérimentales de leurs solutions. À notre avis, cela indique clairement que l'agriculture basée sur la blockchain n'en est qu'à ses débuts, et que l'accent est davantage mis sur les choix de conception architecturale et fonctionnelle que sur l'évaluation comparative des performances. Pour les études évaluant les performances de leur solution, le débit et la latence sont les paramètres les plus utilisés, considérés respectivement dans 65% et 50% de la littérature examinée). On peut observer que seul un petit nombre d'études a évalué les aspects de sécurité. La principale raison en est que la majorité des solutions proposées s'appuient sur des solutions blockchain sur étagère, dont les performances de sécurité – *qui se caractérise par le nombre de participants de confiance nécessaires pour sécuriser la blockchain* – ont été largement étudiées et décrites dans la littérature (Ali et al., 2019 ; Xiao et al., 2020). En fait, le niveau de sécurité d'une technologie blockchain donnée est directement dérivé du protocole de consensus supportant la chaîne.

Référence	**Objectif**	**Support des Smart Contract (SC) & focus**					**Plateforme**	**Performance**
		SCs	***(F-I)2D***	***D2(F-I)***	***D2R***	***R2C***		
Pinna et Ibba (2019)	Temp employ.	F	☐	☐	☐	☐	Ethereum	-
Shyamala Devi et al. (2019)	Track	N/F	☐	☐	☐	☐	Ethereum	1. Latence
Patil et al. (2018)	Track	-	☒	☐	☐	☐	N/S	-
Tse et al. (2017)	Trace	-	☒	☐	☐	☐	N/S	-
Carbone et al. (2018)	Track	N/F	☒	☐	☐	☐	Hyperledger	-
Hang et al. (2020)	Track	F	☒	☐	☐	☐	Hyperledger (v1.4.3)	1. Débit 2. Latence
Y.-P. Lin et al. (2017)	Trace & Track	N/F	☒	☐	☐	☐	-	-
Lucena et al. (2018)	Track	N/F	☒	☒	☐	☐	Hyperledger	-
Mao et al. (2019)	Trade	N/F	☒	☒	☒	☐	Own (FTS-CON)	1. Temps d'exéc. 2. Bénéfice, 3. Sécurité
Tian (2017)	Trace	N/F	☒	☒	☒	☐	Ethereum	-
Bore et al. (2020)	Trade	N/F	☒	☒	☒	☐	Hyperledger	1. Débit 2. Latence
Stefanova et Salampasis (2019)	Trace	N/F	☒	☒	☒	☐	Hyperledger	-
Leng et al. (2018)	Trade	N/F	☒	☒	☒	☐	N/S	1. Débit 2. Latence
M. V. Kumar et al. (2017)	Trace & Track	N/F	☒	☒	☒	☐	-	-
Iqbal et Butt (2020)	Track	N/F	☒	☒	☒	☐	-	1. ZigBee-related
George et al. (2019)	Trace & Track	N/F	☒	☒	☒	☒	-	-
Caro et al. (2018)	Trace & Track	N/F	☒	☒	☒	☒	Ethereum & Hyperledger	1. Débit 2. Latence 3. CPU
Surasak et al. (2019)	Trace & Track	N/F	☒	☒	☒	☒	SQL-based	-
Bumblauskas et al. (2020)	Trace & Track	N/F	☒	☒	☒	☒	Hyperledger	-
Malik et al. (2018)	Trace	F	☒	☒	☒	☒	Hyperledger	1. Temps
Hua et al. (2018)	Trace & Track	F	☒	☒	☒	☒	N/S	-
Shakhbulatov et al. (2019)	Track	F	☒	☒	☒	☒	Raft-like consensus	1. Débit 2. Temps
J. Lin et al. (2018)	Trace	N/F	☒	☒	☒	☒	-	-
Balakrishna Reddy et Ratna Kumar (2020)	Trace & Track	N/F	☒	☒	☒	☒	-	-
C. Xie et al. (2017)	Track	F	☒	☒	☒	☒	Ethereum (v1.9)	1. Débit
Papa (2017/06)	Trace	N/F	☒	☒	☒	☒	-	-
Awan et al. (2019)	Trace & Track	N/F	☒	☒	☒	☒	N/S	1. Débit

TABLEAU A.1 – État des lieux des initiatives de *Smart Farming*

Références

Adams, S. (2019, septembre). *Boss Recommends Blockchain.* Consulté le 2022-07-28, sur `http://dilbert.com/strip/2019-09-29`

Al-Bassam, M., Sonnino, A., Bano, S., Hrycyszyn, D., & Danezis, G. (2018). Chainspace : A Sharded Smart Contracts Platform. In *Proceedings 2018 Network and Distributed System Security Symposium* (Vol. abs/1708.03778). San Diego, CA : Internet Society. Consulté le 2022-09-09, sur `https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_09-2_Al-Bassam_paper.pdf` doi: doi:10.14722/ndss.2018.23241

Albert, E., Correas, J., Gordillo, P., Román-Díez, G., & Rubio, A. (2019, décembre). GASOL : Gas Analysis and Optimization for Ethereum Smart Contracts. *arXiv :1912.11929 [cs]*. Consulté le 2020-02-12, sur `http://arxiv.org/abs/1912.11929` (arXiv : 1912.11929)

Ali, M. S., Vecchio, M., Pincheira, M., Dolui, K., Antonelli, F., & Rehmani, M. H. (2019, Secondquarter). Applications of blockchains in the internet of things : A comprehensive survey. *IEEE Communications Surveys & Tutorials*, *21*(2), 1676-1717. doi: doi:10.1109/COMST.2018.2886932

Almakhour, M., Wehby, A., Sliman, L., Samhat, A. E., & Mellouk, A. (2021, April). Smart contract based solution for secure distributed sdn. In *2021 11th ifip international conference on new technologies, mobility and security (ntms)* (p. 1-6). doi: doi:10.1109/NTMS49979.2021.9432647

Anceaume, E., Del Pozzo, A., Ludinard, R., Potop-Butucaru, M., & Tucci-Piergiovanni, S. (2019, juin). Blockchain Abstract Data Type. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures* (pp. 349–358). Phoenix AZ USA : ACM. Consulté le 2022-08-03, sur `https://dl.acm.org/doi/10.1145/3323165.3323183` doi: doi:10.1145/3323165.3323183

Andoni, M., Robu, V., Flynn, D., Abram, S., Geach, D., Jenkins, D., ... Peacock, A. (2019, février). Blockchain technology in the energy sector : A systematic review of challenges and opportunities. *Renewable and Sustainable Energy Reviews*, *100*, 143–174. Consulté le 2022-09-01, sur `https://www.sciencedirect.com/science/article/pii/S1364032118307184` doi: doi:10.1016/j.rser.2018.10.014

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., ... Yellick, J. (2018). Hyperledger Fabric : A Distributed Operating System for Permissioned Blockchains. *Proceedings of the Thirteenth EuroSys Conference on - EuroSys '18*, 1–15. Consulté le 2018-09-28, sur `http://arxiv.org/abs/1801.10228` (arXiv : 1801.10228) doi: doi:10.1145/3190508.3190538

Angelis, S. D., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., & Sassone, V. (2018, January). Pbft vs proof-of-authority : applying the cap theorem to permissioned blockchain. In *Italian conference on cyber security (06/02/18)* (p. 11). Consulté sur `https://eprints.soton.ac.uk/415083/`

Aponte-Novoa, F. A., Orozco, A. L. S., Villanueva-Polanco, R., & Wightman, P. (2021). The

51% Attack on Blockchains : A Mining Behavior Study. *IEEE Access*, *9*, 140549–140564. (Conference Name : IEEE Access) doi: doi:10.1109/ACCESS.2021.3119291

Arora, S., Jain, A., & Gujar, S. (2020). Ashwachain : A fast, scalable and strategy-proof committee-based blockchain protocol. In *Workshop on game theory in blockchain at wine* (Vol. 2020, p. 9).

Asheralieva, A., & Niyato, D. (2021, mars). Bayesian Reinforcement Learning and Bayesian Deep Learning for Blockchains With Mobile Edge Computing. *IEEE Transactions on Cognitive Communications and Networking*, *7*(1), 319–335. (Conference Name : IEEE Transactions on Cognitive Communications and Networking) doi: doi:10.1109/TCCN.2020.2994366

Awan, S. H., Ahmed, S., Safwan, N., Najam, Z., Hashim, M. Z., & Safdar, T. (2019). Role of internet of things (iot) with blockchain technology for the development of smart farming. *Journal of Mechanics of Continua and Mathematical Sciences*, *14*(5), 170–188. doi: doi:http ://doi.org/10.26782/jmcms.2019.10.00014

Back, A. (2002). Hashcash - A Denial of Service Counter-Measure. , 10.

Baird, L., Harmon, M., & Madsen, P. (2018). Hedera : A governing council & public hashgraph network. *The trust layer of the internet, whitepaper*, *1*, 1–97.

Balakrishna Reddy, G., & Ratna Kumar, K. (2020). Quality improvement in organic food supply chain using blockchain technology. In B. Deepak, D. Parhi, & P. C. Jena (Eds.), *Innovative product design and intelligent manufacturing systems* (pp. 887–896). Singapore : Springer Singapore. doi: doi:10.1007/978-981-15-2696-1_86

Baliga, A., Solanki, N., Verekar, S., Pednekar, A., Kamat, P., & Chatterjee, S. (2018, juin). Performance Characterization of Hyperledger Fabric. In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)* (pp. 65–74). doi: doi:10.1109/CVCBT.2018.00013

Balouek, D., Carpen Amarie, A., Charrier, G., Desprez, F., Jeannot, E., Jeanvoine, E., ... Sarzyniec, L. (2013). Adding virtualization capabilities to the Grid'5000 testbed. In I. I. Ivanov, M. van Sinderen, F. Leymann, & T. Shan (Eds.), *Cloud computing and services science* (Vol. 367, p. 3-20). Springer International Publishing. doi: doi:10.1007/978-3-319-04519-1_1

Bamakan, S. M. H., Motavali, A., & Babaei Bondarti, A. (2020, septembre). A survey of blockchain consensus algorithms performance evaluation criteria. *Expert Systems with Applications*, *154*, 113385. Consulté le 2020-05-05, sur `https://linkinghub.elsevier.com/retrieve/pii/S0957417420302098` doi: doi:10.1016/j.eswa.2020.113385

Basescu, C., Nowlan, M. F., Nikitin, K., Faleiro, J. M., & Ford, B. (2018, mai). Crux : Locality-Preserving Distributed Services. *arXiv :1405.0637 [cs]*. Consulté le 2020-03-09, sur `http://arxiv.org/abs/1405.0637` (arXiv : 1405.0637)

Bellare, M., & Rogaway, P. (1993). Random oracles are practical : a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security - CCS '93* (pp. 62–73). Fairfax, Virginia, United States : ACM Press. Consulté le 2022-07-11, sur `http://portal.acm.org/citation.cfm?doid=168588.168596` doi: doi:10.1145/168588.168596

Bellini, E., Iraqi, Y., & Damiani, E. (2020). Blockchain-based distributed trust and reputation management systems : A survey. *IEEE Access*, *8*, 21127-21151. doi: doi:10.1109/ACCESS.2020.2969820

Belotti, M., Božić, N., Pujolle, G., & Secci, S. (2019, Fourthquarter). A vademecum on blockchain technologies : When, which, and how. *IEEE Communications Surveys & Tutorials*, *21*(4), 3796-3838. doi: doi:10.1109/COMST.2019.2928178

Benahmed, S., Pidikseev, I., Hussain, R., Lee, J., Kazmi, S. A., Oracevic, A., & Hussain, F. (2019, septembre). A Comparative Analysis of Distributed Ledger Technologies for Smart

Contract Development. In *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)* (pp. 1–6). (ISSN : 2166-9589) doi: doi:10.1109/PIMRC.2019.8904256

Berrang, P., von Styp-Rekowsky, P., Wissfeld, M., França, B., & Trinkler, R. (2019, June). Albatross – an optimistic consensus algorithm. In *2019 crypto valley conference on blockchain technology (cvcbt)* (p. 39-42). doi: doi:10.1109/CVCBT.2019.000-1

Bez, M., Fornari, G., & Vardanega, T. (2019, avril). The scalability challenge of ethereum : An initial quantitative analysis. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)* (pp. 167–176). San Francisco East Bay, CA, USA : IEEE. Consulté le 2020-10-08, sur `https://ieeexplore.ieee.org/document/8705874/` doi: doi:10.1109/SOSE.2019.00031

Birch, D., Brown, R. G., & Parulava, S. (2016). Towards ambient accountability in financial services : Shared ledgers, translucent transactions and the technological legacy of the great financial crisis. *Journal of Payments Strategy & Systems*, *10*(2), 118–131. (Publisher : Henry Stewart Publications)

Bitcoinity.org. (2022). *Bitcoin trading volume.* Consulté le 2022-09-15, sur `https://data.bitcoinity.org/`

Blank, J., & Deb, K. (2020). Pymoo : Multi-objective optimization in python. *IEEE Access*, *8*, 89497-89509. doi: doi:10.1109/ACCESS.2020.2990567

Blockchain.com. (2022). *hash-rate.* Consulté le 2022-08-03, sur `https://www.blockchain.com/charts/hash-rate`

Boneh, D., & Shoup, V. (2020). A graduate course in applied cryptography. *Draft 0.5*.

Bore, N., Kinai, A., Waweru, P., Wambugu, I., Mutahi, J., Kemunto, E., ... Weldemariam, K. (2020, May). Agws : Blockchain-enabled small-scale farm digitization. In *2020 ieee international conference on blockchain and cryptocurrency (icbc)* (p. 1-9). doi: doi:10.1109/ICBC48266.2020.9169450

Bretto, A., Faisant, A., & Hennecart, F. (2012). *Éléments de théorie des graphes.* Paris Berlin Heidelberg [etc.] : Springer Paris. doi: doi:10.1007/978-2-8178-0281-7

Brewer, E. A. (2000). Towards robust distributed systems (abstract). In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing - PODC '00* (p. 7). Portland, Oregon, United States : ACM Press. Consulté le 2020-10-02, sur `http://portal.acm.org/citation.cfm?doid=343477.343502` doi: doi:10.1145/343477.343502

Buchman, E., Kwon, J., & Milosevic, Z. (2018). The latest gossip on bft consensus. doi: doi:10.48550/ARXIV.1807.04938

Bumblauskas, D., Mann, A., Dugan, B., & Rittmer, J. (2020). A blockchain use case in food distribution : Do you know where your food has been ? *International Journal of Information Management*, *52*, 102008. Consulté sur `https://www.sciencedirect.com/science/article/pii/S026840121930461X` doi: doi:https ://-doi.org/10.1016/j.ijinfomgt.2019.09.004

Buterin, V. (2014). Ethereum : A Next-Generation Smart Contract and Decentralized Application Platform. , 36. Consulté sur `https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf`

Buterin, V. (2020, mar). *Ethereum Sharding Research Compendium.* Consulté le 2022-09-09, sur `https://notes.ethereum.org/@serenity/H1PGqDhpm?type=view`

Cachin, C., Guerraoui, R., & Rodrigues, L. (2011). *Introduction to Reliable and Secure Distributed Programming.* Berlin, Heidelberg : Springer Berlin Heidelberg. Consulté le 2022-07-17, sur `http://link.springer.com/10.1007/978-3-642-15260-3` doi: doi:10.1007/978-3-642-15260-3

Caldarelli, G. (2020). Understanding the blockchain oracle problem : A call for action. *Information*, *11*(11), 509. doi: doi:10.3390/info11110509

Cao, L. (2022, May). Decentralized ai : Edge intelligence and smart blockchain, metaverse, web3, and desci. *IEEE Intelligent Systems*, *37*(3), 6-19. doi: doi:10.1109/MIS.2022.3181504

Carbone, A., Davcev, D., Kocarev, L., Mitresk, K., & Stankovski, V. (2018, février). Blockchain based distributed cloud fog platform for IoT supply chain management. In *Eighth international conference on advances in computing, electronics and electrical technology - CEET 2018*. Institute of Research Engineers and Doctors. Consulté sur `https://doi.org/10.15224/978-1-63248-144-3-37` doi: doi:10.15224/978-1-63248-144-3-37

Caro, M. P., Ali, M. S., Vecchio, M., & Giaffreda, R. (2018, May). Blockchain-based traceability in agri-food supply chain management : A practical implementation. In *2018 iot vertical and topical summit on agriculture - tuscany (iot tuscany)* (p. 1-4). doi: doi:10.1109/IOT-TUSCANY.2018.8373021

Castro, M., & Liskov, B. (1999, février). Practical byzantine fault tolerance. In *Proceedings of the third symposium on operating systems design and implementation* (p. 173–186). USA : USENIX Association.

Cerioli, M. R., Faria, L., Ferreira, T. O., Martinhon, C. A. J., Protti, F., & Reed, B. (2008, juin). Partition into cliques for cubic graphs : Planar case, complexity and approximation. *Discrete Applied Mathematics*, *156*(12), 2270–2278. Consulté le 2022-05-06, sur `https://www.sciencedirect.com/science/article/pii/S0166218X07004647` doi: doi:10.1016/j.dam.2007.10.015

Churyumov, A. (2016). Byteball : A decentralized system for storage and transfer of value. *Obyte [Online resource]*. Consulté sur `http://bitomer.com/byteball-web-origin/Byteball.pdf`

Clarke, I., Sandberg, O., Wiley, B., & Hong, T. W. (2001). Freenet : A Distributed Anonymous Information Storage and Retrieval System. In G. Goos, J. Hartmanis, J. van Leeuwen, & H. Federrath (Eds.), *Designing Privacy Enhancing Technologies* (Vol. 2009, pp. 46–66). Berlin, Heidelberg : Springer Berlin Heidelberg. Consulté le 2022-08-02, sur `http://link.springer.com/10.1007/3-540-44702-4_4` (Series Title : Lecture Notes in Computer Science) doi: doi:10.1007/3-540-44702-4_4

Coelho, I. M., Coelho, V. N., Araujo, R. P., Yong Qiang, W., & Rhodes, B. D. (2020, août). Challenges of PBFT-Inspired Consensus for Blockchain and Enhancements over Neo dBFT. *Future Internet*, *12*(8), 129. Consulté le 2022-08-05, sur `https://www.mdpi.com/1999-5903/12/8/129` (Number : 8 Publisher : Multidisciplinary Digital Publishing Institute) doi: doi:10.3390/fi12080129

Cohen, B. (2003). Incentives build robustness in bittorrent. In *Workshop on economics of peer-to-peer systems* (Vol. 6, pp. 68–72).

Coinbase.com. (2022). *Bitcoin (BTC / EUR) Prix, graphiques et actualités | Coinbase.* Consulté le 2022-07-28, sur `https://www.coinbase.com/fr/price/bitcoin`

Consensys. (2021, novembre). *Quorum.* ConsenSys Software. Consulté le 2021-11-18, sur `https://github.com/ConsenSys/quorum` (original-date : 2016-11-14T05 :42 :57Z)

Corbett, J. C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J. J., ... Woodford, D. (2013, aug). Spanner : Google's globally distributed database. *ACM Trans. Comput. Syst.*, *31*(3). Consulté sur `https://doi.org/10.1145/2491245` doi: doi:10.1145/2491245

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2011). *Distributed systems : Concepts and design.* Pearson Education. Consulté sur `https://books.google.fr/books?id=3ZouAAAAQBAJ`

Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A., ... Wattenhofer, R. (2016).

On Scaling Decentralized Blockchains. In J. Clark, S. Meiklejohn, P. Y. Ryan, D. Wallach, M. Brenner, & K. Rohloff (Eds.), *Financial Cryptography and Data Security* (pp. 106–125). Berlin, Heidelberg : Springer. doi: doi:10.1007/978-3-662-53357-4_8

Dai, Y., Xu, D., Maharjan, S., Chen, Z., He, Q., & Zhang, Y. (2019, mai). Blockchain and Deep Reinforcement Learning Empowered Intelligent 5G Beyond. *IEEE Network*, *33*(3), 10–17. (Conference Name : IEEE Network) doi: doi:10.1109/MNET.2019.1800376

Dang, Q. H. (2015, juillet). *Secure Hash Standard* (Rapport technique N° NIST FIPS 180-4). National Institute of Standards and Technology. Consulté le 2022-07-11, sur `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf` doi: doi:10.6028/NIST.FIPS.180-4

Daniel, E., & Tschorsch, F. (2022, Firstquarter). Ipfs and friends : A qualitative comparison of next generation peer-to-peer data networks. *IEEE Communications Surveys & Tutorials*, *24*(1), 31-52. doi: doi:10.1109/COMST.2022.3143147

Decker, C., & Wattenhofer, R. (2013, septembre). Information propagation in the Bitcoin network. In *IEEE P2P 2013 Proceedings* (pp. 1–10). Trento, Italy : IEEE. Consulté le 2020-01-06, sur `http://ieeexplore.ieee.org/document/6688704/` doi: doi:10.1109/P2P.2013.6688704

Dehez-Clementi, M., Deneuville, J., Lacan, J., Asghar, H., & Kaafar, D. (2020). Who let the dogs out : anonymous but auditable communications using group signature schemes with distributed opening. In J. Garcia-Alfaro, G. Navarro-Arribas, & J. Herrera-Joancomarti (Eds.), *Data privacy management, cryptocurrencies and blockchain technology* (pp. 437–446). United States : Springer, Springer Nature. (15th International Workshop on Data Privacy Management, DPM 2020 and 4th International Workshop on Cryptocurrencies and Blockchain Technology, CBT 2020 held in conjunction with 25th European Symposium on Research in Computer Security, ESORICS 2020 ; Conference date : 17-09-2020 Through 18-09-2020) doi: doi:10.1007/978-3-030-66172-4_28

Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., ... Terry, D. (1987, décembre). Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing* (pp. 1–12). New York, NY, USA : Association for Computing Machinery. Consulté le 2022-08-01, sur `https://doi.org/10.1145/41840.41841` doi: doi:10.1145/41840.41841

de Vries, A. (2018, mai). Bitcoin's Growing Energy Problem. *Joule*, *2*(5), 801–805. Consulté le 2022-07-28, sur `https://www.sciencedirect.com/science/article/pii/S2542435118301776` doi: doi:10.1016/j.joule.2018.04.016

Di Francesco Maesa, D., & Mori, P. (2020). Blockchain 3.0 applications survey. *Journal of Parallel and Distributed Computing*, *138*, 99-114. Consulté sur `https://www.sciencedirect.com/science/article/pii/S0743731519308664` doi: doi:https ://-doi.org/10.1016/j.jpdc.2019.12.019

Di Francesco Maesa, D., Mori, P., & Ricci, L. (2017). Blockchain Based Access Control. In L. Y. Chen & H. P. Reiser (Eds.), *Distributed Applications and Interoperable Systems* (Vol. 10320, pp. 206–220). Cham : Springer International Publishing. Consulté le 2021-02-22, sur `http://link.springer.com/10.1007/978-3-319-59665-5_15` (Series Title : Lecture Notes in Computer Science) doi: doi:10.1007/978-3-319-59665-5_15

Dinh, T. T. A., Liu, R., Zhang, M., Chen, G., Ooi, B. C., & Wang, J. (2018, juillet). Untangling Blockchain : A Data Processing View of Blockchain Systems. *IEEE Transactions on Knowledge and Data Engineering*, *30*(7), 1366–1385. (Conference Name : IEEE Transactions on Knowledge and Data Engineering) doi: doi:10.1109/TKDE.2017.2781227

Dinh, T. T. A., Wang, J., Chen, G., Liu, R., Ooi, B. C., & Tan, K.-L. (2017, mars). BLOCKBENCH : A Framework for Analyzing Private Blockchains. *arXiv :1703.04057 [cs]*.

Consulté le 2018-09-28, sur http://arxiv.org/abs/1703.04057 (arXiv : 1703.04057)

Divya, M., & Biradar, N. B. (2018, avril). Iota-next generation block chain. *International Journal of Engineering and Computer Science*, *7*(04), 23823-23826. Consulté sur http://ijecs.in/index.php/ijecs/article/view/4007 doi: doi:10.18535/ijecs/v7i4.05

Dong, Z., Zheng, E., Choon, Y., & Zomaya, A. Y. (2019, juillet). DAGBENCH : A Performance Evaluation Framework for DAG Distributed Ledgers. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)* (pp. 264–271). (ISSN : 2159-6190) doi: doi:10.1109/CLOUD.2019.00053

Douceur, J. J. (2002, January). The sybil attack. In *Proceedings of 1st international workshop on peer-to-peer systems (iptps)* (Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS) éd.). Consulté sur https://www.microsoft.com/en-us/research/publication/the-sybil-attack/

Dujak, D., & Sajter, D. (2019). Blockchain applications in supply chain. In A. Kawa & A. Maryniak (Eds.), *Smart supply network* (pp. 21–46). Cham : Springer International Publishing. Consulté sur https://doi.org/10.1007/978-3-319-91668-2_2 doi: doi:10.1007/978-3-319-91668-2_2

Dwork, C., & Naor, M. (1993). Pricing via Processing or Combatting Junk Mail. In E. F. Brickell (Ed.), *Advances in Cryptology — CRYPTO' 92* (pp. 139–147). Berlin, Heidelberg : Springer. doi: doi:10.1007/3-540-48071-4_10

Edelman, G. (2022). *The Father of Web3 Wants You to Trust Less.* Consulté le 2022-09-28, sur https://www.wired.com/story/web3-gavin-wood-interview/ (Section : tags)

Eng Keong Lua, Crowcroft, J., Pias, M., Sharma, R., & Lim, S. (2005). A survey and comparison of peer-to-peer overlay network schemes. *IEEE Commun. Surv. Tutorials*, *7*(2), 72–93. Consulté le 2020-03-26, sur http://ieeexplore.ieee.org/document/1610546/ doi: doi:10.1109/COMST.2005.1610546

Esposito, C., Ficco, M., & Gupta, B. B. (2021, mars). Blockchain-based authentication and authorization for smart city applications. *Information Processing & Management*, *58*(2), 102468. Consulté le 2021-04-20, sur https://www.sciencedirect.com/science/article/pii/S0306457320309584 doi: doi:10.1016/j.ipm.2020.102468

Etherscan.io. (2022, septembre). *Ethereum charts & statistics.* Consulté le 2022-09-12, sur http://etherscan.io/charts

Fan, C., Ghaemi, S., Khazaei, H., & Musilek, P. (2020). Performance Evaluation of Blockchain Systems : A Systematic Survey. *IEEE Access*, *8*, 126927–126950. (Conference Name : IEEE Access) doi: doi:10.1109/ACCESS.2020.3006078

Feng, H., Wang, X., Duan, Y., Zhang, J., & Zhang, X. (2020). Applying blockchain technology to improve agri-food traceability : A review of development methods, benefits and challenges. *Journal of Cleaner Production*, *260*, 121031. Consulté sur https://www.sciencedirect.com/science/article/pii/S0959652620310787 doi: doi:https ://-doi.org/10.1016/j.jclepro.2020.121031

Fernández-Caramés, T. M., & Fraga-Lamas, P. (2018). A Review on the Use of Blockchain for the Internet of Things. *IEEE Access*, *6*, 32979–33001. (Conference Name : IEEE Access) doi: doi:10.1109/ACCESS.2018.2842685

Fischer, M. J., Lynch, N. A., & Paterson, M. S. (1985, apr). Impossibility of distributed consensus with one faulty process. *J. ACM*, *32*(2), 374–382. Consulté sur https://doi.org/10.1145/3149.214121 doi: doi:10.1145/3149.214121

Foundation, H. (2018, oct). *Hyperledger Blockchain Performance Metrics White Paper – Hyperledger Foundation.* Consulté le 2022-09-03, sur https://www.hyperledger.org/learn/publications/blockchain-performance-metrics

Foundation, H. (2020, jan). *A Blockchain Platform for the Enterprise — hyperledger-fabricdocs master documentation.* Consulté le 2020-09-30, sur `https://hyperledger-fabric.readthedocs.io/en/release-2.0/`

Foundation, H. (2022, septembre). *Hyperledger caliper.* Hyperledger. Consulté le 2022-09-03, sur `https://github.com/hyperledger/caliper` (original-date : 2018-03-20T01 :46 :34Z)

Fullmer, D., & Morse, A. S. (2018, Dec). Analysis of difficulty control in bitcoin and proof-of-work blockchains. In *2018 ieee conference on decision and control (cdc)* (p. 5988-5992). doi: doi:10.1109/CDC.2018.8619082

Galvez, J. F., Mejuto, J., & Simal-Gandara, J. (2018). Future challenges on the use of blockchain for food traceability analysis. *TrAC Trends in Analytical Chemistry*, *107*, 222-232. Consulté sur `https://www.sciencedirect.com/science/article/pii/S0165993618301304` doi: doi:https ://doi.org/10.1016/j.trac.2018.08.011

George, R. V., Harsh, H. O., Ray, P., & Babu, A. K. (2019). Food quality traceability prototype for restaurants using blockchain and food quality data index. *Journal of Cleaner Production*, *240*, 118021. Consulté sur `https://www.sciencedirect.com/science/article/pii/S0959652619328914` doi: doi:https ://doi.org/10.1016/j.jclepro.2019.118021

Geyer, F., Kinkelin, H., Leppelsack, H., Liebald, S., Scholz, D., Carle, G., & Schupke, D. (2019, mars). Performance Perspective on Private Distributed Ledger Technologies for Industrial Networks. In *2019 International Conference on Networked Systems (NetSys)* (pp. 1–8). Munich, Germany : IEEE. Consulté le 2022-09-04, sur `https://ieeexplore.ieee.org/document/8854512/` doi: doi:10.1109/NetSys.2019.8854512

Gilad, Y., Hemo, R., Micali, S., Vlachos, G., & Zeldovich, N. (2017). Algorand : Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th symposium on operating systems principles* (p. 51–68). New York, NY, USA : Association for Computing Machinery. doi: doi:10.1145/3132747.3132757

Gilbert, S., & Lynch, N. (2002). Brewer's Conjecture and the Feasibility of Consistent Available Partition-Tolerant Web Services. In *In ACM SIGACT News* (p. 2002).

Goldberg, S., Vcelak, J., Papadopoulos, D., & Reyzin, L. (2018). Verifiable Random Functions (VRFs).

Google. (2012). *The Go Programming Language.* Consulté le 2021-03-08, sur `https://golang.org/`

Gummadi, K., Gummadi, R., Gribble, S., Ratnasamy, S., Shenker, S., & Stoica, I. (2003, août). The impact of DHT routing geometry on resilience and proximity. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications* (pp. 381–394). New York, NY, USA : Association for Computing Machinery. Consulté le 2022-04-21, sur `https://doi.org/10.1145/863955.863998` doi: doi:10.1145/863955.863998

Guo, H., Li, W., Nejad, M., & Shen, C.-C. (2019, juillet). Access Control for Electronic Health Records with Hybrid Blockchain-Edge Architecture. In *2019 IEEE International Conference on Blockchain (Blockchain)* (pp. 44–51). doi: doi:10.1109/Blockchain.2019.00015

Haber, S., & Stornetta, W. S. (1991). How to time-stamp a digital document. In A. J. Menezes & S. A. Vanstone (Eds.), *Advances in cryptology-crypto' 90* (pp. 437–455). Berlin, Heidelberg : Springer Berlin Heidelberg.

Haerder, T., & Reuter, A. (1983, décembre). Principles of transaction-oriented database recovery. *ACM Computing Surveys*, *15*(4), 287–317. Consulté le 2022-07-21, sur `https://dl.acm.org/doi/10.1145/289.291` doi: doi:10.1145/289.291

Han, R., Gramoli, V., & Xu, X. (2018, février). Evaluating Blockchains for IoT. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)* (pp.

1–5). Paris : IEEE. Consulté le 2021-09-20, sur http://ieeexplore.ieee.org/document/8328736/ doi: doi:10.1109/NTMS.2018.8328736

Han, R., Shapiro, G., Gramoli, V., & Xu, X. (2020, juin). On the performance of distributed ledgers for Internet of Things. *Internet of Things*, *10*, 100087. Consulté le 2022-09-05, sur https://www.sciencedirect.com/science/article/pii/S2542660518300416 doi: doi:10.1016/j.iot.2019.100087

Hang, L., Ullah, I., & Kim, D.-H. (2020). A secure fish farm platform based on blockchain for agriculture data integrity. *Computers and Electronics in Agriculture*, *170*, 105251. Consulté sur https://www.sciencedirect.com/science/article/pii/S016816991932006X doi: doi:https ://doi.org/10.1016/j.compag.2020.105251

Herlihy, M. P., & Wing, J. M. (1990, juillet). Linearizability : a correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems*, *12*(3), 463–492. Consulté le 2022-07-22, sur https://dl.acm.org/doi/10.1145/78969.78972 doi: doi:10.1145/78969.78972

Hjálmarsson, F. T., Hreiðarsson, G. K., Hamdaqa, M., & Hjálmtýsson, G. (2018, July). Blockchain-based e-voting system. In *2018 ieee 11th international conference on cloud computing (cloud)* (p. 983-986). doi: doi:10.1109/CLOUD.2018.00151

Hors, A. L. (2019, avril). *Demystifying hyperledger fabric ordering and decentralization.* Consulté le 2020-09-30, sur https://developer.ibm.com/articles/blockchain-hyperledger-fabric-ordering-decentralization/

Hua, J., Wang, X., Kang, M., Wang, H., & Wang, F.-Y. (2018, June). Blockchain based provenance for agricultural products : A distributed platform with duplicated and shared bookkeeping. In *2018 ieee intelligent vehicles symposium (iv)* (p. 97-101). doi: doi:10.1109/IVS.2018.8500647

Hughes, W., Russo, A., & Schneider, G. (2021, mai). MultiCall : A Transaction-batching Interpreter for Ethereum. In *Proceedings of the 3rd ACM International Symposium on Blockchain and Secure Critical Infrastructure* (pp. 25–35). Virtual Event Hong Kong : ACM. Consulté le 2022-09-06, sur https://dl.acm.org/doi/10.1145/3457337.3457839 doi: doi:10.1145/3457337.3457839

Hyperledger, F. (2022). *Introduction to hyperledger sawtooth.* Consulté le 2022-04-19, sur https://sawtooth.hyperledger.org/docs/1.2/

Iqbal, R., & Butt, T. A. (2020, 01 Sep). Safe farming as a service of blockchain-based supply chain management for improved transparency. *Cluster Computing*, *23*(3), 2139-2150. Consulté sur https://doi.org/10.1007/s10586-020-03092-4 doi: doi:10.1007/s10586-020-03092-4

Jeong, S., Maurer, B., & Swartz, L. (2017, 05). Dogecoin. In *Paid : Tales of Dongles, Checks, and Other Money Stuff.* The MIT Press. Consulté sur https://doi.org/10.7551/mitpress/9780262035750.003.0006 doi: doi:10.7551/mitpress/9780262035750.003.0006

Juma, H., Shaalan, K., & Kamel, I. (2019). A survey on using blockchain in trade supply chain solutions. *IEEE Access*, *7*, 184115-184132. doi: doi:10.1109/ACCESS.2019.2960542

Kamilaris, A., Fonts, A., & Prenafeta-Boldu, F. X. (2019). The rise of blockchain technology in agriculture and food supply chains. *Trends in Food Science & Technology*, *91*, 640-652. Consulté sur https://www.sciencedirect.com/science/article/pii/S0924224418303686 doi: doi:https ://doi.org/10.1016/j.tifs.2019.07.034

Kan, G. (2002). Gnutella. In D. Schoder, K. Fischbach, & R. Teichmann (Eds.), *Peer-to-Peer : Ökonomische, technologische und juristische Perspektiven* (pp. 189–199). Berlin, Heidelberg : Springer. Consulté le 2022-08-02, sur https://doi.org/10.1007/978-3-642-56176-4_11 doi: doi:10.1007/978-3-642-56176-4_11

Kar, A. K., & Navin, L. (2021). Diffusion of blockchain in insurance industry : An analysis through the review of academic and trade literature. *Telematics and Informatics*, *58*, 101532. Consulté sur https://www.sciencedirect.com/science/article/pii/S073658532030191X doi: doi:https ://doi.org/10.1016/j.tele.2020.101532

Karr, A. F. (1984). Stochastic processes (sheldon m. ross). *SIAM Review*, *26*(3), 448.

Kaur, M., Khan, M. Z., Gupta, S., Noorwali, A., Chakraborty, C., & Pani, S. K. (2021). MBCP : Performance Analysis of Large Scale Mainstream Blockchain Consensus Protocols. *IEEE Access*, *9*, 80931–80944. (Conference Name : IEEE Access) doi: doi:10.1109/ACCESS.2021.3085187

Kencana Ramli, C. D. P., Nielson, H. R., & Nielson, F. (2012). The Logic of XACML. In F. Arbab & P. C. Ölveczky (Eds.), *Formal Aspects of Component Software* (pp. 205–222). Berlin, Heidelberg : Springer Berlin Heidelberg.

Khan, D., Jung, L. T., Ahmed Hashmani, M., & Waqas, A. (2020, janvier). A Critical Review of Blockchain Consensus Model. In *2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)* (pp. 1–6). doi: doi:10.1109/iCoMET48670.2020.9074107

Kiffer, L., Salman, A., Levin, D., Mislove, A., & Nita-Rotaru, C. (2021). Under the Hood of the Ethereum Gossip Protocol. In N. Borisov & C. Diaz (Eds.), *Financial Cryptography and Data Security* (pp. 437–456). Berlin, Heidelberg : Springer Berlin Heidelberg.

King, S., & Nadal, S. (2012). Ppcoin : Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, *19*(1).

Koens, T., & Poll, E. (2018). What Blockchain Alternative Do You Need ? In J. Garcia-Alfaro, J. Herrera-Joancomartí, G. Livraga, & R. Rios (Eds.), *Data Privacy Management, Cryptocurrencies and Blockchain Technology* (Vol. 11025, pp. 113–129). Cham : Springer International Publishing. Consulté le 2022-07-28, sur http://link.springer.com/10.1007/978-3-030-00305-0_9 (Series Title : Lecture Notes in Computer Science) doi: doi:10.1007/978-3-030-00305-0_9

Kogias, E. K., Jovanovic, P., Gailly, N., Khoffi, I., Gasser, L., & Ford, B. (2016, août). Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th usenix security symposium (usenix security 16)* (pp. 279–296). Austin, TX : USENIX Association.

Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., & Ford, B. (2018, mai). OmniLedger : A Secure, Scale-Out, Decentralized Ledger via Sharding. In *2018 IEEE Symposium on Security and Privacy (SP)* (pp. 583–598). (ISSN : 2375-1207) doi: doi:10.1109/SP.2018.000-5

Kolasa, Y., Bastogne, T., Georges, J.-P., & Kubler, S. (2020, juillet). Quality-by-design-engineered pBFT consensus configuration for medical device development. In *EMBC 2020 - 42nd Engineering in Medicine and Biology Conference*. Montreal, Canada. Consulté le 2021-02-12, sur https://hal.archives-ouvertes.fr/hal-02568428

KotET. (2016, feb). *Post-Mortem Investigation.* Consulté le 2022-07-28, sur https://www.kingoftheether.com/postmortem.html

Kumar, M. V., Iyengar, N., et al. (2017, 11). A framework for blockchain technology in rice supply chain management. *Adv. Sci. Technol. Lett*, *146*, 125–130. doi: doi:10.14257/astl.2017.146.22

Kumar, R. L., Khan, F., Kadry, S., & Rho, S. (2022, août). A Survey on blockchain for industrial Internet of Things. *Alexandria Engineering Journal*, *61*(8), 6001–6022. Consulté le 2022-07-05, sur https://www.sciencedirect.com/science/article/pii/S1110016821007560 doi: doi:10.1016/j.aej.2021.11.023

Lamport, L. (1977, mars). Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering*, *SE-3*(2), 125–143. Consulté le 2022-07-22, sur `http://ieeexplore.ieee.org/document/1702415/` doi: doi:10.1109/TSE.1977.229904

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. In *Concurrency : The works of leslie lamport* (p. 179–196). New York, NY, USA : Association for Computing Machinery. doi: doi:10.1145/3335772.3335934

Lamport, L. (1998, mai). The part-time parliament. *ACM Trans. Comput. Syst.*, *16*(2), 133–169. Consulté le 2020-07-24, sur `https://doi.org/10.1145/279227.279229` doi: doi:10.1145/279227.279229

Lamport, L. (2001, December). Paxos made simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, 51-58. Consulté sur `https://www.microsoft.com/en-us/research/publication/paxos-made-simple/`

Lamport, L. (2005, March). Generalized consensus and paxos. (MSR-TR-2005-33), 60. Consulté sur `https://www.microsoft.com/en-us/research/publication/generalized-consensus-and-paxos/`

Lamport, L. (2006, 01 Oct). Fast paxos. *Distributed Computing*, *19*(2), 79-103. Consulté sur `https://doi.org/10.1007/s00446-006-0005-x` doi: doi:10.1007/s00446-006-0005-x

Lamport, L., Shostak, R., & Pease, M. (1982). The byzantine generals problem. In *Concurrency : The works of leslie lamport* (p. 203–226). New York, NY, USA : Association for Computing Machinery. Consulté sur `https://doi.org/10.1145/3335772.3335936`

Lange, F. (2015). *Ethereum Wire Protocol (ETH).* Consulté le 2022-08-01, sur `https://github.com/ethereum/devp2p/blob/master/caps/eth.md`

Laux, C., & Hurburgh, C. R. (2010). Using quality management systems for food traceability. *Journal of Industrial Technology*, *26*(3).

Leduc, G., Kubler, S., & Georges, J.-P. (2021). Innovative blockchain-based farming marketplace and smart contract performance evaluation. *Journal of Cleaner Production*, *306*, 127055. Consulté sur `https://www.sciencedirect.com/science/article/pii/S0959652621012749` doi: doi:https ://doi.org/10.1016/j.jclepro.2021.127055

Leduc, G., Kubler, S., & Georges, J.-P. (2022a). Adapting the number of validator according the need in a bft chain with sabine. *IFAC-PapersOnLine*, *55*(8), 64-70. Consulté sur `https://www.sciencedirect.com/science/article/pii/S2405896322010849` (6th IFAC Symposium on Telematics Applications TA 2022) doi: doi:https ://doi.org/10.1016/j.ifacol.2022.08.011

Leduc, G., Kubler, S., & Georges, J.-P. (2022b, août). Sabine : Self-Adaptive BlockchaIn coNsEnsus. In *9th International Conference on Future Internet of Things and Cloud, FiCloud 2022.* Rome, Italy. Consulté sur `https://hal.archives-ouvertes.fr/hal-03777391`

LeMahieu, C. (2018). Nano : A feeless distributed cryptocurrency network. *Nano [Online resource]*. Consulté sur `http://media.abnnewswire.net/media/cs/whitepaper/rpt/91948-whitepaper.pdf`

Leng, K., Bi, Y., Jing, L., Fu, H.-C., & Van Nieuwenhuyse, I. (2018). Research on agricultural supply chain system with double chain architecture based on blockchain technology. *Future Generation Computer Systems*, *86*, 641-649. Consulté sur `https://www.sciencedirect.com/science/article/pii/S0167739X18304527` doi: doi:https ://-doi.org/10.1016/j.future.2018.04.061

Li, P., Wang, G., Chen, X., Long, F., & Xu, W. (2020). Gosig : A scalable and high-performance byzantine consensus for consortium blockchains. In *Proceedings of the 11th acm symposium on cloud computing* (p. 223–237). New York, NY, USA : Association for Computing Machinery. doi: doi:10.1145/3419111.3421272

Lin, J., Shen, Z., Zhang, A., & Chai, Y. (2018). Blockchain and iot based food traceability for smart agriculture. In *Proceedings of the 3rd international conference on crowd science and engineering*. New York, NY, USA : Association for Computing Machinery. Consulté sur https://doi.org/10.1145/3265689.3265692 doi: doi:10.1145/3265689.3265692

Lin, Y.-P., Petway, J. R., Anthony, J., Mukhtar, H., Liao, S.-W., Chou, C.-F., & Ho, Y.-F. (2017). Blockchain : The evolutionary next step for ict e-agriculture. *Environments*, *4*(3). Consulté sur https://www.mdpi.com/2076-3298/4/3/50 doi: doi:10.3390/environments4030050

Lin, Y.-T. (2017, June). *Istanbul Byzantine Fault Tolerance · Issue #650 · ethereum/EIPs.* Consulté le 2021-03-12, sur https://github.com/ethereum/EIPs/issues/650

Liu, L., Zhou, S., Huang, H., & Zheng, Z. (2021). From Technology to Society : An Overview of Blockchain-Based DAO. *IEEE Open Journal of the Computer Society*, *2*, 204–215. (Conference Name : IEEE Open Journal of the Computer Society) doi: doi:10.1109/OJCS.2021.3072661

Liu, M., Yu, F. R., Teng, Y., Leung, V. C. M., & Song, M. (2019, juin). Performance Optimization for Blockchain-Enabled Industrial Internet of Things (IIoT) Systems : A Deep Reinforcement Learning Approach. *IEEE Transactions on Industrial Informatics*, *15*(6), 3559–3570. (Conference Name : IEEE Transactions on Industrial Informatics) doi: doi:10.1109/TII.2019.2897805

Lucena, P., Binotto, A. P. D., Momo, F. d. S., & Kim, H. (2018). A case study for grain quality assurance tracking based on a blockchain business network. *arXiv preprint*. Consulté sur https://arxiv.org/abs/1803.07877 doi: doi:10.48550/ARXIV.1803.07877

Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., & Saxena, P. (2016). A secure sharding protocol for open blockchains. In *Proceedings of the 2016 acm sigsac conference on computer and communications security* (p. 17–30). New York, NY, USA : Association for Computing Machinery. Consulté sur https://doi.org/10.1145/2976749.2978389 doi: doi:10.1145/2976749.2978389

Malatras, A. (2015, septembre). State-of-the-art survey on P2P overlay networks in pervasive computing environments. *Journal of Network and Computer Applications*, *55*, 1–23. Consulté le 2020-03-27, sur https://linkinghub.elsevier.com/retrieve/pii/S1084804515000879 doi: doi:10.1016/j.jnca.2015.04.014

Malik, S., Kanhere, S. S., & Jurdak, R. (2018, Nov). Productchain : Scalable blockchain framework to support provenance in supply chains. In *2018 ieee 17th international symposium on network computing and applications (nca)* (p. 1-10). doi: doi:10.1109/NCA.2018.8548322

Manimuthu, A., Venkatesh, V. G., Shi, Y., Sreedharan, V. R., & Koh, S. C. L. (2022, janvier). Design and development of automobile assembly model using federated artificial intelligence with smart contract. *International Journal of Production Research*, *60*(1), 111–135. Consulté le 2022-09-01, sur https://doi.org/10.1080/00207543.2021.1988750 (Publisher : Taylor & Francis _eprint : https ://doi.org/10.1080/00207543.2021.1988750) doi: doi:10.1080/00207543.2021.1988750

Mao, D., Hao, Z., Wang, F., & Li, H. (2019, 01 Apr). Novel automatic food trading system using consortium blockchain. *Arabian Journal for Science and Engineering*, *44*(4), 3439-3455. Consulté sur https://doi.org/10.1007/s13369-018-3537-z doi: doi:10.1007/s13369-018-3537-z

Maymounkov, P., & Mazières, D. (2002). Kademlia : A Peer-to-Peer Information System Based on the XOR Metric. In G. Goos, J. Hartmanis, J. van Leeuwen, P. Druschel, F. Kaashoek, & A. Rowstron (Eds.), *Peer-to-Peer Systems* (Vol. 2429, pp. 53–65). Berlin, Heidelberg : Springer Berlin Heidelberg. Consulté le 2020-03-26, sur http://link.springer.com/10.1007/3-540-45748-8_5 (Series Title : Lecture Notes in Computer Science) doi:

doi:10.1007/3-540-45748-8_5

Merkle, R. C. (1988). A Digital Signature Based on a Conventional Encryption Function. In C. Pomerance (Ed.), *Advances in Cryptology — CRYPTO '87* (pp. 369–378). Berlin, Heidelberg : Springer. doi: doi:10.1007/3-540-48184-2_32

Miglani, A., & Kumar, N. (2021, octobre). Blockchain management and machine learning adaptation for iot environment in 5g and beyond networks : A systematic review. *Computer Communications*, *178*, 37-63. Consulté sur https://www.sciencedirect.com/science/article/pii/S0140366421002632 doi: doi:https ://-doi.org/10.1016/j.comcom.2021.07.009

Moniz, H. (2020). The istanbul bft consensus algorithm. doi: doi:10.48550/ARXIV.2002.03613

Muthe, K. B., Sharma, K., & Sri, K. E. N. (2020, novembre). A Blockchain Based Decentralized Computing And NFT Infrastructure For Game Networks. In *2020 Second International Conference on Blockchain Computing and Applications (BCCA)* (pp. 73–77). doi: doi:10.1109/BCCA50787.2020.9274085

Nakamoto, S. (2008, 31 10). Bitcoin : A peer-to-peer electronic cash system. *Decentralized Business Review*.

Nandi, M., Bhattacharjee, R. K., Jha, A., & Barbhuiya, F. A. (2020, février). A secured land registration framework on Blockchain. In *2020 Third ISEA Conference on Security and Privacy (ISEA-ISAP)* (pp. 130–138). doi: doi:10.1109/ISEA-ISAP49340.2020.235011

Nexo. (2021). *Processing time of cryptocurrency deposits (blockchain confirmations).* Consulté le 2022-08-03, sur https://support.nexo.io/hc/en-us/articles/360010650140-Processing-time-of-cryptocurrency-deposits-blockchain-confirmations-

Nguyen, D. C., Pathirana, P. N., Ding, M., & Seneviratne, A. (2020, décembre). Privacy-Preserved Task Offloading in Mobile Blockchain with Deep Reinforcement Learning. *IEEE Transactions on Network and Service Management*, *17*(4), 2536–2549. Consulté le 2022-09-07, sur http://arxiv.org/abs/1908.07467 (arXiv :1908.07467 [eess]) doi: doi:10.1109/TNSM.2020.3010967

Nguyen, G.-T., & Kim, K. (2018, février). A Survey about Consensus Algorithms Used in Blockchain. *Journal of Information Processing Systems*, *14*(1), 101–128. Consulté le 2022-09-04, sur https://doi.org/10.3745/JIPS.01.0024 doi: doi:10.3745/JIPS.01.0024

Ongaro, D., & Ousterhout, J. (2014). In Search of an Understandable Consensus Algorithm. In (pp. 305–319). Consulté le 2022-08-04, sur https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro

O'Reilly, T. (2005, septembre). *What Is Web 2.0.* Consulté le 2022-09-28, sur https://www.oreilly.com/pub/a//web2/archive/what-is-web-20.html

Papa, S. F. (2017/06). Use of blockchain technology in agribusiness : Transparency and monitoring in agricultural trade. In *Proceedings of the 2017 international conference on management science and management innovation (msmi 2017)* (p. 38-40). Atlantis Press. Consulté sur https://doi.org/10.2991/msmi-17.2017.9 doi: doi:https ://doi.org/10.2991/msmi-17.2017.9

Patarin, J., & Montreuil, A. (2006). Benes and Butterfly Schemes Revisited. In D. H. Won & S. Kim (Eds.), *Information Security and Cryptology - ICISC 2005* (Vol. 3935, pp. 92–116). Berlin, Heidelberg : Springer Berlin Heidelberg. Consulté le 2022-07-11, sur http://link.springer.com/10.1007/11734727_10 (Series Title : Lecture Notes in Computer Science) doi: doi:10.1007/11734727_10

Patil, A. S., Tama, B. A., Park, Y., & Rhee, K.-H. (2018). A Framework for Blockchain Based Secure Smart Green House Farming. In J. J. Park, V. Loia, G. Yi, & Y. Sung

(Eds.), *Advances in Computer Science and Ubiquitous Computing* (Vol. 474, pp. 1162–1167). Singapore : Springer Singapore. Consulté le 2019-10-09, sur `http://link.springer.com/10.1007/978-981-10-7605-3_185` doi: doi:10.1007/978-981-10-7605-3_185

Peng, C., Akca, S., & Rajan, A. (2019, décembre). SIF : A Framework for Solidity Contract Instrumentation and Analysis. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)* (pp. 466–473). (ISSN : 2640-0715) doi: doi:10.1109/APSEC48747.2019.00069

Perard, D., Lacan, J., Bachy, Y., & Detchart, J. (2018, July). Erasure code-based low storage blockchain node. In *2018 ieee international conference on internet of things (ithings) and ieee green computing and communications (greencom) and ieee cyber, physical and social computing (cpscom) and ieee smart data (smartdata)* (p. 1622-1627). doi: doi:10.1109/Cybermatics_2018.2018.00271

Pierro, G. A., & Rocha, H. (2019, mai). The Influence Factors on Ethereum Transaction Fees. In *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)* (pp. 24–31). Montreal, QC, Canada : IEEE. Consulté le 2020-02-06, sur `https://ieeexplore.ieee.org/document/8823908/` doi: doi:10.1109/WETSEB.2019.00010

Pinna, A., & Ibba, S. (2019). A blockchain-based decentralized system for proper handling of temporary employment contracts. In K. Arai, S. Kapoor, & R. Bhatia (Eds.), *Intelligent computing* (pp. 1231–1243). Cham : Springer International Publishing. doi: doi:10.1007/978-3-030-01177-2_88

Pongnumkul, S., Siripanpornchana, C., & Thajchayapong, S. (2017, juillet). Performance Analysis of Private Blockchain Platforms in Varying Workloads. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)* (pp. 1–6). Vancouver, BC, Canada : IEEE. Consulté le 2020-01-09, sur `http://ieeexplore.ieee.org/document/8038517/` doi: doi:10.1109/ICCCN.2017.8038517

Qi, X., Yang, Y., Zhang, Z., Jin, C., & Zhou, A. (2020). Linsbft : Linear-communication one-step bft protocol for public blockchains.
doi: doi:10.48550/ARXIV.2007.07642

Qiu, X., Liu, L., Chen, W., Hong, Z., & Zheng, Z. (2019, août). Online Deep Reinforcement Learning for Computation Offloading in Blockchain-Empowered Mobile Edge Computing. *IEEE Transactions on Vehicular Technology*, *68*(8), 8050–8062. (Conference Name : IEEE Transactions on Vehicular Technology) doi: doi:10.1109/TVT.2019.2924015

Ramiro, A., & de Queiroz, R. (2022, avril). Cypherpunk. *Internet Policy Review*, *11*(2), 1–10. Consulté le 2022-09-28, sur `https://policyreview.info/glossary/cypherpunk` doi: doi:10.14763/2022.2.1664

Ramli, C. D. P. K., Nielson, H. R., & Nielson, F. (2014, avril). The logic of XACML. *Science of Computer Programming*, *83*, 80–105. Consulté le 2021-03-29, sur `https://www.sciencedirect.com/science/article/pii/S0167642313001238` doi: doi:10.1016/j.scico.2013.05.003

Rapuano, A., Iovane, G., & Chinnici, M. (2020, Dec). A scalable blockchain based system for super resolution images manipulation. In *2020 ieee 6th international conference on dependability in sensor, cloud and big data systems and application (dependsys)* (p. 8-15). doi: doi:10.1109/DependSys51298.2020.00011

Rifaï, H., Mohammed, S., & Mellouk, A. (2011, April). A brief synthesis of qos-qoe methodologies. In *2011 10th international symposium on programming and systems* (p. 32-38). doi: doi:10.1109/ISPS.2011.5898880

Rogaway, P., & Shrimpton, T. (2004). Cryptographic Hash-Function Basics : Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and

Collision Resistance. In T. Kanade et al. (Eds.), *Fast Software Encryption* (Vol. 3017, pp. 371–388). Berlin, Heidelberg : Springer Berlin Heidelberg. Consulté le 2022-07-11, sur http://link.springer.com/10.1007/978-3-540-25937-4_24 (Series Title : Lecture Notes in Computer Science) doi: doi:10.1007/978-3-540-25937-4_24

Rouhani, S., & Deters, R. (2017, novembre). Performance analysis of ethereum transactions in private blockchain. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)* (pp. 70–74). (ISSN : 2327-0594) doi: doi:10.1109/ICSESS.2017.8342866

Sastry, K., Goldberg, D., & Kendall, G. (2005). Genetic algorithms. In E. K. Burke & G. Kendall (Eds.), *Search methodologies : Introductory tutorials in optimization and decision support techniques* (pp. 97–125). Boston, MA : Springer US. Consulté sur https://doi.org/10.1007/0-387-28356-0_4 doi: doi:10.1007/0-387-28356-0_4

Schäffer, M., di Angelo, M., & Salzer, G. (2019). Performance and scalability of private ethereum blockchains. In C. Di Ciccio et al. (Eds.), *Business process management : Blockchain and central and eastern europe forum* (pp. 103–118). Cham : Springer International Publishing. doi: doi:10.1007/978-3-030-30429-4_8

Schneider, F. B. (1990, décembre). Implementing fault-tolerant services using the state machine approach : a tutorial. *ACM Comput. Surv.*, *22*(4), 299–319. Consulté le 2020-07-08, sur http://dl.acm.org/doi/10.1145/98163.98167 doi: doi:10.1145/98163.98167

Schulz, P., Matthe, M., Klessig, H., Simsek, M., Fettweis, G., Ansari, J., ... Windisch, M. (2017, February). Latency critical iot applications in 5g : Perspective on the design of radio interface and network architecture. *IEEE Communications Magazine*, *55*(2), 70-78. doi: doi:10.1109/MCOM.2017.1600435CM

Shakhbulatov, D., Arora, A., Dong, Z., & Rojas-Cessa, R. (2019, July). Blockchain implementation for analysis of carbon footprint across food supply chain. In *2019 ieee international conference on blockchain (blockchain)* (p. 546-551). doi: doi:10.1109/Blockchain.2019.00079

Shyamala Devi, M., Suguna, R., Joshi, A. S., & Bagate, R. A. (2019). Design of iot blockchain based smart agriculture for enlightening safety and security. In A. K. Somani, S. Ramakrishna, A. Chaudhary, C. Choudhary, & B. Agarwal (Eds.), *Emerging technologies in computer engineering : Microservices in big data analytics* (pp. 7–19). Singapore : Springer Singapore. doi: doi:10.1007/978-981-13-8300-7_2

Singh, N., & Vardhan, M. (2019, avril). Distributed Ledger Technology based Property Transaction System with Support for IoT Devices. *International Journal of Cloud Applications and Computing (IJCAC)*, *9*, 60–78. doi: doi:10.4018/IJCAC.2019040104

Singh, S., Maakar, S. K., & Kumar, D. S. (2013). A Performance Analysis of DES and RSA Cryptography. , *2*(3), 6.

Stefanova, M., & Salampasis, M. (2019). Farm-to-fork traceability : blockchain meets agri-food supply chain. In *12th efita international conference/ rhodes island, greece/ june 27* (Vol. 29, p. 2019).

Steichen, M., Fiz, B., Norvill, R., Shbair, W., & State, R. (2018, July). Blockchain-based, decentralized access control for ipfs. In *2018 ieee international conference on internet of things (ithings) and ieee green computing and communications (greencom) and ieee cyber, physical and social computing (cpscom) and ieee smart data (smartdata)* (p. 1499-1506). doi: doi:10.1109/Cybermatics_2018.2018.00253

Subramanian, H. (2017, dec). Decentralized blockchain-based electronic marketplaces. *Commun. ACM*, *61*(1), 78–84. Consulté sur https://doi.org/10.1145/3158333 doi: doi:10.1145/3158333

Sumathi, M., & Sangeetha, S. (2020, avril). *Blockchain Based Sensitive Attribute Storage and Access Monitoring in Banking System* [article]. Consulté le 2021-02-12, sur www.igi-global.com/article/blockchain-based-sensitive-attribute-storage-and-access-monitoring-in-banking-system/249163 (ISSN : 2156-1834 Issue : 2 Journal Abbreviation : IJCAC Pages : 77-92 Publisher : IGI Global Volume : 10) doi: doi:10.4018/IJCAC.2020040105

Surasak, T., Wattanavichean, N., Preuksakarn, C., & Huang, S. C.-H. (2019). Thai agriculture products traceability system using blockchain and internet of things. *International Journal of Advanced Computer Science and Applications*, *10*(9). Consulté sur http://dx.doi.org/10.14569/IJACSA.2019.0100976 doi: doi:10.14569/IJACSA.2019.0100976

Szabo, N. (1997, Sep.). Formalizing and securing relationships on public networks. *First Monday*, *2*(9). Consulté sur https://firstmonday.org/ojs/index.php/fm/article/view/548 doi: doi:10.5210/fm.v2i9.548

Tanenbaum, A. S., & Steen, M. v. (2007). *Distributed systems : principles and paradigms* (2nd ed éd.). Upper Saddle RIiver, NJ : Pearson Prentice Hall. (OCLC : ocm70707891)

Technologies, P. (2017, nov). *A Postmortem on the Parity Multi-Sig Library Self-Destruct | Parity Technologies.* Consulté le 2022-07-28, sur https://www.parity.io/blog/a-postmortem-on-the-parity-multi-sig-library-self-destruct/

Thomason, J., Ahmad, M., Bronder, P., Hoyt, E., Pocock, S., Bouteloupe, J., ... Shrier, D. (2018). Chapter 10 - blockchain—powering and empowering the poor in developing countries. In A. Marke (Ed.), *Transforming climate finance and green investment with blockchains* (p. 137-152). Academic Press. Consulté sur https://www.sciencedirect.com/science/article/pii/B9780128144473000100 doi: doi:https ://doi.org/10.1016/B978-0-12-814447-3.00010-0

Tian, F. (2016, June). An agri-food supply chain traceability system for china based on rfid & blockchain technology. , 1-6. doi: doi:10.1109/ICSSSM.2016.7538424

Tian, F. (2017, June). A supply chain traceability system for food safety based on haccp, blockchain & internet of things. In *2017 international conference on service systems and service management* (p. 1-6). doi: doi:10.1109/ICSSSM.2017.7996119

Tolmach, P., Li, Y., Lin, S.-W., Liu, Y., & Li, Z. (2021, avril). *A Survey of Smart Contract Formal Specification and Verification.* arXiv. Consulté le 2022-07-28, sur http://arxiv.org/abs/2008.02712 (arXiv :2008.02712 [cs])

Toutes les Crypto-monnaies. (2022). Consulté le 2022-09-29, sur https://coinmarketcap.com/fr/all/views/all/

Tse, D., Zhang, B., Yang, Y., Cheng, C., & Mu, H. (2017, Dec). Blockchain application in food supply information security. In *2017 ieee international conference on industrial engineering and engineering management (ieem)* (p. 1357-1361). doi: doi:10.1109/IEEM.2017.8290114

Udokwu, C., Kormiltsyn, A., Thangalimodzi, K., & Norta, A. (2018). The state of the art for blockchain-enabled smart-contract applications in the organization. In *2018 ivannikov ispras open conference (ispras)* (p. 137-144). doi: doi:10.1109/ISPRAS.2018.00029

Vasin, P. (2014). BlackCoin's Proof-of-Stake Protocol v2. , 2. Consulté sur https://blackcoin.org/blackcoin-pos-protocol-v2-whitepaper.pdf

Vora, S., & Gor, R. (2022, jun). Study of average transaction confirmation time in a blockchain using queueing model. *IOSR Journal of Computer Engineerin*, *24*(3), 60-68. doi: doi:10.9790/0661-2403026068

Wang, G., Koshy, J., Subramanian, S., Paramasivam, K., Zadeh, M., Narkhede, N., ... Stein, J. (2015, août). Building a replicated logging system with Apache Kafka. *Proc. VLDB Endow.*, *8*(12), 1654–1655. Consulté le 2020-09-30, sur https://dl.acm.org/doi/10.14778/

2824032.2824063 doi: doi:10.14778/2824032.2824063

Wang, J., & Wang, H. (2019). Monoxide : Scale out blockchain with asynchronous consensus zones. In *Proceedings of the 16th usenix conference on networked systems design and implementation* (p. 95–112). USA : USENIX Association.

Wang, Q., Li, R., Wang, Q., & Chen, S. (2021). Non-fungible token (nft) : Overview, evaluation, opportunities and challenges. arXiv. Consulté sur https://arxiv.org/abs/2105.07447 doi: doi:10.48550/ARXIV.2105.07447

Wilhelmi, F., Barrachina-Muñoz, S., & Dini, P. (2022, février). End-to-End Latency Analysis and Optimal Block Size of Proof-of-Work Blockchain Applications. arXiv. Consulté le 2022-09-13, sur http://arxiv.org/abs/2202.01497 (arXiv :2202.01497 [cs])

Wood, G. (2018, septembre). *Why We Need Web 3.0.* Consulté le 2022-09-28, sur https://gavofyork.medium.com/why-we-need-web-3-0-5da4f2bf95ab

Wood, G., et al. (2022, août). ETHEREUM : A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER. *Ethereum project yellow paper*, *151*, 1–41.

Wu, Y., Song, P., & Wang, F. (2020, 13 Apr). Hybrid consensus algorithm optimization : A mathematical method based on pos and pbft and its application in blockchain. *Mathematical Problems in Engineering*, *2020*, 7270624. doi: doi:10.1155/2020/7270624

Wüst, K., & Gervais, A. (2017). Do you need a Blockchain ? In *2018 crypto valley conference on blockchain technology (cvcbt)* (pp. 45–54). Consulté le 2019-10-11, sur http://eprint.iacr.org/2017/375

Xiao, Y., Zhang, N., Lou, W., & Hou, Y. T. (2020, Secondquarter). A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys & Tutorials*, *22*(2), 1432-1465. doi: doi:10.1109/COMST.2020.2969706

Xie, C., Sun, Y., & Luo, H. (2017, Aug). Secured data storage scheme based on block chain for agricultural products tracking. In *2017 3rd international conference on big data computing and communications (bigcom)* (p. 45-50). Chengdu. Consulté sur http://ieeexplore.ieee.org/document/8113046/ doi: doi:10.1109/BIGCOM.2017.43

Xie, J., Tang, H., Huang, T., Yu, F. R., Xie, R., Liu, J., & Liu, Y. (2019). A Survey of Blockchain Technology Applied to Smart Cities : Research Issues and Challenges. *IEEE Commun. Surv. Tutorials*, *21*(3), 2794–2830. Consulté le 2019-10-03, sur https://ieeexplore.ieee.org/document/8642861/ doi: doi:10.1109/COMST.2019.2899617

Xu, C., Wang, K., & Guo, M. (2017, novembre). Intelligent Resource Management in Blockchain-Based Cloud Datacenters. *IEEE Cloud Computing*, *4*(6), 50–59. Consulté le 2022-09-07, sur https://ieeexplore.ieee.org/document/8260822/ doi: doi:10.1109/MCC.2018.1081060

Xu, X., Weber, I., & Staples, M. (2019). Case study : Agridigital. In *Architecture for blockchain applications* (pp. 239–255). Cham : Springer International Publishing. Consulté sur https://doi.org/10.1007/978-3-030-03035-3_12 doi: doi:10.1007/978-3-030-03035-3_12

Yadav, S. P., Agrawal, K. K., Bhati, B. S., Al-Turjman, F., & Mostarda, L. (2022, avril). Blockchain-Based Cryptocurrency Regulation : An Overview. *Computational Economics*, *59*(4), 1659–1675. Consulté le 2022-09-01, sur https://doi.org/10.1007/s10614-020-10050-0 doi: doi:10.1007/s10614-020-10050-0

Yaga, D., Mell, P., Roby, N., & Scarfone, K. (2018, octobre). *Blockchain technology overview* (Rapport technique N° NIST IR 8202). Gaithersburg, MD : National Institute of Standards and Technology. Consulté le 2022-06-22, sur https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf doi: doi:10.6028/NIST.IR.8202

Yoo, J., Jung, Y., Shin, D., Bae, M., & Jee, E. (2019, février). Formal Modeling and Verification

of a Federated Byzantine Agreement Algorithm for Blockchain Platforms. In *2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)* (pp. 11–21). doi: doi:10.1109/IWBOSE.2019.8666514

Yu, G., Wang, X., Yu, K., Ni, W., Zhang, J. A., & Liu, R. P. (2020). Survey : Sharding in Blockchains. *IEEE Access*, *8*, 14155–14181. (Conference Name : IEEE Access) doi: doi:10.1109/ACCESS.2020.2965147

Yuan, Y., & Wang, F.-Y. (2018, septembre). Blockchain and Cryptocurrencies : Model, Techniques, and Applications. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, *48*(9), 1421–1428. Consulté le 2022-07-29, sur `https://ieeexplore.ieee.org/document/8419306/` doi: doi:10.1109/TSMC.2018.2854904

Zamani, M., Movahedi, M., & Raykova, M. (2018, octobre). Rapidchain : Scaling blockchain via full sharding. In *Proceedings of the 2018 acm sigsac conference on computer and communications security* (p. 931–948). New York, NY, USA : Association for Computing Machinery. Consulté sur `https://doi.org/10.1145/3243734.3243853` doi: doi:10.1145/3243734.3243853

Zeng, Z., Zhang, X., & Xia, Z. (2022, février). Intelligent Blockchain-Based Secure Routing for Multidomain SDN-Enabled IoT Networks. *Wireless Communications and Mobile Computing*, *2022*, 1–10. Consulté le 2022-05-02, sur `https://www.hindawi.com/journals/wcmc/2022/5693962/` doi: doi:10.1155/2022/5693962

Zhan, Y., Wang, B., Lu, R., & Yu, Y. (2021). Drbft : Delegated randomization byzantine fault tolerance consensus protocol for blockchains. *Information Sciences*, *559*, 8-21. doi: doi:https ://doi.org/10.1016/j.ins.2020.12.077

Zhao, G., Liu, S., Lopez, C., Lu, H., Elgueta, S., Chen, H., & Boshkoska, B. M. (2019). Blockchain technology in agri-food value chain management : A synthesis of applications, challenges and future research directions. *Computers in Industry*, *109*, 83-99. Consulté sur `https://www.sciencedirect.com/science/article/pii/S0166361518305670` doi: doi:https ://doi.org/10.1016/j.compind.2019.04.002

Zhao, W., Jin, S., & Yue, W. (2019, août). Analysis of the Average Confirmation Time of Transactions in a Blockchain System. In *Queueing Theory and Network Applications : 14th International Conference, QTNA 2019, Ghent, Belgium, August 27–29, 2019, Proceedings* (pp. 379–388). Berlin, Heidelberg : Springer-Verlag. Consulté le 2022-09-15, sur `https://doi.org/10.1007/978-3-030-27181-7_23` doi: doi:10.1007/978-3-030-27181-7_23

Zheng, P., Zheng, Z., Luo, X., Chen, X., & Liu, X. (2018, mai). A detailed and real-time performance monitoring framework for blockchain systems. In *Proceedings of the 40th International Conference on Software Engineering : Software Engineering in Practice* (pp. 134–143). Gothenburg Sweden : ACM. Consulté le 2022-09-02, sur `https://dl.acm.org/doi/10.1145/3183519.3183546` doi: doi:10.1145/3183519.3183546

Résumé

La recherche sur les cadres applicatifs de la blockchain propose rarement une évaluation de performances. Cette thèse propose une méthodologie complète pour aider les intégrateurs logiciels à mieux comprendre et mesurer l'influence des paramètres de configuration sur la qualité globale des performances du service à long terme. Afin d'améliorer les performances, le nouveau protocole de consensus adaptatif Sabine (Self-Adaptive BlockchaIn coNsEnsus) est proposé afin de modifier dynamiquement l'un de ces paramètres dans le cadre du consensus PBFT. Le paramètre de configuration de ce consensus est le nombre de validateurs impliqués et résulte d'un compromis entre sécurité et performance. Le protocole Sabine vient donc maximiser ce nombre sous réserve que le débit de sortie corresponde au débit d'entrée. Sabine est évaluée et validée dans des contextes réels, dont les résultats montrent que Sabine a une erreur relative acceptable entre les débits de transaction demandée et engagée, pour une chaîne soumise à une forte demande. Deux nouveaux algorithmes de sélection des validateurs sont proposés et renversent le paradigme aléatoire des protocoles actuels pour choisir les nœuds amenant à de meilleures performances. Le premier se base sur un système de réputation récompensant les nœuds les plus rapides. Le second sélectionne les nœuds les plus proches en imposant un roulement continu de la sélection. Ces deux algorithmes ont été simulés et leurs impacts sur la décentralisation discutés. Cette sélection, associée avec Sabine, permet d'améliorer la sécurité en laissant plus de marge au système pour augmenter le nombre de validateurs. Ces différents travaux ouvrent la voie à des chaînes plus réactives, avec moins de latence et plus de débit.

Mots-clés: Blockchain, Tolérance aux fautes byzantine, Sécurité, Consensus, Contrôle adaptatif.

Abstract

Research on blockchain application frameworks rarely offers performance evaluation. This thesis proposes a comprehensive methodology to help software integrators better understand and measure the influence of configuration parameters on the overall quality of long-term service performance. In order to improve performance, the new adaptive consensus protocol Sabine (Self-Adaptive BlockchaIn coNsEnsus) is proposed to dynamically modify one of these parameters in the PBFT consensus. The configuration parameter of this consensus is the number of validators involved and result of a trade-off between security and performance. The Sabine protocol maximises this number provided that the output rate matches the input rate. Sabine is evaluated and validated in real-world settings, the results of which show that Sabine has an acceptable relative error between the requested and committed transaction rates, for a chain subject to high demand. Two new validator selection algorithms are proposed that reverse the random paradigm of current protocols to select the nodes leading to better performance. The first is based on a reputation system that rewards the fastest nodes. The second selects the closest nodes by imposing a continuous rotation of the selection. These two algorithms have been simulated and their impact on decentralisation discussed. This selection, associated with Sabine, improves security by giving the system more margin to increase the number of validators. This work opens the way to more reactive chains, with less latency and more throughput.

Keywords: Blockchain, Byzantine Fault Tolerance, Security, Consensus, Adaptative control.