MASTER EN SCIENCES INFORMATIQUES

Modèle de simulation par événements discrets pour les performances de synchronisation de la blockchain

El Hamdaoui, Jamal

Award date:
2021

Awarding institution:
Universite de Namur

UNIVERSITÉ DE NAMUR

FACULTÉ D'INFORMATIQUE

Modèle de simulation par événements discrets pour les performances de synchronisation de la blockchain

Jamal EL HAMDAOUI

RUE GRANDGAGNAGE, 21 ● B-5000 NAMUR(BELGIUM)

Remerciements

Je tiens à exprimer mes sincères remerciements à ma directrice de mémoire, Madame Marie-Ange Remiche, qui m'a encadré et orienté dans l'élaboration de ce travail. Je la remercie encore pour sa remarquable disponibilité et le temps qu'elle m'a accordé, pour ses relectures et remarques constructives, ainsi que pour ses précieux conseils.

Je remercie également l'ensemble de l'équipe pédagogique et tous les membres de l'université de Namur, qui m'ont accueilli au sein de leur institution et qui ont permis la réalisation de ce travail.

Résumé

Un modèle mathématique peut être un outil intéressant pour l'étude d'un système réel. Lorsque la solution analytique d'un modèle ne peut être envisagée, il est possible de le simuler afin d'obtenir des résultats numériques.

Le modèle mathématique étudié, dans le cadre de ce mémoire, est un modèle un peu particulier, basé sur la théorie des files d'attente, inspiré du système de crypto-monnaie Bitcoin. Il s'intéresse à un aspect primordial de ce système, fondement de sa fiabilité, dont il propose d'étudier le comportement et d'évaluer les performances.

L'objectif de ce travail est d'essayer de déterminer le niveau de précision avec lequel le modèle étudié permet d'évaluer les performances du système qu'il représente.

Un modèle de simulation à événements discrets est développé pour tenter de répondre à cette question. L'analyse des résultats ainsi obtenus par la simulation montre que le modèle étudié peut être une bonne approche pour étudier le comportement du système réel auquel il s'intéresse. Pour tendre vers des résultats plus précis, il serait, éventuellement, intéressant de mener des études statistiques afin disposer de données, pour les paramètres d'entrée du modèle, plus spécifiques au système réel.

Mots-clés : Bitcoin, Synchronisation, File attente serveurs infini, Départs par lot

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

Les modèles sont des représentations simplifiées de systèmes réels dont l'objectif est d'étudier le comportement ou d'évaluer les performances de ces derniers. La précision d'un modèle dépend du niveau de détail qu'il contient. Plus le niveau de détail est important, plus le modèle sera précis mais sa complexité augmentera également. Il y a donc un équilibre à trouver entre la précision attendue et la complexité d'un modèle.

Le diagramme suivant donne une vue globale des différentes possibilités pour l'étude d'un système réel [1, Fig. 1.1].

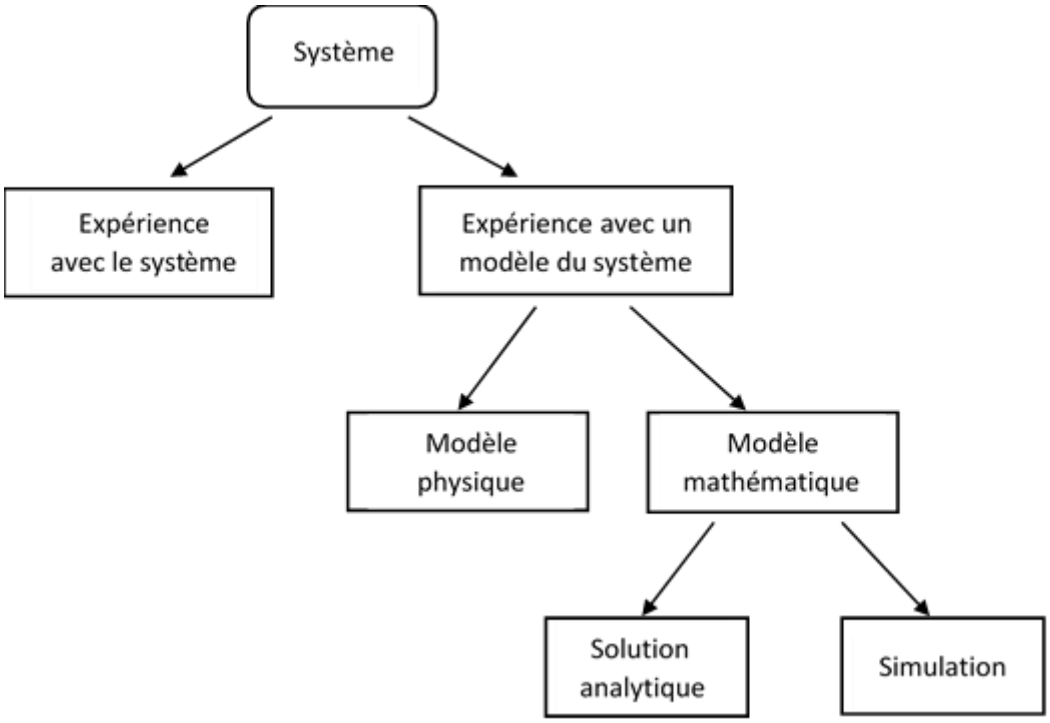


Figure 1 - Les différentes possibilités d'étudier un système réel. [1, Fig. 1.1]

Une étude pourrait être réalisée en faisant des essais, directement, sur un système réel et en observant son fonctionnement. La démarche serait plus rapide et les résultats ainsi obtenus seraient systématiquement valides. Mais ce n'est pas toujours possible. De plus, les risques et les coûts d'une telle approche pourraient être considérables voire catastrophiques en fonction de la complexité du système. Il est donc préférable d'utiliser un modèle du système pour réaliser cette étude [1, p. 4].

La première distinction entre les modèles, pour la réalisation d'une étude, est la possibilité de choisir entre un modèle physique et un modèle mathématique. Les modèles physiques sont une forme de reproduction, à l'échelle, du système réel dans le but d'effectuer des essais ou mener des expériences. Ils peuvent être utiles dans certaines situations mais ce ne sont pas les modèles les plus utilisés pour étudier un système. Les modèles les plus répandus et les plus intéressants, pour l'étude des systèmes, sont les modèles mathématiques qui contiennent la logique et les relations qui existent entre les grandeurs des systèmes qu'ils représentent. L'étude du système peut ainsi être réalisée en travaillant sur ces différents aspects contenus dans le modèle [1, p. 4].

L'étude d'un modèle mathématique a pour objectif de trouver des réponses aux questions qui ont conduit à s'intéresser au système réel. Dans le cas où le modèle est relativement simple, une solution analytique est possible. Autrement dit, un résultat peut être obtenu à partir des grandeurs contenues dans le modèle et des relations qui existent entre elles. Mais lorsque le modèle devient trop complexe, il peut être très difficile d'obtenir une solution analytique ou alors les ressources nécessaires pour y parvenir doivent être considérables. Dans ce cas, l'étude du modèle mathématique passe nécessairement par la simulation et on parle alors de modèle de simulation [1, p. 5]. Il est à noter que la solution analytique, lorsqu'elle est possible, est toujours préférable à la simulation car elle est plus précise.

1.1 MISE EN CONTEXTE DE LA PROBLÉMATIQUE

Le modèle étudié dans ce mémoire est inspiré du système Bitcoin. Il s'agit d'un système décentralisé à travers lequel s'échangent des biens et des services, entre les différents utilisateurs du réseau, au moyen d'une monnaie virtuelle. Le fonctionnement de ce système est régi par un ensemble de technologies et de protocoles. Il s'appuie sur la blockchain qui est une technologie désignant un ensemble de blocs, contenants des informations relatives aux échanges, reliés entre eux dans un ordre précis. C'est une base de données contenant tous les échanges effectués depuis le début de sa mise en application. Certains utilisateurs du réseau Bitcoin stockent une copie complète de la blockchain. Pour que le système reste fiable, Il est important que les différentes copies locales restent identiques. Des échanges de blocs s'effectuent de manière constante, entre les utilisateurs concernés, afin de mettre à jour les copies locales de la blockchain. C'est le mécanisme de synchronisation de la blockchain. Il arrive très souvent que les différentes copies locales de la blockchain ne soient pas complètement synchronisées durant un certain moment. C'est une situation normale mais qui ne doit pas durer trop longtemps. En effet, si la durée de cette désynchronisation se prolonge, la probabilité de réussite d'un échange frauduleux augmente également. Par conséquent, c'est la fiabilité du système qui risque d'être remise en cause. La synchronisation de la blockchain est donc un aspect essentiel du système. L'intérêt du modèle proposé est d'étudier le comportement de ce mécanisme et d'évaluer ses performances.

Le modèle, publié dans la littérature scientifique, est un système de file d'attente particulier, dont la description sera détaillée dans les chapitres suivants. L'approche de modélisation utilisée pour la représentation du système, qui vient d'être présenté, est basée sur la théorie des files d'attente, dont les notions importantes seront également abordées dans la suite de ce travail. Les

systèmes de files d'attente, qui sont des modèles mathématiques, permettent d'étudier le comportement et d'évaluer les performances des systèmes réels, notamment ceux basés sur la blockchain.

1.2 OBJECTIFS ET MÉTHODE

L'objectif de ce mémoire est de simuler un modèle mathématique publié dans une étude [2], pour lequel une solution analytique semble très compliquée à obtenir. Les relations mathématiques sont difficiles à exploiter pour obtenir des résultats numériques concernant le système modélisé. Le but est, par conséquent, de tenter de répondre à la question de recherche suivante : « Avec quel niveau de précision le modèle étudié permet-il d'évaluer le mécanisme de synchronisation de la blockchain sur laquelle s'appuie le système Bitcoin ? »

Pour atteindre cet objectif, un modèle de simulation à événements discrets sera développé dans le cadre de ce travail. Le modèle pourra, ainsi, être simulé afin d'obtenir des résultats numériques. Mais avant cela, il est d'abord nécessaire de bien comprendre le système réel qui est modélisé dans l'étude proposée [2] et l'intérêt pour un tel système. Ensuite, les outils nécessaires à cette simulation seront étudiés. Finalement, les résultats seront présentés et analysés.

1.3 PLAN DU MÉMOIRE

La structure de ce mémoire est composée de quatre chapitres et se présente comme suit :

- **Le chapitre 1 :** est une description du système réel et de son fonctionnement, avec toutes les notions utiles pour bien comprendre le modèle.

- **Le chapitre 2 :** passe en revue les publications de la littérature en rapport avec le système étudié. La synthèse des différentes approches de la modélisation et d'autres modèles sont également analysés. Le modèle étudié est également présenté de manière détaillée dans ce chapitre.

- **Le chapitre 3 :** présente les outils mathématiques utilisés pour la mise en œuvre du modèle de simulation.

- **Le chapitre 4 :** présente la méthodologie utilisée pour l'analyse et la mise en œuvre du modèle de simulation, la vérification de ce dernier, ainsi que l'analyse des résultats obtenus.

Le mémoire se clôture par une discussion relative aux résultats obtenus, suivie par une conclusion générale sur l'ensemble de ce travail.

CHAPITRE 1

2. PRÉSENTATION DU SYSTÈME BITCOIN

Le Bitcoin est un système complet à travers lequel s'échangent des biens et des services entre différents utilisateurs au moyen d'une monnaie virtuelle qu'on appelle le bitcoin. Afin d'être précis et pour éviter toute confusion, il est nécessaire de bien faire la distinction entre le « bitcoin » qui s'écrit avec une lettre minuscule et le « Bitcoin » qui s'écrit avec une lettre majuscule [3, p. 2].

Le bitcoin désigne l'unité de monnaie virtuelle, c'est-à-dire le moyen de paiement, utilisé au sein du système informatique pour l'échange de valeurs, biens et services, entre les différents participants [4, p. 1]. C'est équivalent, par analogie avec la monnaie physique que nous connaissons mieux, à une pièce d'un euro. On parle alors de crypto-monnaie et il est important de noter qu'il existe d'autres formes de crypto-monnaies semblables au bitcoin, mais celle-ci étant la plus connue.

Le Bitcoin, par contre, désigne le système dans sa globalité, c'est-à-dire l'ensemble des technologies et des protocoles qui régissent le fonctionnement de cette monnaie virtuelle. C'est un système complètement décentralisé où les participants effectuent des échanges directement entre eux, sans passer par un quelconque intermédiaire pour assurer cet échange. Il s'agit, en terme d'architecture, d'un style pair-à-pair où chaque participant peut jouer les rôles de serveur et de client. Le fait qu'il n'y ait plus d'autorité centrale pour garantir l'authenticité des échanges ne signifie pas que le système est moins fiable

Un autre concept important qu'il convient, également, de clarifier afin d'éviter la confusion est le concept de « blockchain » qui signifie chaîne de blocs. C'est la technologie sur laquelle s'appuie le Bitcoin et qui désigne un ensemble de blocs reliés entre eux dans un ordre précis, où chaque bloc contient des informations relatives à un certain nombre d'échanges effectués entre les participants et des informations relatives au bloc qui le précède dans la chaîne. C'est une base de données qui contient tous les échanges effectués depuis le début de sa mise en application. Elle est partagée par l'ensemble des participants au système, ce qui rend presque impossible la production d'une fausse copie vu le nombre élevé de participants. La cryptographie est le moyen utilisé pour valider, sécuriser et partager les informations contenues dans cette chaîne.

2.1 LE RÉSEAU BITCOIN

Le Bitcoin est système complètement décentralisé où chaque ordinateur participant au réseau peut jouer à la fois les rôles de serveur et de client. Il s'agit, en terme d'architecture, d'un style pair-à-pair formant un réseau maillé où chaque ordinateur met ses ressources à la disposition du réseau et peut fournir un service lorsqu'il joue le rôle de serveur ou demander un service lorsqu'il joue le rôle de client. Chaque ordinateur présent sur le réseau est appelé un « nœud ». Les nœuds du réseau bitcoin sont interconnectés et utilisent tous le même protocole pour communiquer entre eux [4, p. 139].

2.1.1 LES DIFFÉRENTS TYPES DE NŒUDS ET LEURS FONCTIONS

Il faut distinguer différents types de nœuds en fonction du type d'activité qu'ils exercent dans le réseau. Un nœud peut disposer d'une ou plusieurs fonctions parmi les suivantes [4, p. 140] :

- Le routage
- La blockchain
- Le minage
- Le portefeuille

Tous les nœuds du réseau disposent de la fonction de routage. En effet, c'est la fonction de base qui permet de participer à l'activité du réseau en validant et en propageant les blocs et les transactions.

Un nœud complet dispose de l'ensemble de ces fonctions. Il conserve une copie complète de la blockchain et celle-ci est constamment maintenue à jour. De cette manière, le nœud est capable de vérifier, de lui-même, n'importe quelle transaction sans avoir besoin de faire appel à une référence extérieure. Un nœud léger, par contre, ne conserve qu'une partie de la blockchain. Dans ce cas, pour vérifier une transaction, il utilise une méthode différente appelée « *Simplified Payment Verification* » qui signifie littéralement « *Vérification de Paiement Simplifiée* » [4, p. 140].

Un nœud de minage ou mineur dispose, en plus du routage, de la fonction de minage. C'est un nœud équipé de matériel informatique suffisamment puissant pour être capable de résoudre un algorithme de cryptage afin de créer un nouveau bloc. Il est en concurrence avec d'autres nœuds du réseau qui tentent également de trouver une solution. Une récompense est attribuée au premier nœud qui parvient à trouver une solution. Il est intéressant de noter que parmi les nœuds de minage, on peut également trouver des nœuds complets et des nœuds légers lorsque ces derniers participent à la recherche d'une solution [4, p. 141].

La fonction de portefeuille est l'application qui contient les adresses et les clés privées permettant d'effectuer des transactions. Généralement, le portefeuille utilisateur est géré par un nœud complet [4, p. 141].

2.1.2 LES POOLS DE MINAGE

Les nœuds de minage ou simplement « mineurs » peuvent travailler seuls ou en participant à un groupe. L'inconvénient pour un mineur, lorsqu'il travaille seul, est que sa puissance de calcul est fortement réduite et, par conséquent, la possibilité de produire une solution avant les autres nœuds est fortement réduite également. Il est donc plus intéressant pour un mineur de mettre sa puissance de calcul au service d'un groupe de mineurs pour former un pool de minage. Pratiquement, les mineurs sont reliés à un serveur de pool qui gère la participation des membres du groupe en utilisant un protocole de minage. Ce serveur est lui-même relié aux nœuds complets du réseau lui permettant d'avoir accès à une copie de la chaîne de blocs. Il peut, de cette manière, valider des transactions et blocs pour l'ensemble des mineurs qu'il gère, les dispensant alors de cette tâche. En cas de découverte d'un bloc valide, la récompense est d'abord versée à l'adresse bitcoin du pool qui se charge ensuite de la partager entre les différents mineurs en tenant compte de la contribution de chacun et moyennant des frais liés au service proposé.

Le problème avec cette configuration est que les mineurs sont dépendants du pool de minage. En effet, ils pourraient non seulement subir les conséquences d'une possible défaillance du serveur, mais pourraient également être complices d'une opération frauduleuse, sans s'en rendre compte, si le gestionnaire du serveur décide d'utiliser la puissance de calcul du pool pour engager une attaque. Pour pallier à cet inconvénient, une configuration décentralisée des pools de minage est utilisée. Les pools de minage sont alors organisés en sous-réseau pair-à-pair à l'intérieur du réseau Bitcoin et fonctionnent de la même manière que le système lui-même avec, toutefois, une difficulté plus faible [4, p. 211].

2.2 LES TRANSACTIONS

Une transaction est le terme qu'il convient d'utiliser pour désigner un échange entre les utilisateurs du système Bitcoin. C'est simplement une structure de données contenant différentes informations relatives aux participants et le montant en bitcoins des valeurs échangées. Toutes les transactions validées seront finalement inscrites de manière permanente dans un emplacement de la blockchain [4, p. 111].

Dans le cadre de ce travail, nous nous intéressons plus particulièrement à la manière dont les transactions circulent sur le réseau et à la manière dont celles-ci sont traitées par les différents participants. Par conséquent, la structure interne d'une transaction sera présentée de manière superficielle, sans détails inutiles.

2.2.1 STRUCTURE INTERNE D'UNE TRANSACTION

La structure interne d'une transaction se compose d'un certain nombre de champs que l'émetteur de celle-ci devra compléter, puis signer avant de l'envoyer à son destinataire. La transaction étant sécurisée par des procédés cryptographiques, seul son destinataire est en mesure d'utiliser le contenu. Les parties concernées par un échange disposent chacune d'une adresse propre qui sera renseignée dans le champ correspondant. C'est équivalent à virement bancaire traditionnel où les numéros de comptes représentent les adresses de l'expéditeur et du bénéficiaire [4, p. 111].

De manière générale, une transaction contient des entrées et des sorties renfermant des valeurs de bitcoins à transférer. Suivant le type de transaction, il peut y avoir une ou plusieurs entrées et une ou plusieurs sorties. Par exemple, une transaction peut avoir une seule entrée dirigée vers plusieurs sorties ou l'inverse. Les nombres d'entrées et de sorties sont spécifiées dans des champs distincts [4, p. 20].

Un autre champ, qui peut être intéressant pour l'analyse de ce travail et qui va être un peu plus détaillé, est le « Locktime » qui signifie « temps de verrouillage ». Il permet de préciser l'instant à partir duquel une transaction peut être validée. C'est équivalent à demander au réseau de ne pas traiter la transaction immédiatement, pour diverses raisons, et de préciser la condition d'un traitement ultérieur. Par conséquent, lorsqu'une transaction comprenant un temps de verrouillage arrive dans un nœud du réseau, son traitement est différé jusqu'au moment où la contrainte imposée est satisfaite. Dans la plupart des transactions habituelles, la valeur de ce champ est nulle, signifiant que la transaction peut être traitée immédiatement [4, p. 114].

2.2.2 LES POOLS DE TRANSACTIONS

La plupart des nœuds participant au réseau conservent, de manière temporaire, une copie des transactions qu'ils ont validé en attendant de pouvoir les insérer dans un nouveau bloc afin qu'elles puissent être confirmées. Cette liste temporaire, contenant des transactions valides en attente de confirmation, est appelée « transaction pool » ou encore « memory pool ». En fait, lorsqu'une transaction arrive dans un nœud, elle d'abord validée par ce dernier puis une copie est rajoutée à la liste des transactions en attente de confirmation. Ensuite, elle est diffusée, de proche en proche, aux autres nœuds pour finalement se propager sur l'ensemble du réseau [4, p. 160].

Certains nœuds conservent également une copie des transactions orphelines dans un pool à part, complètement distinct du précédent. Les transactions orphelines sont des transactions dont les entrées font référence aux sorties d'autres transactions qui n'ont pas encore été identifiées par le nœud en question. Elles sont donc conservées de manière temporaire, dans un pool de transactions orphelines, jusqu'à ce que les transactions de référence soient reçues par le nœud. Chaque fois qu'une nouvelle transaction est validée et mise en attente de confirmation, toutes les transactions orphelines sont analysées. Si certaines d'entre elles ont des entrées qui correspondent aux sorties de la dernière transaction validée, elles sont alors retirées du pool

pour validation et ensuite déplacées vers le pool des transactions en attente de confirmation [4, p. 160].

2.3 LES BLOCS

Un bloc est une structure de données permettant de regrouper une ou plusieurs transactions. Il se compose d'une en-tête contenant des informations relatives au bloc lui-même, et d'une partie rassemblant les dernières transactions qui ont été validées depuis le dernier bloc. La taille d'un bloc est très variable. Elle est déterminée par le nombre de transactions qu'il contient puisque la taille d'une en-tête reste fixe. Dès qu'un bloc est validé, il est rajouté à la blockchain pour constituer la base de données de toutes les transactions [4, p. 164].

Un bloc est identifié, de manière unique, par son empreinte numérique. Mais il peut également être identifié par sa position dans la chaîne. C'est ce qu'on appelle la hauteur du bloc, elle est représentée par son numéro d'ordre dans la blockchain. Mais la hauteur peut, dans certaines situations temporaires, identifier plusieurs blocs lorsque ceux-ci ont été générés de manière presque simultanée. Par conséquent, la hauteur ne peut pas être considérée comme un identifiant unique comme c'est le cas de l'empreinte numérique. Cependant, ces deux informations d'identification n'apparaissent pas de manière explicite dans la structure interne du bloc mais sont déduite, dynamiquement, par les nœuds du réseau [4, p. 165].

2.3.1 STRUCTURE D'UN BLOC

Un bloc de transactions se compose globalement d'une en-tête et d'un corps. Les informations relatives au bloc lui-même sont contenues dans son en-tête, permettant de l'identifier de manière unique dans la chaîne. Le reste des informations concerne les transactions qu'il regroupe. La figure suivante (Fig. 2) illustre la structure d'un bloc [5] :

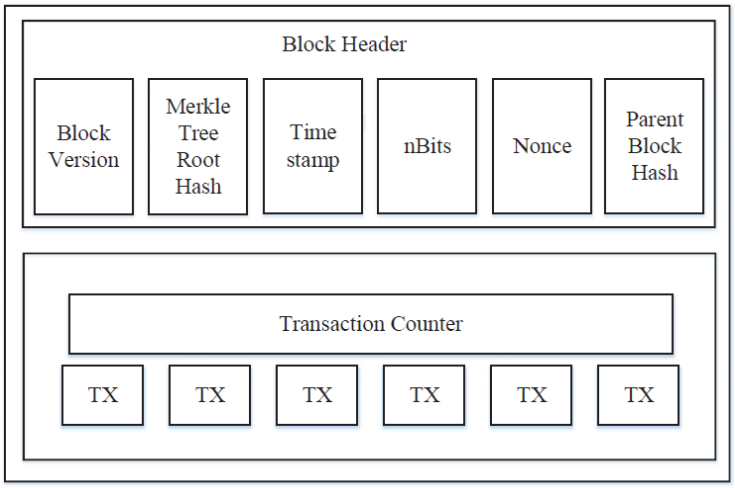


Figure 2 - Structure d'un bloc [5]

Les données contenues dans l'entête du bloc sont utiles pour lier correctement les blocs entre eux. Les principaux champs (Fig. 2) contiennent les informations suivantes [5] :

- L'empreinte numérique du bloc précédent pour son insertion correcte dans la blockchain.
- La version du bloc pour permettre les mises à jour logicielles et les protocoles de communication.
- L'horodatage, c'est-à-dire le temps, en secondes, qui s'est écoulé depuis que le bloc a été généré.
- L'empreinte de la racine de l'arbre de Merkle qui est une structure de données, sous forme d'arbre, permettant d'obtenir un sommaire des transactions contenues dans le bloc.
- Les champs *nonce* et *nBits* contiennent des valeurs utilisées pour l'ajustement du niveau de difficulté et le minage du bloc.

Le corps du bloc, quant à lui, contient un compteur pour les transactions qui se suivent et les transactions elles-mêmes.

2.3.2 LE BLOC DE GENÈSE

Le bloc de genèse est le tout premier bloc de la blockchain. Sa hauteur est zéro. Il est toujours valide et fourni, par défaut, dans les applications client. Ainsi, chaque nœud qui démarre dans le réseau possède toujours un bloc valide à partir duquel il peut reconstruire la blockchain complète [4, p. 166].

2.4 LA BLOCKCHAIN

Une blockchain, ou chaîne de blocs, est simplement un ensemble de blocs contenant toutes les transactions qui ont été validées et confirmées par le réseau. Elle constitue une base de données. Ces différents blocs sont liés entre eux, comme les maillons d'une chaîne, et agencés dans un ordre très précis où chaque bloc fait référence à son prédécesseur dans la chaîne. Un bloc contient une empreinte numérique qui permet de l'identifier de manière unique, ainsi que l'empreinte numérique de son prédécesseur appelé le bloc « parent ». Le premier bloc de la chaîne est un peu particulier en ce sens qu'il n'a pas été généré par un processus de minage, c'est un bloc qui est fourni par défaut comme bloc d'origine valide. C'est le « genesis block » qui signifie littéralement « bloc de genèse ».

Il peut arriver à un certain moment, dans le processus de génération des blocs, d'obtenir plusieurs blocs faisant référence à un même prédécesseur. C'est problématique puisqu'un bloc ne peut avoir qu'un seul parent. Par conséquent la chaîne n'est plus linéaire, elle présente un « fork » c'est-à-dire une bifurcation. C'est une situation qui peut se produire mais de manière temporaire. C'est le cas lorsque différents blocs sont générés (découverts) à des instants presque identiques. Dans un premier temps, il y aura différentes chaînes et finalement c'est la chaîne la plus longue qui sera conservée. En effet, si l'identifiant d'un bloc parent est modifié, il est nécessaire de modifier la référence dans le bloc enfant et de répéter ce processus pour les suivants jusqu'au dernier bloc de la chaîne. Etant donné que les ressources exigées pour le calcul de la solution augmentent avec le nombre de blocs, il est évident que c'est la chaîne la plus longue qui devient la plus sûre. En effet, plus on remonte loin dans la chaîne, plus les modifications seront difficiles à opérer car il faudra refaire le calcul pour tous les blocs successifs afin de rétablir la cohérence de la chaîne [4, p. 163].

2.4.1 PROCESSUS DE SYNCHRONISATION DE LA BLOCKCHAIN

La plupart des nœuds sur le réseau disposent, localement, d'une copie complète de la blockchain. C'est le cas des nœuds complets. Cette copie est régulièrement mise à jour. Chaque fois qu'un nouveau bloc est généré dans le système, il est rajouté à la blockchain après validation par le nœud qui le reçoit. Pour effectuer cette opération, le nœud recherche, dans l'entête du bloc qu'il reçoit, l'empreinte numérique du bloc précédent. Si cette empreinte numérique correspond à l'identifiant du dernier bloc de la chaîne locale, alors le bloc reçu est valide puisqu'il a le bon parent. Par conséquent, ce bloc est rajouté à la chaîne locale et il devient alors le dernier bloc de la blockchain dont dispose ce nœud [4, p. 167].

Que ce soit un nœud complet qui se connecte pour la première fois au réseau ou un autre qui reprend son activité après une certaine période d'absence, la procédure de synchronisation reste la même. S'il s'agit d'un nouveau nœud qui veut rejoindre le réseau, il dispose alors d'un seul bloc valide, le bloc de genèse, qui est fourni, par défaut, avec l'application client. Il devra donc construire toute la blockchain à partir de ce premier bloc. S'il s'agit, par contre, d'un nœud qui avait quitté le réseau et qui décide de reprendre son activité, il dispose alors,

localement, d'une copie incomplète de la blockchain. Il devra donc compléter sa chaîne locale avec les blocs manquants, ceux qui ont été générés durant son absence. Dans les deux cas, lorsque le nœud complet se connecte au réseau, il va tenter de mettre à jour sa chaîne de blocs locale. Pour cela, il échange une série de messages avec les autres nœuds complets du réseau afin de comparer les longueurs des chaînes et déterminer ainsi le nombre de blocs manquants. Le nœud peut ensuite demander les blocs nécessaires, en répartissant la charge sur l'ensemble des nœuds connectés, et finalement compléter sa blockchain locale. Si un nombre important de blocs sont manquants, la synchronisation s'effectue en plusieurs étapes, avec un nombre de blocs en transit qui doit rester constamment inférieur à une certaine limite [4, p. 148].

2.5 PROCESSUS DE MINAGE

Le minage est le processus qui permet de valider les transactions sur le réseau Bitcoin et de les intégrer dans la blockchain. Ce processus de validation des transactions est assuré par les nœuds du réseau qui s'entendent entre eux, sans avoir besoin de passer par une autorité centrale comme c'est le cas pour le système bancaire traditionnel. Cette validation par consensus est l'un des aspects les plus importants du minage car il permet de garantir, de manière décentralisée, la sécurité du système contre d'éventuelles transactions douteuses [4, p. 177].

Un autre aspect important du minage est la création de monnaie. En effet, pour inciter les nœuds du réseau à participer au minage, une récompense en bitcoins est accordée à chaque fois qu'un nouveau bloc de transactions est créé. Par conséquent, la quantité de bitcoins alimentant le système augmente avec l'activité de minage [4, p. 177].

Les nœuds qui participent au minage, appelés « mineurs », sont équipés de matériel informatique relativement performant et adapté à ce type d'activité. En effet, ils doivent disposer d'une puissance de calcul suffisante pour résoudre un problème compliqué de nature cryptographique. Les nœuds sont en concurrence les uns par rapport aux autres mais certains peuvent s'unir pour accroître leur puissance de calcul. Finalement, c'est le premier qui a résout le problème qui a droit à la récompense. La solution de ce problème est appelée « Proof-of-Work » qui signifie la « preuve de travail » et qui témoigne de l'effort consenti par le nœud pour parvenir à cette solution. En échange de cet effort, le nœud reçoit donc une récompense en bitcoins. Cette récompense provient de deux sources différentes : un montant fixe pour la création du bloc et des frais pour chaque transaction validée dans ce bloc [4, p. 178].

Le temps nécessaire pour la création d'un bloc est variable. En moyenne, un bloc est créé toutes les dix minutes. Et pour chaque nouveau bloc, un certain nombre fixe de bitcoins sont également créés et attribués au nœud qui a trouvé la solution. Ceci a pour conséquence d'augmenter la quantité totale de bitcoins dans le système. Mais cette augmentation n'est pas sans limite. En effet, un nombre total de bitcoins, qui ne peut pas être dépassé, a été fixé, dès le début de son lancement, dans les règles du système. Pour réguler le système, le montant fixe attribué pour la création d'un bloc est réduit de manière régulière, à chaque fois qu'un certain nombre de blocs a été atteint, et la difficulté du problème suivant à résoudre est constamment

adaptée. Par conséquent, la création de bitcoins décroit régulièrement, en diminuant de moitié tous les quatre ans environ, ce qui correspond au nombre de blocs à atteindre avant de procéder à une nouvelle diminution. Pour être plus précis, cette récompense est divisée par deux tous les 210.000 blocs. Cela se traduit par une diminution exponentielle jusqu'à atteindre, après quelques années, la limite qui été fixée. A ce moment-là, plus aucun nouveau bitcoin ne sera créé suite à la création de blocs. Et pour permettre au système de continuer à fonctionner, Il restera encore, comme seul incitant financier pour les mineurs, les frais de transaction. Pour que cela reste attrayant pour les mineurs, les frais de transaction adoptent un comportement inverse. Ils sont très faibles au départ et augmentent progressivement avec le nombre de transactions pour finalement devenir la seule source de récompense intéressante pour l'activité de minage [4, p. 178].

2.6 MÉCANISME DU CONSENSUS DÉCENTRALISÉ

Le réseau Bitcoin, avec sa structure décentralisée, permet de réaliser des échanges, de manière sûre, sans avoir besoin de passer par un quelconque intermédiaire de contrôle pour assurer cet échange. Les transactions sont authentifiées par l'ensemble des nœuds qui composent le réseau au moyen d'un mécanisme de consensus. Ce consensus est obtenu par l'interaction de différents processus qui opèrent de manière indépendante sur les nœuds du réseau. Les étapes de ce mécanisme du consensus se composent des quatre processus suivants [4, p. 181] :

- La validation des transactions
- L'insertion des transactions dans des nouveaux blocs
- La validation des nouveaux blocs
- L'assemblage et la sélection des chaînes de blocs

Chacun de ces processus sera détaillé séparément dans la suite de ce chapitre.

2.6.1 VALIDATION DES TRANSACTIONS

Lorsqu'un nœud complet reçoit une nouvelle transaction, il procède d'abord à une vérification de celle-ci afin de la valider. Pour cela, Il analyse la transaction en se basant sur une série de critères et de règles telles que la taille, la syntaxe, la correspondance entre les entrées et les sorties, etc. Si ces conditions sont satisfaites, la transaction devient valide. Dans le cas contraire, elle est directement rejetée. La transaction valide est alors insérée, localement, dans un pool de transactions et ensuite propagée à travers le réseau. En travaillant de cette manière, en toute indépendance, les nœuds évitent de propager des transactions invalides et construisent, localement et progressivement, un pool avec, uniquement, des transactions valides. Ce pool sera utile dans l'étape suivante du mécanisme du consensus [4, p. 182].

2.6.2 INSERTION DES TRANSACTIONS DANS UN BLOC

Tous les nœuds du réseau Bitcoin sont dotés de la fonction de routage, c'est-à-dire qu'ils valident et propagent les transactions à travers le réseau. Mais lorsqu'un nœud de minage valide une transaction, il l'enregistre dans un pool de transaction avant de la propager. Le but étant d'inclure les transactions reçues dans un nouveau bloc. Pour cela, le nœud de minage construit un nouveau bloc, appelé bloc candidat, destiné à contenir ces nouvelles transactions. Ensuite, il récupère les transactions qui ont été enregistrées dans le pool de transactions, depuis le dernier bloc, pour les inclure dans le bloc qu'il vient de construire tout en continuant à alimenter le pool pour préparer le bloc suivant. A ce stade, ce bloc n'est pas encore valide. Il va alors, en même temps qu'il continue à valider et propager les transactions, utiliser sa puissance de calcul pour miner le nouveau bloc qu'il vient de construire afin de le valider. Mais il est aussi en compétition avec d'autres nœuds du réseau qui tentent réaliser la même opération.

Si, entre temps, il reçoit un bloc, de même hauteur, miné par un autre nœud du réseau, il le valide et met fin à la recherche de la solution pour ce bloc. Il supprime également de son pool de transactions celles qui sont contenues dans le bloc qu'il vient de valider. Enfin, il recommence avec la construction du bloc suivant.

Si, par contre, le nœud considéré parvient à trouver une solution avant les autres, c'est-à-dire qu'il réussit à miner le bloc qu'il vient de construire, il propage alors celui-ci à travers le réseau. De cette manière, les nœuds concurrents qui recevront ce bloc pourront le valider et mettre fin à la recherche d'une solution pour ce bloc et recommencer avec le bloc suivant.

La difficulté du minage est régulièrement réévaluée de sorte qu'un bloc soit créé, en moyenne, toutes les dix minutes. Toutes les transactions contenues dans le pool de transactions sont valides mais pas encore confirmées. Dès qu'un bloc est validé, les transactions qu'il contient deviennent des transactions confirmées [4, p. 183].

Les transactions du pool de transactions sont insérées dans les blocs candidats en fonction de leur priorité. Les règles qui déterminent les priorités sont principalement liées au frais contenus dans les transactions et à leur ancienneté dans le pool [4, p. 184].

2.6.3 VALIDATION DES NOUVEAUX BLOCS

Comme dans le cas des transactions, Lorsqu'un nœud complet reçoit un nouveau bloc, il procède d'abord à une vérification de celui-ci afin de le valider. Pour cela, Il analyse le bloc, de la même manière que pour les transactions, en se basant sur une série de critères et de règles telles que la taille du bloc, la syntaxe, la validité des transactions qu'il contient, etc. Si ces conditions sont satisfaites, le bloc devient valide et il est alors propagé à travers le réseau. Dans le cas contraire, il est invalide et directement rejeté. En travaillant de cette manière, en toute indépendance, les nœuds évitent de propager des blocs qui ne sont pas valides. Finalement, seuls les blocs valides seront utilisés dans la dernière étape du mécanisme du consensus [4, p. 201].

2.6.4 MÉCANISME DE CONSTRUCTION DE LA BLOCKCHAIN

Une fois qu'un bloc a été validé par les nœuds du réseau, il ne reste plus qu'à le lier à la chaîne de blocs actuelle. Le nœud complet insère simplement le nouveau bloc reçu dans la chaîne de bloc principale en le liant à son parent qui est le dernier bloc de la chaîne. Il dispose alors d'une copie complète et à jour de la blockchain.

La première difficulté à laquelle peut être confronté le nœud, durant cette étape, c'est lorsqu'il y a plusieurs chaînes de blocs. En effet, lorsque plusieurs blocs valides sont créés, par des mineurs différents, à des instants très proches les uns des autres, ils font tous référence au même bloc parent. Par conséquent, la chaîne de blocs principale présente des bifurcations c'est-à-dire des chaînes secondaires. Lorsqu'une telle situation se présente pour un nœud, celui-ci n'aura pas d'autre choix que d'étendre l'une de ces chaînes. Lorsque les chaînes sont de même longueur, il se base sur le bloc ayant le niveau de difficulté le plus élevé sinon c'est la chaîne la plus longue qui est choisie. La régulation s'opère, avec l'ajout de nouveaux blocs, dès qu'une chaîne commence à devenir plus longue que les autres. La chaîne principale, la plus longue, sera gardée et les chaînes secondaires finiront par être abandonnées.

L'autre difficulté qui peut se présenter, c'est lorsqu'un nœud reçoit un bloc valide avant son parent. Par conséquent, il est impossible d'insérer le bloc dans une chaîne puisqu'il ne trouve aucune référence qui lui correspond. Il s'agit alors d'un bloc « orphelin ». Cette situation se produit lorsqu'un bloc et son parent sont créés avec des écarts de temps relativement courts et que le bloc considéré est reçu, par un nœud complet, avant son parent. Dans ce cas, le bloc valide sera mis en attente dans un pool de blocs orphelins jusqu'à ce que son parent soit trouvé et insérer dans la chaîne. A ce moment, le bloc orphelin peut être retiré du pool et inséré dans la chaîne également, en le lien alors à son parent [4, p. 202].

Le processus complet, depuis l'arrivée des transactions jusqu'à l'insertion du bloc dans la blockchain, est résumé à la figure 3 [6] :

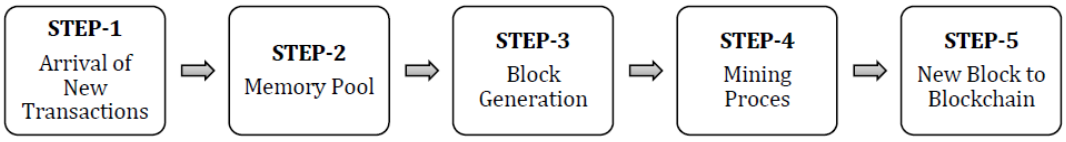


Figure 3 - Mécanisme de construction de la blockchain. [6]

Cette figure (Fig. 3) permet d'avoir un aperçu global des différentes étapes nécessaires à la construction de la blockchain. Ces différentes étapes ont été détaillées dans les sections précédentes de ce même chapitre.

2.7 FORMATION DE BRANCHES SECONDAIRES DANS LA BLOCKCHAIN

Il peut arriver que des blocs soient minés par des nœuds distincts à des instants très proches l'un de l'autre. Il est important de souligner que ce sont des situations qui se produisent régulièrement. Par conséquent, les blocs se propagent à travers le réseau et les autres nœuds les reçoivent à des moments différents et dans des ordres différents. Lorsqu'un nœud reçoit le premier bloc, il le valide et le lie à son parent sur la copie locale de la chaîne de blocs. Un instant plus tard, il reçoit le second bloc qui fait référence au même parent. Puisque le bloc est valide, il va également être lié à son parent dans la chaîne de blocs locale. Il apparaît alors une bifurcation et formation d'une chaîne secondaire (Fig.4). Suivant l'ordre dans lequel les blocs sont reçus par les nœuds, il existera dans le système, à un moment donné, deux versions différentes de la chaîne de blocs. Mais cette situation, relativement fréquente, n'est que temporaire. En effet, les nœuds choisissent toujours la chaîne la plus longue pour lier le dernier bloc reçu. Et lorsque les chaînes sont de même longueur, le nœud se base sur celle qui a cumulé la plus importante difficulté. Généralement, la situation est régulée après un seul bloc, une situation avec deux blocs étant beaucoup moins fréquente. La chaîne secondaire est alors progressivement abandonnée et il ne restera plus que la chaîne principale vers laquelle vont tendre tous les nœuds du réseau afin d'obtenir une même version locale de la chaîne de blocs [4, p. 204].

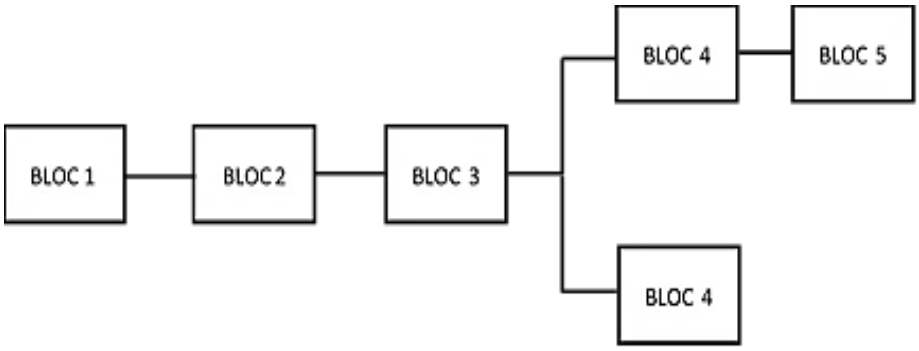


Figure 4 - Blockchain avec plusieurs branches

2.8 ATTAQUES PAR CONSENSUS

Si un ensemble de mineurs décident de mutualiser leur puissance de calcul dans le but d'adopter un comportement frauduleux, ils peuvent atteindre leur objectif de manière certaine s'ils totalisent la majorité de la puissance de calcul. Ils pourraient y parvenir avec une puissance moindre mais la majorité permet d'assurer une réussite certaine. Evidemment, dans le cas d'un réseau décentralisé, une telle situation est fort peu réalisable d'autant plus que la puissance de calcul globale du réseau augmente sans cesse. Par conséquent, il est très difficile de concentrer une puissance suffisante pour réaliser, de manière certaine, des opérations frauduleuses.

Lorsque la puissance de calcul le permet, il peut se produire deux types d'opérations frauduleuses dans le système :

- Attaque par double-dépense
- Attaque par déni de service

Une attaque par double-dépense consiste à dépenser deux fois la même transaction. Un attaquant peut, par exemple, faire une dépense puis récupérer le montant de cette dépense. Pour cela, il suffit de posséder une puissance de calcul suffisante pour invalider un bloc contenant la première transaction confirmée en minant un bloc identique, de même hauteur, avec la transaction modifiée. Cela provoque une chaîne de blocs secondaire. Il apparaît alors deux chaînes avec des blocs valides en derniers mais dont l'une de deux contient le bloc frauduleux. Il suffit alors à l'attaquant de continuer à miner les blocs suivants pour allonger la chaîne contenant le bloc frauduleux afin qu'elle puisse devenir la chaîne principale vers laquelle vont tendre les autres nœuds du réseau. L'autre chaîne sera progressivement abandonnée. Généralement, on considère qu'après six blocs une transaction est définitivement confirmée et qu'il difficile pour un attaquant d'aller aussi loin dans la chaîne pour manipuler un bloc.

Une attaque par déni de service permet de faire en sorte que les transactions d'un participant particulier ou d'un ensemble de participants soient ignorées par le système. L'attaquant, qui possède une puissance de calcul suffisante, peut un bloc en omettant les transactions provenant des adresses des participants ciblés. Si, par contre, les transactions font déjà partie d'un bloc valide miné par un autre nœud, alors l'attaquant peut procéder de la même manière que dans le cas d'une double-dépense en minant un bloc identique duquel il retire les transactions concernées.

Evidemment, pour parvenir à réaliser ce genres d'attaques de manière certaine, il est nécessaire de disposer de la majorité de la puissance de calcul. Etant donné que les nœuds sont de plus en plus nombreux sur le réseau et que la puissance de calcul nécessaire pour miner un nouveau bloc augmente sans cesse, il est peu probable qu'une telle situation puisse devenir réalisable [4, p. 214].

CHAPITRE 2

3. MODÈLES DE FILES D'ATTENTE POUR L'ANALYSE DU SYSTÈME BITCOIN

Ce chapitre passe en revue quelques modèles de files d'attente proposés dans la littérature scientifique. Le but est de mieux comprendre la manière dont les différents mécanismes qui régissent le fonctionnement des systèmes basés sur la blockchain sont modélisés au moyen d'une approche par files d'attente.

La démarche adoptée pour le développement de ce chapitre est de partir d'articles [7] [8] proposant une vue suffisamment large sur l'état de la blockchain dans la littérature scientifique pour ensuite converger, de manière progressive, vers des modèles de files d'attente relatifs aux systèmes utilisant la blockchain et en particulier le système Bitcoin. Dans un premier temps, les différentes approches de modélisation de tels systèmes sont étudiées. Puis quelques modèles sont analysés, en retenant uniquement l'approche par files d'attente. D'abord des modèles simples, ensuite un modèle plus complexe semblable au modèle en cours d'étude. Finalement, le modèle qui fait l'objet de ce mémoire sera analysé à partir d'éléments concrets appris sur les modèles précédents avant d'être finalement simulé.

3.1 PRÉSENTATION GÉNÉRALE LA BLOCKCHAIN

Le premier article [7] est très intéressant dans la mesure où il permet d'avoir une vue très générale de ce qui est publié dans la littérature relative aux systèmes utilisant la technologie blockchain. Les auteurs de cet article présentent d'abord une cartographie des différentes études publiées ces dernières années au sujet de la blockchain, au moyen d'un classement par catégories des différents articles analysés. Ils ont effectué une revue de la littérature et sélectionné soixante-six articles pertinents. Ensuite, ils classent ces différents articles dans différentes catégories en fonction des sujets traités par ceux-ci. Ce premier classement est présenté, par ces auteurs, à la figure 5. Elle permet de voir (Fig. 5), en ordonnée, la quantité d'articles publiés pour chaque catégorie de sujets représentée en abscisse.

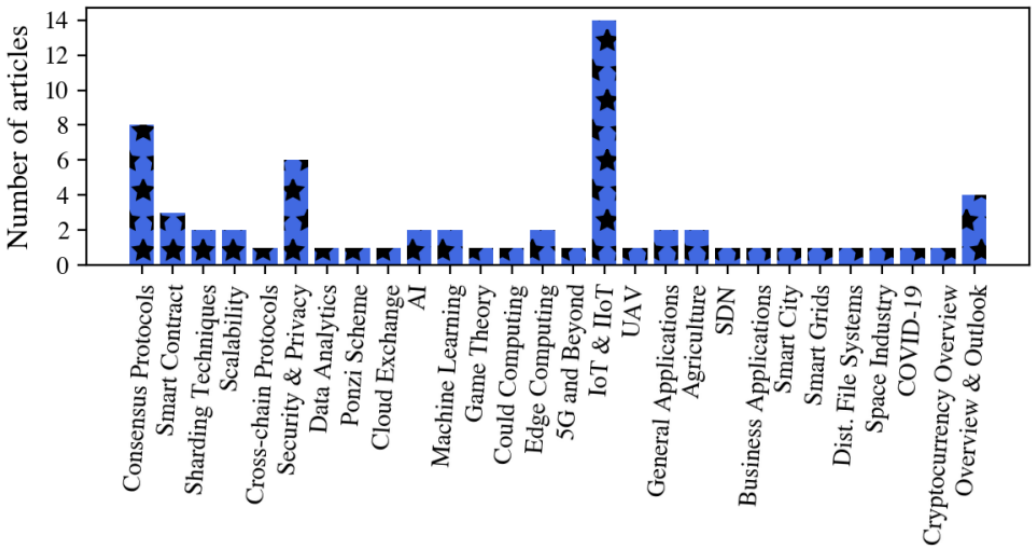


Figure 5 - Nombre d'articles pertinents, classés par catégories de sujets, publiés ces dernières années sur la blockchain. [7, Fig. 1]

Partant de ce premier classement, les auteurs établissent déjà un premier constat intéressant : parmi ces différentes études qui s'intéressent à la blockchain, trois sujets sont traités de manière plus fréquente que les autres, c'est-à-dire qu'ils présentent un intérêt particulier pour la recherche dans ce domaine. Ces trois sujets sont :

- L'Internet des Objets
- Les protocoles de consensus
- La sécurité et la confidentialité

Les auteurs effectuent ensuite un classement chronologique de ces différents articles, en fonction de leur année de publication. La figure 6 montre, pour chaque année, le nombre de catégories concernées ainsi que la quantité d'articles pertinents publiés pour chacune de ces catégories. Ce second classement leur permet d'établir deux autres constats intéressants (Fig. 6) :

- Le nombre annuel de publications relatives à la blockchain ne cesse de croître
- Le nombre de sujets différents traités chaque année, c'est-à-dire la diversité des sujets, augmente chaque année

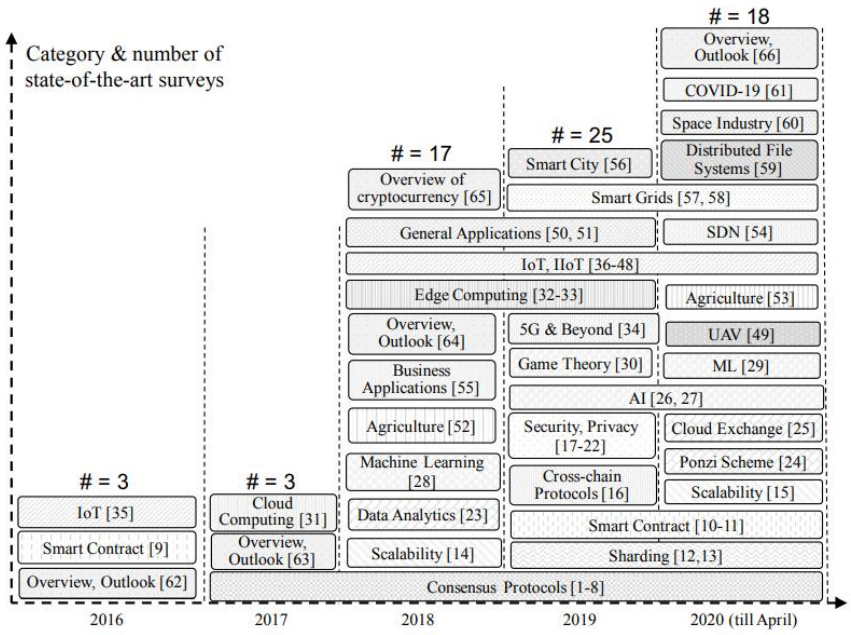


Figure 6 - Classement chronologique des différents catégories d'articles pertinents, publiés ces dernières années sur la blockchain. [7, Fig. 2]

Partant des classements précédents, les auteurs constatent que la plupart de ces études s'intéressent plus particulièrement à des applications s'appuyant sur la blockchain dans des domaines très variés. Ils déduisent de cette observation qu'il n'y a pas encore d'étude complète s'intéressant aux fondements des différentes implémentations de la blockchain, basée sur des théories récentes, des modélisations et des outils d'évaluation spécifiques, dans le but de mieux comprendre les blockchains et d'améliorer leurs performances.

La motivation des auteurs pour la réalisation de cette étude est justement de combler ce manque en proposant une étude complète permettant d'améliorer la compréhension et les performances des blockchains. Pour permettre cela, les auteurs se concentrent principalement sur des études récentes et pertinentes qui traitent des points suivants :

- Les théories permettant l'amélioration des performances des blockchains
- Les modélisations et techniques permettant une meilleure compréhension des blockchains
- Les mesures de performance et les outils d'évaluation spécifiques aux blockchains

Le contenu de ses différents points est synthétisé de la manière suivante :

1. Les théories permettant l'amélioration des performances des blockchains

a) Les théories récentes permettant l'amélioration des performances des blockchains

Pour améliorer les performances des blockchains, il faut agir sur les éléments mesurables qui les composent, c'est-à-dire les métriques de performance. Pour cela, les auteurs présentent plusieurs études récentes proposant des théories pour l'amélioration des performances des blockchains. Ces théories récentes se focalisent principalement sur les métriques de performance suivantes :

- **Le débit et le délai de confirmation (ou latence)**
  Le débit et le délai de confirmation sont liés. En effet, si le délai de confirmation diminue alors le débit augmente et inversement. Par conséquent, plusieurs théories ont été proposées dans le but de réduire le temps de confirmation des transactions.

- **L'efficacité du stockage**
  La taille des blockchains augmente sans arrêt. En effet, les blocs contenant des nouvelles transactions sont ajoutés à intervalles plus ou moins réguliers à la blockchain existante, ce qui a pour effet d'augmenter la taille de celle-ci. L'espace nécessaire pour son stockage augmente également ce qui pose un problème pour les nœuds du réseau qui stockent une copie complète de la blockchain. Par conséquent, différentes théories ont été proposées dans le but de permettre un stockage plus efficace.

- **La fiabilité de blockchains**
  Les participants d'une blockchain qui désirent rejoindre le système doivent nécessairement passer par un nœud pair puisqu'il s'agit d'un réseau pair-à-pair. Le choix de ce nœud est très important pour le participant, pas uniquement pour des raisons de proximité ou de ressources, mais aussi pour des raisons de sécurité. Le pair sélectionné doit être fiable. Certaines théories ont alors été proposées afin de permettre d'évaluer la fiabilité d'un pair.

D'autres aspects de la performance comme l'évolutivité, les protocoles de consensus et les structures de données nécessitent également, selon les auteurs, de proposer des nouvelles solutions afin d'améliorer les performances des blockchains.

b) Amélioration de l'évolutivité

Lorsque la taille d'un réseau utilisant la blockchain augmente, son débit n'évolue pas en conséquence. Autrement dit, le débit reste le même alors que la taille du réseau augmente. Cela a donc un impact sur les performances du système.

c) Nouveaux protocoles

Des nouvelles implémentations de la blockchain peuvent exiger d'autres mécanismes de consensus (voir chapitre 1) que ceux utilisés dans les implémentations classiques. Par conséquent, les protocoles de consensus classiques peuvent ne plus être adaptés à ces nouvelles exigences de consensus.

d) Nouvelles structures de données (infrastructures)

L'utilisation des blocs comme structures de données dans les blockchains n'est pas toujours la meilleure solution pour les différents types de blockchains. En effet, le nombre de transactions qu'un bloc peut contenir est limité et le délai pour le traitement des transactions qu'il contient peut être relativement long. Ainsi, pour des blockchains où le nombre de nœuds est plus restreint, comme les blockchains privées qui nécessitent une autorisation pour devenir participant, l'utilisation du bloc n'est pas adaptée puisqu'il influence les performances.

2. Les modélisations et techniques permettant une meilleure compréhension des blockchains

Les auteurs de cet article estiment que les études qui se sont intéressées à développer une meilleure compréhension des blockchains, à partir d'études existantes, ne se focalisent que sur des sujets très restreints telles que la confidentialité, la sécurité, etc. Les auteurs suggèrent d'élargir le champ de ces sujets afin de permettre une meilleure compréhension des blockchains. Ils proposent donc d'étendre ces études à d'autres sujets comme ceux qui suivent.

a) Les théories basées sur les graphes

Les auteurs présentent différentes études récentes qui utilisent la théorie des graphes pour permettre une meilleure compréhension des blockchains. Ils considèrent que la théorie des graphes et les techniques permettant d'explorer les données structurées par ces derniers sont des outils efficaces qui permettent de mettre en évidence des résultats intéressants concernant les informations véhiculées par les blockchains.

b) Les modélisations stochastiques et la théorie des files d'attente

Les différents mécanismes régissant le fonctionnement des blockchains peuvent être représentés par des modèles stochastiques. Le but de ces modèles théoriques étant d'étudier

le comportement des blockchains en utilisant les probabilités et permettre de faire des prédictions du comportement dans des conditions différentes.
La théorie des files d'attente permet de représenter la dynamique des blocs et transactions dans le réseau d'une blockchain. Les différentes phases de leur traitement peuvent être représentées par des modèles de file d'attente afin d'être analysées. Cela permet de mieux comprendre la dynamique de leur comportement dans un réseau blockchain afin d'améliorer les performances d'un système.

Les auteurs présentent, dans cet article [7], des études récentes qui utilisent la modélisation et la théorie des files d'attente pour décrire les différents mécanismes qui régissent le comportement des blockchains.

La modélisation par les files d'attente étant un aspect important de ce mémoire, d'autres articles seront étudiés plus en détail dans la suite de ce chapitre.

c) **Analyse de données pour les systèmes de crypto-monnaie**

Les auteurs présentent plusieurs études récentes qui utilisent des techniques d'analyse de données dans le but de détecter des risques de fraudes dans les échanges qui s'effectuent au sein des systèmes de crypto-monnaie, telles que les tentatives de blanchiment d'argent, les escroqueries, etc. Il ne s'agit donc pas d'identifier un risque de sécurité ou de confidentialité, et d'évaluer son impact sur la blockchain, mais de détecter ce risque alors qu'il ne s'est pas encore produit.

3. Les mesures de performance et les outils d'évaluation spécifiques aux blockchains

D'après le constat effectué par les auteurs de cet article [7], très peu d'études se sont intéressées aux mesures de performance et aux outils d'évaluation spécifiques aux blockchains. Ils ne trouvent aucun article à ce sujet parmi tous ceux qu'ils ont sélectionné. Par conséquent, les auteurs présentent, dans cette partie, les études pertinentes qui ont traité cet aspect.

Pour les métriques de mesure, les études présentées traitent du débit du réseau, du délai de confirmation pour les transactions, de la sécurité, etc. En ce qui concerne les outils d'évaluation, elles proposent une plate-forme logicielle appelée «BlockSci» [9] permettant d'analyser les données brutes d'une blockchain afin de produire des informations utiles, un cadre de référence pour des analyses comparatives des performances de différentes blockchains [10] et un simulateur de réseau configurable pour le système Bitcoin [11] permettant des mesures de performance du réseau avec des conditions paramétrables.

Finalement, ce premier article [7], très intéressant, a permis de mettre en lumière l'état de la recherche actuelle concernant les différentes mises en œuvre de la blockchain. Le classement réalisé par les auteurs de cette étude permet de mettre en évidence que la plupart des articles traitent des applications de la blockchain dans des domaines très variés, mais peu d'entre eux

s'intéressent vraiment aux mécanismes de fonctionnement de ces différents systèmes, à leurs modélisations ou à des outils d'évaluation spécifiques. En réalisant ce travail, les auteurs voulaient combler ce manque en proposant une étude complète, basée sur des articles récents et pertinents, dont l'objectif est d'améliorer la compréhension et les performances des blockchains.

3.2 APPROCHES DE MODÉLISATION DE LA BLOCKCHAIN

La revue de la littérature fait apparaître qu'il existe, actuellement, différentes approches de modélisation pour l'évaluation des systèmes utilisant la technologie blockchain. L'article [8] permet de mettre en évidence trois types d'approches en fonction du point de vue adopté pour l'évaluation de la blockchain :

- Approches analytiques
- Approches basées sur la simulation
- Approches basées sur l'émulation

L'approche basée sur l'émulation permet d'imiter un système réel ou une partie de celui-ci afin d'étudier son comportement et d'évaluer ses performances. C'est un modèle qui reproduit le même comportement que le système réel. L'émulation est une approche d'évaluation de systèmes qui peut être très précise mais nécessite des ressources importantes pour les calculs et le stockage. C'est une approche qui peut être utilisée lorsque les ressources disponibles ne constituent pas un frein et que les performances du système utilisant la blockchain nécessitent d'être évaluées avec un niveau de détail relativement important.

L'approche analytique peut être utilisée lorsque le comportement d'un système peut être décrit au moyen d'équations mathématiques. Mais lorsque le fonctionnement du système est trop complexe et qu'il devient difficile ou impossible de décrire son comportement par un modèle mathématique alors la simulation peut être utilisée.

L'approche basée sur la simulation permet d'approcher le comportement d'un système réel ainsi que son évolution au cours du temps au moyen d'un modèle informatique. La simulation peut être utilisée pour étudier le comportement de systèmes complexes pour lesquels une solution analytique est difficile à développer ou simplement impossible. Elle est moins précise que l'émulation mais nécessite une infrastructure moins importante.

Selon l'auteur [8], faisant référence à un autre article [6], la modélisation analytique et la simulation sont des outils incontournables pour la plupart des systèmes utilisant la technologie blockchain. En effet, d'après [6], ils sont très utiles lorsqu'il s'agit d'étudier le comportement et d'évaluer les performances de ces systèmes. Généralement, les modèles s'appuyant sur ce type d'approches s'obtiennent en effectuant des abstractions, c'est-à-dire des simplifications du système réel. Ils sont le résultat d'une observation limitée à un aspect particulier de son fonctionnement. Les modèles ainsi obtenus embarquent un certain niveau de détail qui dépend des abstractions effectuées. Par conséquent, cela signifie que la précision de ces modèles est également influencée par les abstractions effectuées, c'est-à-dire que plus les simplifications sont importantes, moins le modèle sera précis. Il est donc important d'avoir un niveau de détail suffisant, tout en faisant les simplifications nécessaires, pour que les mesures effectuées se rapprochent le plus possible de la réalité.

L'auteur explique également que les approches analytiques et la simulation sont plutôt utilisées dans les travaux académiques alors que l'émulation est une approche qui est plutôt réservée au domaine industriel, permettant d'évaluer les performances d'applications de la blockchain dont l'utilisation est plutôt restreinte au domaine privé, comme les applications d'entreprises.

L'article [8] permet de passer en revue l'utilisation de ces différentes approches de modélisation de la blockchain afin d'effectuer un classement. Pour cela, il reprend une classification précédente réalisée par l'étude [12], qu'il propose d'étendre aux outils industriels et présente, ensuite, les spécificités et limites d'utilisation de ces différentes classes. L'auteur se base sur une revue de la littérature, qu'il analyse de manière critique en tentant d'identifier les stratégies et les défis propres à chacune des approches identifiées. Il complète ensuite cette analyse en l'étendant aux outils utilisés dans le domaine industriel. Il aboutit ainsi à une classification en cinq groupes (Fig. 7), en fonction de la stratégie qui est mise en œuvre par les différentes approches identifiées et des défis auxquels elles tentent de répondre.

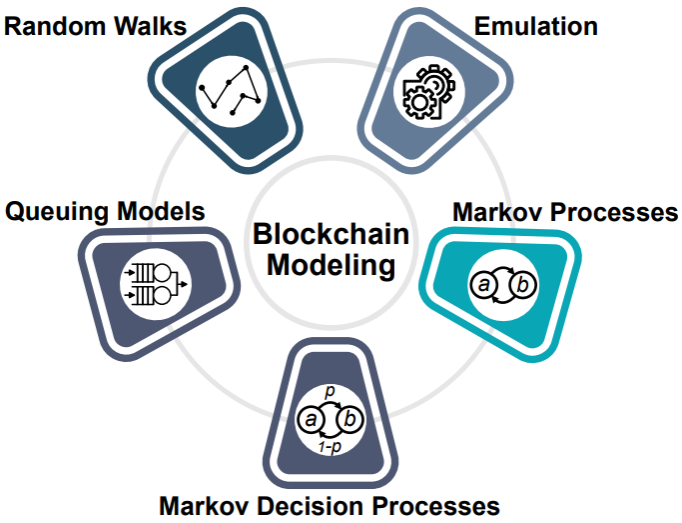


Figure 7 - Classification des approches de modélisation de la blockchain. [8, Fig. 1]

Mis à part l'émulation, la plupart de ces classes se basent sur les approches analytiques ou de simulation. Mais la seule classe à mettre en évidence, dans le cadre de ce travail, est celle relative aux modèles de files d'attente, en anglais « Queuing Models ». Cette classe sera décrite dans la suite de ce chapitre. Les autres classes, bien qu'intéressantes également, ne seront pas reprises dans le contenu de ce travail.

Les modèles de files d'attente sont des modèles mathématiques, basés sur la théorie des files d'attente, permettant d'étudier le comportement et d'analyser les performances de différents types de systèmes [6]. Pour permettre de modéliser un système utilisant la blockchain, au moyen de la théorie des files d'attente, il est nécessaire, d'après [12], d'observer certains mécanismes clés qui régissent le fonctionnement de ce système tels que la construction de la chaîne de blocs, le processus d'arrivée des transactions, le processus de génération des blocs, le processus de synchronisation des blocs, les frais de transaction, etc. Ce sont des mécanismes importants dont il faudra tenir compte pour modéliser correctement un système s'appuyant sur la technologie blockchain [12].

L'article [8] explique également que tout système utilisant la technologie blockchain peut être évalué avec des finalités très différentes en se focalisant sur l'un ou l'autre aspect particulier de son fonctionnement. Par exemple, un même système peut être évalué du point de vue de la sécurité, de la performance, de la disponibilité, etc. L'intérêt de ce travail porte, principalement, sur l'évaluation des performances de tels systèmes. Les principales mesures, relatives à la performance de ces systèmes, pourraient être regroupées, d'après la proposition faite par l'auteur de l'article [8], en trois catégories selon les parties qui composent le système :

- Mesures de performance liées au réseau
- Mesures de performance liées aux nœuds composant le réseau
- Mesures de performance liées à la blockchain

Chaque mesure permettant d'évaluer, du point de vue de la performance, le fonctionnement d'un système mettant en œuvre la technologie blockchain, peut être issue de l'une des trois catégories précédentes.

Le présent travail se concentre sur les modèles de files d'attente dans le but d'évaluer les performances des systèmes utilisant la blockchain tel que le système Bitcoin. Pour ce type de systèmes, l'approche basée sur la simulation est relativement peu abordée dans la littérature comparée à l'approche analytique, d'après les auteurs de l'article [6] faisant référence à un autre article [13]. C'est la principale motivation de ce travail. En effet, la simulation du système étudié sera développée dans la dernière partie de ce mémoire.

Dans la suite de ce chapitre, quelques modèles de files d'attente vont être présentés afin d'analyser les différents processus qui ont conduit à ces modèles et tenter de mettre en évidence des aspects importants qui permettront de construire la discussion concernant le modèle étudié. Mais avant cela, il est d'abord utile d'introduire ce qu'est un modèle de file d'attente.

3.3 SYSTÈME DE FILE D'ATTENTE

Un système de file d'attente, plus communément appelé file d'attente, permet de représenter la dynamique du comportement d'un système ou d'une partie relative à un mécanisme spécifique de ce système afin d'évaluer ses performances. Le but de cette section est d'introduire le vocabulaire utile pour la compréhension des modèles de file d'attente qui vont être présentés dans la suite de ce chapitre.

D'un point de vue fonctionnel, des entités arrivent dans un système de file d'attente pour recevoir un service auprès d'autres entités qui composent le système. Dès que le service se termine, elles quittent le système individuellement ou par groupes. Les entités qui arrivent dans le système sont généralement appelées des « clients » tandis que celles qui fournissent un service sont appelées des « serveurs ». Ce sont des termes génériques qui peuvent représenter différentes entités suivant les domaines d'applications étudiés [14]. Dans le contexte de cette étude, les clients peuvent être des blocs ou des transactions alors que les serveurs, faisant partie intégrante du système de file d'attente, représentent des nœuds du réseau Bitcoin, et plus particulièrement des mineurs. Lorsqu'un client arrive dans le système à un moment où le serveur est encore occupé, il peut être mis en attente dans une file jusqu'à ce que le serveur se libère. Une bonne analogie pour comprendre ce fonctionnement est l'arrivée de clients dans une banque pour effectuer des opérations au guichet. Lorsque ce dernier est occupé, le client attend son tour et quitte la banque dès qu'il termine son opération.

La structure de base d'un système de file d'attente, comme le montre la figure 8, peut être caractérisée par les aspects suivants [14] :

- Le processus d'arrivée
- Le nombre et la disposition des serveurs
- La capacité du système
- La discipline de service
- Le temps de service

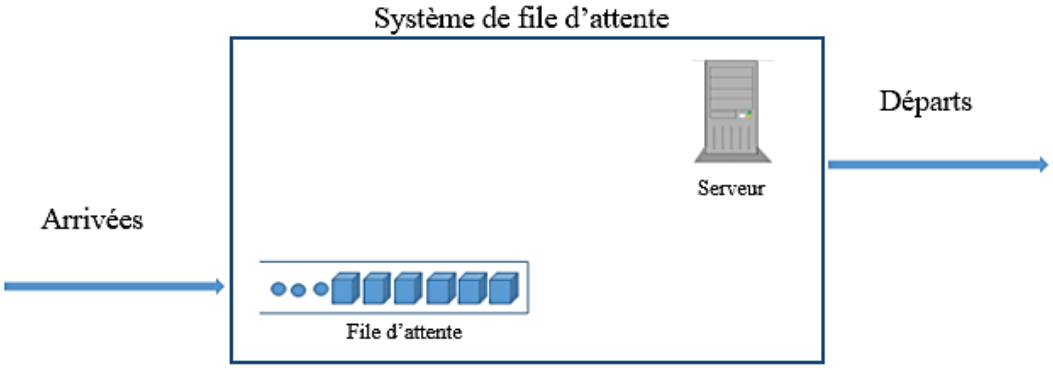


Figure 8 - Le système de file d'attente

Le processus d'arrivée

Le processus d'arrivée décrit la manière avec laquelle les clients arrivent dans le système de file d'attente. Ces derniers peuvent arriver de manière régulière, c'est-à-dire à des instants précis qui peuvent être déterminés par un taux d'arrivée. Mais les clients peuvent aussi arriver de manière aléatoire dans le système. Dans ce cas, le flux d'arrivée peut être décrit par un processus stochastique, le plus souvent par un processus de Poisson (voir chapitre 3).

Nombre et disposition des serveurs

Le système de file d'attente peut être composé d'un ou plusieurs serveurs pouvant être disposés en parallèle ou en série. Les clients peuvent recevoir leur service les uns à la suite des autres auprès d'un seul serveur, simultanément sur des serveurs en parallèles ou en plusieurs étapes sur des serveurs en série. Les serveurs peuvent également être soumis aux pannes, c'est-à-dire qu'ils peuvent arrêter de travailler pendant un certain temps.

La capacité du système

La capacité du système de file d'attente est un autre élément intéressant qui correspond au nombre de clients que celui-ci est capable d'accueillir en même temps. Elle englobe aussi bien les clients en attente que ceux qui sont en service et peut être limitée ou pas.

La discipline de service

La discipline de service décrit la manière avec laquelle les clients en attente dans le système sont sélectionnés pour recevoir leur service. Les clients qui attendent dans une file peuvent être servis en fonction de leur ordre d'arrivée ou dans l'ordre inverse. Une autre discipline consiste à servir les clients, en attente dans une file, en fonction de leur priorité ou encore de manière aléatoire.

Le temps de service

Le temps de service détermine la durée de service nécessaire à un serveur pour le traitement d'un client. Généralement, chaque client peut avoir un besoin de service différent, autrement dit une durée de service différente. Par conséquent, les temps de service peuvent aussi être décrits par une distribution de probabilité, le plus souvent par une loi exponentielle (voir chapitre 3).

A présent, il est intéressant de passer en revue quelques modèles de files d'attente. Dans un premier temps, quelques modèles avec un seul serveur vont être présenté, puis un modèle avec une infinité de serveurs sera étudié plus en détail.

3.4 MODÈLES À UN SEUL SERVEUR

3.4.1 Aspects de la modélisation du processus de minage

Les premiers travaux intéressants, extraits de la littérature scientifique, relatifs à la modélisation de systèmes implémentant la blockchain, tel que le Bitcoin, au moyen de la théorie des files d'attente, sont ceux réalisés par les auteurs de l'article [15]. Le processus de minage (voir chapitre 1) a été modélisé, dans cet article, au moyen d'un système de file d'attente. Le but étant d'étudier le temps de confirmation des transactions. D'après l'article [16], les transactions dans les systèmes utilisant la blockchain sont confirmées avec des temps beaucoup trop longs (en minutes) par rapport aux systèmes de paiement habituels (en secondes). Ce délai de confirmation, relativement important, constitue un véritable frein à l'évolutivité des systèmes de crypto-monnaies tel que le Bitcoin. Comme expliqué dans la présentation du système Bitcoin, une transaction n'est considérée comme confirmée qu'à partir du moment où elle est incluse dans un bloc valide. Le temps que cela peut prendre varie en fonction de différents paramètres dont les frais contenus dans les transactions et la taille maximum des blocs. Cette étude permet de montrer dans quelle mesure la taille d'un bloc

influence le temps de confirmation des transactions. En effet, la taille d'un bloc est initialement limitée, dans le protocole de Bitcoin, pour des raisons de sécurité. Ce qui signifie que le nombre de transactions pouvant être incluses dans un bloc est également limité par la taille de ce dernier. De plus, selon le protocole de Bitcoin, l'intervalle de temps qui sépare la génération des nouveaux blocs est régulièrement maintenu à plus ou moins un bloc toutes les dix minutes. Par conséquent, la taille limite d'un bloc influence directement le temps de confirmation des transactions. Et le fait d'augmenter la taille limite d'un bloc n'est pas nécessairement une bonne solution. Avec des blocs de taille plus grande, le temps de confirmation diminue puisqu'un plus grand nombre de transactions peuvent être incluses dans un même bloc, mais les temps de traitement par les différents nœuds du réseau augmentent également. De ce fait, le délai de propagation de ces blocs augmente en conséquence, c'est-à-dire qu'ils se propagent moins rapidement entre les nœuds du réseau [17]. Une telle situation pose un réel problème de sécurité car il peut arriver qu'une partie des nœuds du réseau ne reçoivent pas encore un bloc en circulation alors que le suivant est déjà généré. Ainsi, la probabilité d'obtenir un embranchement dans la chaîne de blocs augmente à cause de ce délai de propagation. Autrement dit, les différentes copies locales de la blockchain ne sont plus complètement synchronisées entre elles durant un certain moment [18]. En conséquence, le risque d'attaque par double dépense devient plus probable, ce qui fragilise la stabilité du système Bitcoin.

3.4.2 Aspects de la modélisation du processus de confirmation des transactions

Une étude [19] semblable à la précédente [15], dans laquelle les auteurs se sont intéressés à l'influence des frais contenus dans les transactions sur le temps de confirmation de celles-ci. Leur travail a été motivé par l'évolution du système Bitcoin qui pourrait connaître une augmentation du nombre de transactions avec de très petits montants en raison du faible coût de transfert comparé aux systèmes de paiement classiques. Les frais proposés par ce type de transactions pour inciter les mineurs à les inclure dans des blocs sont, raisonnablement, susceptibles d'être peu élevés compte tenu de leurs faibles montants. Or les frais contenus dans les transactions ont une influence sur la priorité de leur traitement. En effet, les mineurs peuvent être intéressés par les transactions les plus rentables, celles qui proposent le plus de frais, afin d'amortir leurs coûts de participation au minage et tenter d'en retirer un bénéfice. Les observations réalisées par les auteurs de l'étude [16] montrent que sur un nombre de transactions examinées durant une certaine période de temps, les frais moyens contenus dans les transactions confirmées sont bien plus importants que ceux des transactions qui n'ont pas pu être confirmées durant cette même période. Et toujours selon cette même étude [16], plus le montant d'une transaction est important, plus sa probabilité d'être confirmée durant un certain intervalle de temps augmente en raison d'un lien qui peut exister entre ce montant élevé et les frais que cette transaction peut contenir. Il est donc important de comprendre l'impact que peut avoir une augmentation des transactions contenant des faibles frais sur leur processus de

confirmation. La taille maximale des blocs contenant les transactions est un autre aspect important qui a été pris en compte par cette étude [19].

Concrètement, lorsqu'une transaction arrive dans un nœud du réseau Bitcoin, elle est directement traitée mais n'est pas encore considérée comme valide. Elle est stockée localement par le nœud qui la reçoit, après avoir été validée, en attendant la génération d'un nouveau bloc. Pendant ce temps, les autres transactions qui arrivent à ce nœud continuent de subir le même traitement. Dès qu'un nouveau bloc est généré, les transactions sont insérées dans celui-ci en fonction de l'importance de leurs frais et tant que le remplissage ne dépasse pas la taille limite du bloc. Les transactions contenues dans ce nouveau bloc valide sont alors considérées comme confirmées et leur traitement terminé. Le bloc est ensuite propagé aux autres nœuds du réseau. Il s'agit donc d'un traitement groupé car une transaction n'est pas traitée individuellement, mais c'est un ensemble de transactions qui sont traitées pour être incluses dans un bloc valide afin d'être confirmées.

Les auteurs de cette étude modélisent ce comportement par un système de file d'attente à un seul serveur, représentant un mineur, avec un service par lots, correspondant au traitement d'un ensemble de transactions à insérer dans un bloc valide, par un mineur, pour confirmation. Le modèle proposé utilise également un mécanisme de priorité pour tenir compte des frais pouvant être contenus dans les transactions. Concernant ce dernier aspect, les auteurs classent les transactions dans deux catégories en fonction du montant de leurs frais et à chacune de ces catégories est attribué un niveau de priorité : faible ou élevée. Les transactions contenant des montants de frais relativement faibles, c'est-à-dire n'atteignant pas la valeur d'un dix-millième de bitcoin, sont groupées dans la catégorie faible ou, à l'inverse, dans la catégorie élevée. Les transactions sont servies en tenant compte de leur niveau de priorité dans le système de file d'attente, c'est le mécanisme de priorité.

Le processus de confirmation des transactions dans le système Bitcoin est donc modélisé, dans cette étude [19], par un système de file d'attente à un seul serveur avec un service par lots utilisant un mécanisme de priorité pour servir les transactions. Par hypothèse, la priorité d'une transaction dépend seulement des frais qu'elle contient et rien d'autre. De plus, le temps de service est supposé être, dans cette étude, équivalent au temps de génération d'un bloc dans le système réel. Le modèle permet donc d'étudier le temps de confirmation moyen pour chacune de ces catégories de transactions.

Les résultats ainsi obtenus par ce modèle montrent que pour une taille de bloc donnée, le temps de confirmation pour les transactions contenant des faibles frais devient relativement important avec l'augmentation de leur taux d'arrivée. Autrement dit, dès que le taux d'arrivée de ce type de transactions quadruple, leur temps de confirmation augmente de manière considérable. En outre, bien que l'augmentation de la taille limite des blocs permet, effectivement, de réduire le temps de confirmation des transactions, cette solution s'avère inefficace lorsque le taux d'arrivée des transactions augmente. Par conséquent, une telle solution n'est pas vraiment intéressante pour réduire le temps de confirmation des transactions

dans le cas de l'évolutivité du système Bitcoin. Les figures 9 et 10 permettent de mieux comprendre l'influence du mécanisme de priorité sur le temps de confirmation. Le graphique de droite (Fig. 9) correspond aux transactions avec de faibles frais tandis que celui de gauche (Fig. 10) correspond aux transactions avec des frais plus élevés. Le rapport des taux d'arrivées des deux catégories de transactions reste constant tandis que le taux d'arrivée global, quant à lui, varie. En d'autres termes, les taux d'arrivées varient dans les mêmes proportions. Les graphiques montrent que le mécanisme de priorité influence surtout les transactions avec des faibles frais. Les différentes courbes, sur chaque graphique, correspondent à des tailles de blocs différentes permettant de voir l'influence de la taille limite des blocs sur le temps de confirmation des transactions.

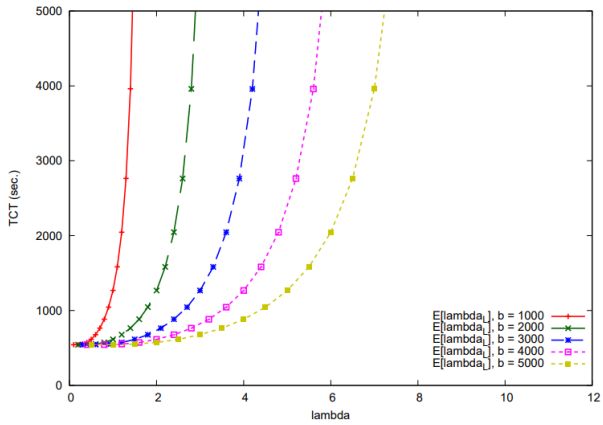


Figure 9 - Temps moyen de confirmation des transactions : cas de la priorité élevée. [4, Fig. 7]

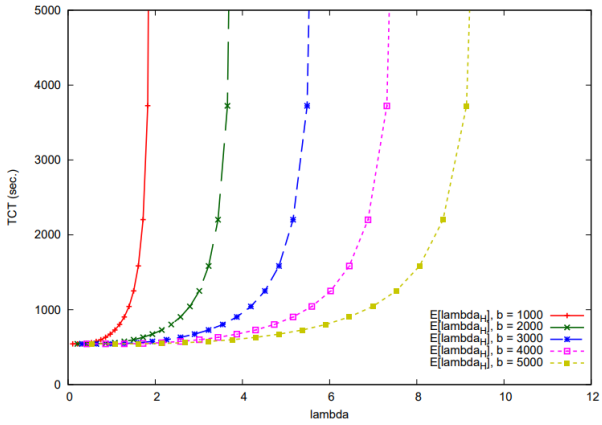


Figure 10 - Temps moyen de confirmation des transactions : cas de la priorité faible. [4, Fig. 8]

Dans les études précédentes, les auteurs ne s'intéressent qu'au traitement des transactions, en construisant des modèles dans le but d'analyser le temps de confirmation de ces transactions. D'autres études s'intéressent également au traitement des blocs (voir chapitre 1) en proposant des modèles permettant d'étudier leur comportement. Ces modèles vont maintenant être présenté dans la suite de ce chapitre.

3.4.3 Processus de confirmation des transactions en deux étapes

L'article [13] propose d'étudier le processus de confirmation des transactions de manière plus détaillée, permettant ainsi de compléter les deux études précédentes. Les auteurs de cet article [13] décomposent le processus de confirmation des transactions en deux phases bien distinctes afin d'analyser le traitement de celles-ci avec plus de détails. Lorsqu'une nouvelle transaction arrive dans un nœud du réseau, elle est d'abord vérifiée par le mineur qui la reçoit avant de la stocker dans son pool de transactions. Elle reste en attente jusqu'à ce qu'elle soit sélectionnée pour être incluse dans un nouveau bloc miné. Le mineur qui génère un nouveau bloc peut y inclure l'ensemble des transactions qu'il a sélectionné pour ce bloc et compléter ensuite sa blockchain locale en lui rajoutant ce bloc. Il y a donc deux traitements différents, l'un concerne les transactions et l'autre concerne les blocs. Les auteurs de cette étude proposent alors un système de file d'attente permettant d'analyser à la fois le traitement des transactions et celui des blocs.

Le modèle proposé dans cet article [13] n'est pas très différent du précédent [19]. Il s'agit encore d'un système de file d'attente à un seul serveur, représentant un mineur dans le système. La différence se situe au niveau du service. En fait, dans ce modèle, le service par lots s'effectue en deux étapes successives permettant d'étudier le processus de confirmation des transactions de manière plus détaillée (Fig. 11). La première étape du service correspond au traitement des transactions, depuis leur vérification jusqu'à leur insertion groupée dans un nouveau bloc valide. Cette première phase du service est appelée, dans cette étude, le processus de génération des blocs. La deuxième étape du service correspond au traitement des blocs, depuis leur génération jusqu'à leur insertion dans la blockchain locale du mineur. Cette seconde phase du service est appelée, dans cette étude, le processus de construction de la blockchain.

Le temps de confirmation d'une transaction se compose alors du temps nécessaire pour générer un bloc incluant cette transaction et du temps qu'il faut pour que ce nouveau bloc soit ajouté à la blockchain. Les auteurs considèrent que le temps de service, dans chacune des étapes, suit une loi de distribution exponentielle avec des taux de service différents et que les transactions arrivent selon un processus de Poisson. Ces notions seront expliquées dans le chapitre 3.

La capacité de la file d'attente n'est pas limitée, c'est-à-dire que toutes les transactions qui arrivent dans le système pourront être traitées, aucune d'entre elles n'est rejetée. Lorsque le serveur est occupé, elles sont mises en attente dans une file jusqu'à ce qu'elles puissent être servies. Mais dans ce modèle, les transactions ne sont pas nécessairement traitées en fonction de l'ordre d'arrivée. Enfin, la transaction qui accède au service est traitée en deux étapes successives comme illustré à la figure 11.

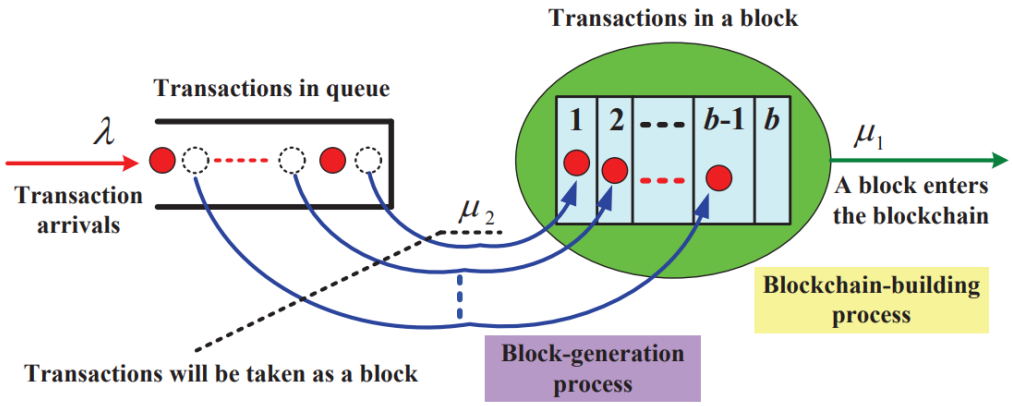


Figure 11 - Modèle de file d'attente. [13, Fig. 1]

Le modèle permet ainsi d'obtenir des expressions pour l'évaluation, dans un régime stationnaire, des trois mesures de performance suivantes :

- Le nombre moyen de transactions dans la file d'attente
- Le nombre moyen de transactions dans un bloc
- Le temps de confirmation moyen d'une transaction

Finalement les auteurs utilisent quelques exemples chiffrés afin de prouver la calculabilité de ces différentes expressions dans l'évaluation des performances de systèmes basés sur la blockchain tel que Bitcoin.

L'article de [20], est une étude semblable à la précédente [13]. L'auteur, qui s'intéresse également au temps de traitement des transactions, confirme une information importante déjà mise en évidence dans l'un des articles précédents [7]. En effet, selon l'auteur, la littérature fait apparaître que la plupart des recherches actuelles s'intéressent, généralement, aux applications et développements de la blockchain dans différents domaines. Par conséquent, les études qui s'intéressent aux modèles mathématiques et à l'analyse des performances de systèmes basés sur la blockchain sont beaucoup moins nombreuses.

Dans cette étude, l'auteur propose également un modèle de file d'attente afin d'évaluer le temps de traitement des transactions. L'étude est semblable à celle de [13] avec un seul serveur représentant un mineur. Le processus de départ se compose de deux étapes distinctes. La première phase représente l'insertion des transactions dans un nouveau bloc, tandis que la seconde phase représente l'insertion du bloc dans la blockchain. La conclusion de cette étude montre que, d'une part, le temps d'attente d'un bloc, avant son insertion dans la blockchain, est inversement proportionnel au nombre de blocs, et d'autre part, le temps de traitement d'un

bloc est indépendant du nombre de transactions qu'il contient. La figure 12 montre que plus le nombre de blocs augmente, plus le temps d'attente pour l'insertion d'un bloc dans la blockchain diminue (Fig. 12). Concrètement, cela signifie que le mineur qui génère un nouveau bloc doit d'abord obtenir le consensus des autres nœuds du réseau avant de pouvoir l'insérer dans sa blockchain locale. Par conséquent, plus le nombre de nouveaux blocs augmente, plus le nombre de validations pour l'insertion de ces derniers dans la blockchain augmente également.

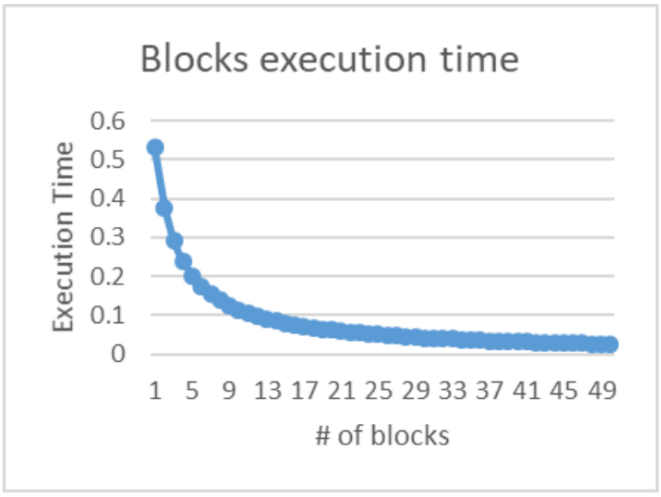


Figure 12 - Insertion des blocs dans la blockchain. [20, Fig. 3]

Les modèles précédents s'intéressent plus particulièrement à la circulation des transactions dans le réseau Bitcoin. Or, les transactions ne sont pas les seuls éléments en mouvement dans le réseau, il y a aussi les blocs. Ces derniers circulent de la même manière que les transactions entre les nœuds du réseau. C'est pour cela que les études qui vont suivre s'intéressent plus particulièrement au traitement des blocs dans le système Bitcoin.

3.5 PROCESSUS D'ARRIVÉE DES BLOCS DANS LA BLOCKCHAIN

Selon l'étude réalisée par les auteurs de l'article [21], la littérature considère souvent que les blocs de transactions arrivent dans la blockchain selon un processus de Poisson homogène. Ils montrent que c'est une hypothèse qui est très fréquemment adoptée dans la littérature. Les auteurs affirment, également, qu'il y a peu de recherches qui se sont intéressées au processus d'arrivée des blocs dans la blockchain pour voir s'il s'agit vraiment d'un processus de Poisson homogène. Ils pensent que ce n'est pas systématiquement le cas et réalisent alors une étude [21] s'intéressant à ce processus pour tenter de comprendre et d'analyser cette hypothèse. Le résultat de cette étude montre que le processus d'arrivée des blocs dans la blockchain n'est pas toujours un processus de Poisson homogène, c'est-à-dire qu'il y a des intervalles de temps où ce n'est plus un processus de Poisson. Les auteurs proposent donc un modèle mathématique, qui prend en considération le caractère variable de la difficulté de minage (voir chapitre1), permettant de représenter le processus d'arrivée des blocs de manière plus précise au cours du temps.

3.6 PROPAGATION DES BLOCS DANS LE RÉSEAU

Dans l'étude réalisée par [22], l'auteur ne fournit pas de modèle de file d'attente mais explique l'effet du retard de propagation des blocs sur la blockchain. En effet, l'augmentation du délai de propagation d'un bloc dans le réseau Bitcoin augmente aussi la probabilité d'obtenir un embranchement dans la blockchain (voir chapitre 1). Par conséquent, cela augmente également la probabilité d'une attaque réussie par des participants qui décideraient d'adopter un comportement frauduleux.

Les modèles de files d'attente présentés dans les articles précédents ne se composent que d'un seul serveur représentant un mineur du réseau. Ils permettent d'étudier des mécanismes de fonctionnement localisés aux nœuds tel que le traitement d'une transaction par un mineur par exemple. Mais pour des mécanismes impliquant des échanges entre les différents nœuds du réseau, telle que la synchronisation de la blockchain par exemple, il est nécessaire d'utiliser des modèles de file d'attente avec plusieurs serveurs. C'est pourquoi les articles qui vont suivre ont proposé des modèles avec un nombre plus important de serveurs, théoriquement une infinité de serveurs.

3.7 MODÈLE AVEC UNE INFINITÉ DE SERVEURS

L'étude de Brian Fralix [2] se base sur un article précédent [23], une étude similaire réalisée par Frolkova et M. Mandjes concernant un modèle de files d'attente inspiré du système Bitcoin. Les auteurs de cette étude se sont intéressés au mécanisme de synchronisation de la blockchain à travers le réseau Bitcoin. En effet, comme expliqué dans le premier chapitre, certains nœuds du réseau disposent, localement, d'une copie complète de la blockchain. Il peut arriver, par moments, que ces copies ne soient pas toutes identiques entre elles. Cela peut être dû au délai de propagation des blocs de transactions dans le réseau ou lorsque certains mineurs interrompent leur participation de manière provisoire. Il en résulte alors une désynchronisation temporaire du réseau. Or l'intérêt du système Bitcoin est justement de garantir que tous les participants puissent disposer d'une version authentique de la blockchain. Il est donc primordial que le réseau se resynchronise le plus rapidement possible. La réalisation de cette étude est ainsi justifiée par la nécessité de mieux comprendre le mécanisme de la désynchronisation du système provoquée, principalement, par le délai de propagation des blocs contenant les transactions.

Dans le fonctionnement du système réel, les blocs de transactions se propagent de proche en proche entre les différents nœuds du réseau. Les mineurs participent également à la production de nouveaux blocs et valident ceux qu'ils reçoivent des nœuds voisins avant de les propager. Chaque bloc contient une référence à un autre bloc qui est son prédécesseur dans la chaîne de blocs. Ainsi, les blocs se lient entre eux selon un ordre précis pour permettre de construire la blockchain authentique. Lorsqu'un mineur reçoit un bloc valide, il tente de compléter sa copie locale de la blockchain. Mais il peut arriver qu'un bloc reçu ne puisse pas directement être ajouté à la blockchain car il manque encore des blocs intermédiaires. C'est une situation normale et fréquente en raison des délais de propagation dans le réseau, certains blocs peuvent arriver avant celui qui est attendu, mais à condition qu'elle ne dure pas trop longtemps. A partir des liens qui existent entre les blocs, le mineur peut déduire les blocs manquants pour la construction de sa blockchain locale. Il va alors tenter de récupérer tous les blocs manquants en établissant une communication avec d'autres mineurs. Ce n'est que lorsque tous les blocs manquants auront été récupérés que la blockchain locale pourra être complétée, en une seule fois, par l'ensemble de ces blocs. C'est de cette manière que les différentes copies locales de la blockchain se synchronisent entre elles.

Les auteurs de cette étude [23] modélisent ce mécanisme de synchronisation au moyen d'une file d'attente avec une infinité de serveurs, correspondants aux mineurs, en raison du nombre important de nœuds constituant le réseau. Les arrivées de blocs dans le système de file d'attente, appelées des clients dans ce modèle-ci, représentent les nouveaux blocs de transactions générés par les mineurs et qui se propagent entre les nœuds du réseau. Les auteurs considèrent un échange de blocs entre deux parties du réseau Bitcoin. L'article [24] précise que les parties concernées sont des mineurs, c'est-à-dire qu'ils sont tous capables de générer des nouveaux blocs et de les échanger entre eux. Mais par hypothèse, dans ce modèle-ci, les auteurs supposent que l'échange de blocs ne s'effectue que dans un seul sens. Autrement dit, une partie génère ou valide des blocs et les propage sur le réseau. Cela correspond aux arrivées de clients dans le modèle de file

d'attente. L'autre partie reçoit les nouveaux blocs générés mais avec un certain retard dû à leur délai de propagation. Ensuite, elle traite les blocs reçus pour les rajouter à sa blockchain locale. Cela se traduit, dans le modèle de file d'attente, par le service fourni aux clients lorsque ces derniers arrivent sur des serveurs parallèles. Le nombre de serveurs étant tellement élevé qu'il n'y a pas d'attente dans le système, les clients sont directement servis. Les auteurs supposent également que la partie émettrice possède toujours plus de blocs que la partie réceptrice, ce qui signifie que leurs blockchains locales respectives sont régulièrement désynchronisées même si par moments, durant les échanges, elles parviennent à les resynchroniser de manière temporaire. Le modèle tient compte de ce comportement.

Les liens qui existent entre les blocs dans le système réel et le mécanisme qui permet de construire la blockchain se traduisent dans le modèle de file d'attente par un fonctionnement très particulier : les clients interagissent les uns avec les autres. En fait, dans le cas d'une file d'attente classique avec plusieurs serveurs, les clients sont indépendants et quittent le système l'un après l'autre à chaque fois qu'un service s'achève. Or, dans le modèle présenté, les clients quittent le système de manière groupée à cause de cette interaction qui existe entre eux. Autrement dit, les départs se produisent par lots. Dans l'article [24], l'auteur parle également de départs synchronisés.

Un dernier élément, relatif au fonctionnement de ce modèle, est la discipline de service. Les auteurs supposent que les clients qui arrivent dans le système s'alignent chacun sur un serveur libre en suivant l'ordre d'arrivée. Et dès qu'un client quelconque termine son service, il quitte le système avec tous ceux qui le précèdent dans le système de file d'attente. Cette hypothèse peut facilement se comprendre par le comportement des blocs dans le système réel. En effet, le mineur qui doit synchroniser la copie de sa blockchain locale est capable de déduire l'ensemble des blocs manquants dès la réception d'un premier bloc. Une fois que le dernier bloc manquant a été récupéré, c'est le premier bloc avec tous ses prédécesseurs qui sont ajoutés, en une seule fois, à la blockchain locale.

Les auteurs font également l'hypothèse d'un processus d'arrivées de renouvellement et des temps de service distribués de façon exponentielle et complètement indépendants.

Les résultats obtenus par l'étude de ce modèle sont de différents types. Premièrement, la distribution des périodes occupées, c'est-à-dire les intervalles de temps où le système n'est pas vide, ne dépend pas du taux d'arrivées alors qu'il serait plus logique de penser le contraire. Ensuite, d'autres résultats telle que la distribution de la taille de la file d'attente à un instant donné est obtenue sous l'hypothèse supplémentaire des arrivées de Poisson. Finalement, pour le dernier type de résultats, les auteurs utilisent la loi des grands nombres pour proposer une limite fluide à leur modèle.

Finalement, ce modèle est très intéressant car il a permis de mettre en évidence un premier concept important que sont les départs par lots caractérisant le fonctionnement particulier d'une file d'attente avec une infinité de serveurs ainsi son influence sur les résultats obtenus.

Pour conclure cette revue de la littérature, il est intéressant de souligner qu'elle a permis de mettre en évidence la faible proportion des modèles de simulation par rapport aux modèles analytiques. Autrement dit, très peu de modèles de files d'attente ont été simulés dans la littérature. C'est également le cas du modèle analytique concerné par cette étude. Par conséquent, c'est ce qui justifie la réalisation de ce travail.

3.8 LE MODÈLE EN COURS D'ÉTUDE

L'article scientifique, qui est le point de départ de ce mémoire, est l'étude présentée par Brian Fralix [2] qui propose un modèle particulier de files d'attente inspiré de la dynamique du système Bitcoin. Le but étant d'obtenir un modèle analytique pour l'évaluation de la performance de systèmes présentant des comportements identiques. Evidemment, des études similaires ont déjà été réalisées sur le même sujet mais avec des points de vue et des approches différentes. L'auteur se base sur une étude précédente réalisée par Frolkova et Mandjes [23], qu'il propose d'approfondir en utilisant d'autres outils, notamment la loi de Little, et en faisant des généralités.

3.8.1 Présentation du modèle de files d'attente étudié

Le modèle analytique proposé par Brian Fralix [2] est un système de files d'attente avec un nombre infini de serveurs dans lequel les clients interagissent les uns avec les autres, ce qui provoque des départs par lots. Le fait que les clients interagissent entre eux est une particularité de ce système puisque, dans le cas classique d'une file d'attente avec un nombre infini de serveurs, les clients agissent de manière indépendante et quittent le système un par un après chaque achèvement de service [23].

Le modèle de file d'attente avec une infinité de serveurs se justifie très bien pour les systèmes tel que le Bitcoin. En effet, étant donné la taille et l'architecture du réseau, le nombre de nœuds est très largement suffisant que pour permettre à un client d'être servi immédiatement à son arrivée. Autrement dit, le nombre de nœuds dans le réseau est tel qu'une transaction ou un bloc sera traité immédiatement à son arrivée dans le système. Par conséquent il n'y a pas de temps d'attente provoquée par un manque de disponibilité des serveurs.

Les clients, qui sont appelés dans ce contexte des « jobs », et pouvant être des transactions ou des blocs, arrivent dans le système de files d'attente les uns à la suite des autres, en suivant un processus d'arrivée. Lorsqu'un client arrive dans le système, il est directement dirigé vers un serveur disponible pour y être servi. Dans ce modèle, les serveurs sont ordonnés suivant un numéro d'ordre. Par conséquent le nouveau client arrivant sera dirigé vers le serveur libre dont le numéro d'ordre suit immédiatement celui du dernier serveur encore occupé. Il est important de préciser qu'initialement, à l'instant zéro, le système ne contient aucun client.

Les départs du système se produisent par lots, c'est-à-dire que les clients quittent le système par groupes à cause de l'interaction qui existe entre ceux-ci. C'est donc une particularité de ce type de systèmes puisque, dans la file d'attente classique, les clients quittent le système les uns à la suite des autres dès qu'un service se termine. Le lot quittant le système se compose du client dont le service vient d'être achevé, ainsi que de tous les clients en traitement sur les serveurs de numéro d'ordre inférieur à celui qui a provoqué ce départ. Immédiatement après ce départ, tous les clients qui se trouvaient sur des serveurs de numéro d'ordre supérieur à celui-ci sont transférés, dans le même ordre, vers les serveurs qui viennent de terminer leur service. Il s'agit d'un décalage des clients vers les serveurs inférieurs devenus libres. Ce système de gestion garde son sens car dans l'évaluation de performances, ce qui compte, c'est la quantité de travail, le temps de séjour d'un job et non l'identification du serveur qui a réalisé le travail. Evidemment, il s'agit ici d'une simplification du modèle car les nœuds du réseau n'ont pas tous la même puissance de calcul. Mais les hypothèses et les différentes simplifications seront discutées plus tard puisqu'il s'agit, dans cette partie, d'une présentation brute du modèle analysé. L'autre aspect important de ce modèle de file d'attente est la discipline de service qui est dite « FIFO-batch ». En effet, à cause de l'interaction qui existe entre les clients dans le système, les départs se font par lots en suivant l'ordre dans lequel ils sont arrivés. C'est une hypothèse qui fera également l'objet d'une discussion plus loin dans le développement de ce mémoire.

CHAPITRE 3

4. ANALYSE DU MODÈLE DE FILE D'ATTENTE ÉTUDIÉ

4.1 VARIABLE ALÉATOIRE

Une variable aléatoire, généralement notée avec une lettre majuscule, par exemple X, est une fonction qui prend des valeurs numériques pouvant varier en fonction du résultat d'une expérience [25, p. 12]. En d'autres termes, c'est une fonction qui lie chaque résultat possible d'une expérience aléatoire à une valeur numérique. Prenons l'exemple du lancer d'une pièce de monnaie. Dans ce cas, la variable aléatoire X est une fonction qui pourrait attribuer les valeurs entières 0 et 1 aux résultats possibles, pile ou face [26, p. 71].

Il est important de faire la distinction entre les variables aléatoires discrètes et continues.

- Une variable aléatoire continue est une variable qui peut prendre n'importe quelle valeur réelle dans un intervalle donné. Par conséquent, l'ensemble des valeurs possibles n'est pas dénombrable. Par exemple, le temps d'attente à un bureau de poste, durant les moments de forte affluence, avant d'être servi [26, p. 71, 73] [27, p. 138].

- Une variable aléatoire discrète est une variable dont l'ensemble des différentes valeurs possibles qu'elle peut prendre est dénombrable. Dans l'exemple précédent de la pièce de monnaie, les valeurs possibles de la variable aléatoire discrète sont {0,1} [26, p. 72] [27, p. 58].

4.2 DISTRIBUTION EXPONENTIELLE

La distribution exponentielle est une loi de probabilité très importante et très fréquente dans les modèles de files d'attente. Il est donc nécessaire de la présenter puisqu'elle est régulièrement utilisée dans le cadre de ce travail.

Dans le système de file d'attente classique M/M/1 (voir nomenclature section 3.4), les arrivées sont, généralement, supposées suivre un processus de Poisson (section 3.3). Les instants qui séparent les différentes arrivées successives suivent alors une distribution exponentielle. De la même manière, le temps de service est, également, supposé suivre une distribution exponentielle. Les variables aléatoires décrivant ces services sont indépendantes les unes des autres et indépendantes du processus de Poisson régissant les arrivées.

Il est donc essentiel de donner une définition de la distribution exponentielle.

Si X est une variable aléatoire qui suit une distribution exponentielle de paramètre λ (avec λ réel strictement positif), alors sa fonction de densité de probabilité est définie comme suit :

$$f(x) = \begin{cases} \boldsymbol{\lambda e^{-\lambda x}} & \text{pour} \quad x \geq 0 \\ \mathbf{0} & \text{pour} \quad x < 0 \end{cases} \qquad (\lambda > 0) \qquad [28, \text{p. } 77]$$

La fonction de densité pour une variable aléatoire continue peut s'interpréter de la manière suivante. La quantité *f(x)dx* est la probabilité que la variable aléatoire *X* prenne une valeur dans l'intervalle [*x*, *x*+*dx*]. Et sa représentation graphique est la suivante (Fig. 13) :

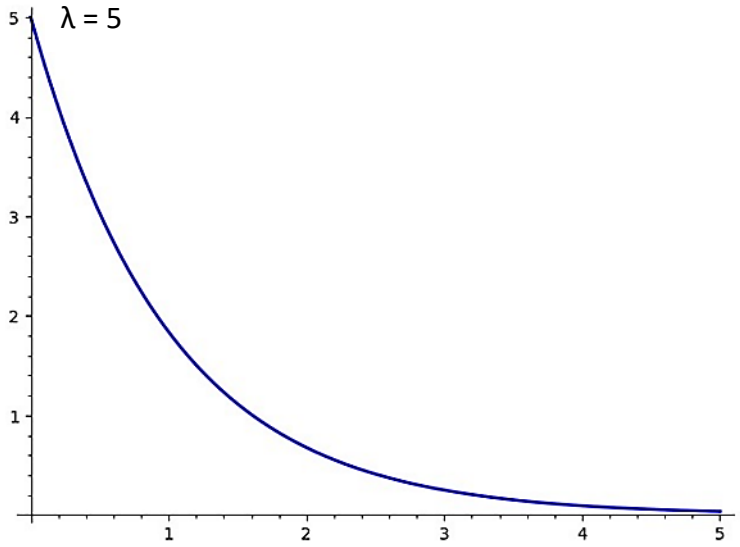


Figure 13 - Fonction de densité

Par définition, la surface totale sous la courbe de la fonction de densité de probabilité vaut 1.

L'espérance *E(x)* de cette variable aléatoire *X*, c'est-à-dire sa valeur moyenne, est donnée par la relation suivante :

$$\mathrm{E(X)} = 1/\lambda \qquad [14, \text{p. } 15]$$

4.3 PROCESSUS DE POISSON

Un processus de Poisson est un processus stochastique, c'est-à-dire un ensemble de variables aléatoires, qui est fréquemment utilisé dans les systèmes de files d'attente, plus particulièrement pour modéliser le processus d'arrivée en spécifiant les instants d'arrivée [27, p. 72] [14, p. 59].

Le processus d'arrivée décrit de différentes manières les arrivées au fil du temps en spécifiant soit les instants d'arrivée, soit les durées inter-arrivées ou soit, encore, via un processus de comptage d'arrivées. La figure suivante (Fig.14) permet de mieux comprendre cela.

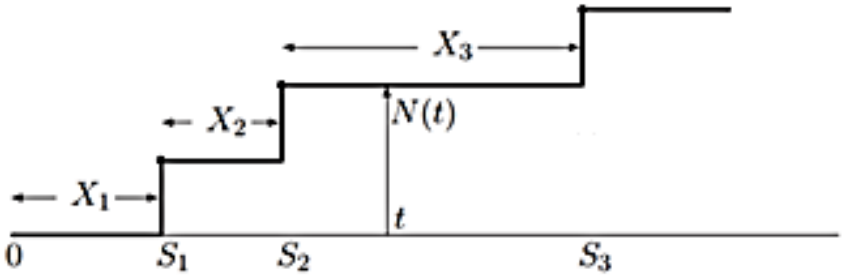


Figure 14 - Processus d'arrivée. [27, p. 2.1]

Sur ce graphique, on a donc trois collections de variables aléatoires à savoir :

- Les instants d'arrivée $\{S_n, n \in N_0\}$. Le processus démarre initialement au temps zéro, puis les instants se succèdent à chaque nouvel événement d'arrivée. Il ne peut y avoir qu'une seule arrivée à un instant donné.

- Les instants inter-arrivées $\{X_i, i \in N_0\}$, temps qui séparent les instants entre les arrivées successives S_n. La relation suivante permet de déterminer les instants de chaque arrivée :

$$S_n = \sum_{i=1}^{n} X_i \qquad \text{[27, p. 73]}$$

- Le processus de comptage $\{N(t), t \geq 0\}$, qui est une autre manière de spécifier le processus d'arrivées. Dans ce cas-ci, le processus permet de comptabiliser le nombre d'arrivées dans un intervalle de temps [0, t), en supposant qu'initialement, au temps zéro, le système est vide.

Le processus de comptage est un processus de Poisson lorsque les conditions suivantes sont remplies [14, p. 59] :

- « *La probabilité d'avoir un nombre d'arrivées dans un intervalle de temps donné ne dépend que de la longueur de celui-ci* ».
- « *Le nombre d'arrivées dans des intervalles disjoints est statistiquement indépendant* ».

Par conséquent, le nombre d'arrivées durant l'intervalle [0, t) est une variable aléatoire discrète qui suit une distribution de Poisson de paramètre $\lambda>0$ dont la fonction de probabilité est donnée par la relation suivante :

$$P[N(t) = n] = \frac{(\lambda t)^n e^{-\lambda t}}{n!} \qquad \text{[27, p. 79]}$$

Cette relation donne la probabilité d'observer n événements ($n \geq 0$), dans ce cas-ci n arrivées, durant un intervalle de temps t donné, correspondant à la période d'observation.

Dans cette relation, λ représente le taux d'occurrences, c'est-à-dire le nombre moyen d'observations d'un événement par unité de temps. Dans le processus d'arrivée, les événements observés sont des arrivées, ce qui signifie que λ représente le taux d'arrivée ou le nombre d'arrivées par unité de temps.

Sur une période d'observation t donnée, λt représente la moyenne de la variable aléatoire *N(t)*, c'est-à-dire le nombre d'occurrences moyen de l'événement durant cette période d'observation [14, p. 59].

Pour nos simulations, il est plus pratique de décrire un processus de Poisson en spécifiant les instants inter-arrivées. En effet, si le nombre d'arrivées durant un intervalle de temps donné suit un processus de Poisson alors les longueurs de temps X_i entre les arrivées suivent une distribution exponentielle de paramètre λ [14, p. 64], donnée par la relation suivante :

$$P[X_i \leq t] = 1 - e^{-\lambda t} \qquad (\forall i \in N_0) \qquad \text{[14, p. 64]}$$

Un processus de Poisson homogène est un processus dans lequel le taux d'arrivée reste constant au cours du temps. Lorsque ce taux varie au cours de la période d'observation selon une fonction $\lambda(x)$, pour $x \in R_0^+$, le processus est dit processus de Poisson non-homogène, lequel permet de mieux décrire le système réel. Mais dans le cadre de travail, c'est uniquement le processus de Poisson homogène qui est utilisé.

4.4 NOTATION DE KENDALL

La notation de Kendall permet de classer et de décrire les systèmes de files d'attente, de manière formelle, en fonction de leurs caractéristiques. Il s'agit d'un ensemble de lettres séparées par un symbole et décrivant, chacune, une caractéristique du système de file d'attente. Elle se présente comme suit [14, p. 50] :

« *A* / *B* / *X* / *Y* / *Z* » [14, p. 50]

Dans cette notation, les lettres peuvent prendre différentes valeurs et désignent les caractéristiques (voir chapitre 2) suivantes :

A : le processus d'arrivée des clients dans le système

B : la distribution des temps de service

X : le nombre de serveurs composant le système

Y : la capacité du système

Z : la discipline de service

En utilisant cette notation, le système de file d'attente étudié est alors décrit de la manière suivante : G/M/∞ ou M/G/∞ signifiant un système de file d'attente avec une distribution générale ou exponentielle des instants inter-arrivées, une distribution exponentielle ou générale des temps de service, une infinité de serveurs et, comme particularité dans notre étude, des départs par lots. Les deux dernières lettres ne sont pas indiquées dans la plupart des cas car elles prennent des valeurs par défaut.

4.5 CARACTÉRISTIQUES DU MODÈLE DE FILE D'ATTENTE ÉTUDIÉ

Tous les éléments nécessaires à la compréhension de la modélisation par file d'attente ont été détaillés dans le chapitre précédent. A présent, il est possible de présenter les caractéristiques du système de file d'attente en cours d'étude tel que le modèle est décrit par son auteur [2].

Les caractéristiques de ce modèle, illustré à la figure 15, sont les suivantes :

- Les clients sont appelés des « jobs » et représentent, dans le contexte de la synchronisation de la blockchain, des blocs de transactions. Cette appellation provient, probablement, du fait que chaque client qui arrive dans le système apporte avec lui une certaine quantité de travail pour le serveur. Pour être plus concret dans la suite de ce chapitre, les termes « jobs » et « clients » seront utilisés de manière indifférente pour désigner la même entité dans le système de file d'attente.

- Les jobs arrivent individuellement et de manière aléatoire dans le système selon un processus de Poisson de paramètre λ.

  - Le paramètre λ (lambda) est le taux d'arrivée, c'est-à-dire le nombre moyen de jobs qui arrivent dans le système par unité de temps.

  - L'inverse, c'est-à-dire 1/λ, correspond à l'intervalle de temps moyen qui sépare les arrivées successives de deux jobs dans le système.

  - Le processus d'arrivée est noté $A(t)$ avec $t \geq 0$ et représente le nombre de jobs arrivant dans le système pendant l'intervalle de temps [0, t].

  - Le nombre de jobs présents dans le système à un instant donné est noté $Q(t)$ et représente la différence entre les arrivées et les départs à un instant t avec $t \geq 0$.
    Initialement, le système est supposé vide. Autrement dit, à l'instant zéro le système ne contient aucun job : $Q(0) = 0$

- La capacité du système n'est pas limitée. En effet, vu la disponibilité des serveurs, tous les jobs qui arrivent dans le système sont directement dirigé vers des serveurs libres. Par conséquent, il n'y a pas d'attente dans le système et aucun blocage lié à la capacité du système de file d'attente. En d'autres termes, toutes les arrivées dans le système sont immédiatement prises en charge par des serveurs libres.

Temps d'attente moyen : $\overline{W} = 0$

- Le nombre de serveurs est infini. C'est un aspect important de ce modèle. En réalité, la disponibilité des serveurs est tellement importante par rapport à la demande de service qu'on considère qu'elle est infinie. Les serveurs sont alignés dans le système de file d'attente et chacun d'entre eux est identifié par son numéro d'ordre. Le premier serveur est appelé Serveur-1, les suivants Serveur-2, Serveur-3, Serveur-4, etc…

- Les serveurs sont disposés en parallèle et sont indépendants les uns des autres. Cela signifie qu'ils travaillent de manière à ce que tous les clients qui se trouvent dans le système à un moment donné soient servis simultanément.

- La discipline de service est une discipline de type « FIFO-batch ». C'est un autre aspect important de ce modèle provenant de l'interaction qui existe entre les jobs dans le système. Chaque job qui arrive dans le système s'aligne sur le serveur libre identifié par le plus petit numéro, en suivant l'ordre d'arrivée. Lorsqu'un job termine son service, il quitte le système avec tous ceux qui le précèdent dans la file. Les jobs restants dans le système subissent immédiatement une translation vers les serveurs de plus petits numéros d'ordre qui sont devenus libres, où ils peuvent continuer à recevoir leur service. Concrètement, supposons que le système soit composé de dix jobs, numérotés da la manière suivante : job-1, job-2, job-3, etc… Le job-1 reçoit son service auprès du Serveur-1, le job-2 auprès du Serveur-2 et ainsi de suite. Supposons maintenant que c'est le job-2 qui termine son service en premier. Il quitte alors le système avec le job-1 même si ce dernier n'a pas encore terminé son service. Immédiatement après que les deux serveurs soient devenus libres, le job-3 est transféré vers le Serveur-1, le job-4 vers le Serveur-2, le job-5 vers le Serveur-3, et ainsi de suite jusqu'au job-10 qui est transféré vers le Serveur-8. Lorsqu'un nouveau job arrivera dans le système, il sera directement dirigé vers le Serveur-9 qui est devenu le serveur libre avec le plus petit numéro d'ordre d'identification.

- Les temps de service suivent une distribution exponentielle de paramètre μ. Cette notion a été expliquée dans la première partie de ce chapitre. Les serveurs dans ce système travaillent avec un taux de service unitaire.
  - Le paramètre μ (mu) est le taux de service moyen, c'est-à-dire le nombre moyen de jobs qu'un serveur peut traiter par unité de temps.

  - L'inverse, c'est-à-dire $1/\mu$, correspond à la durée moyenne de service d'un job.

Le système complet est illustré à la figure suivante (Fig. 15) :

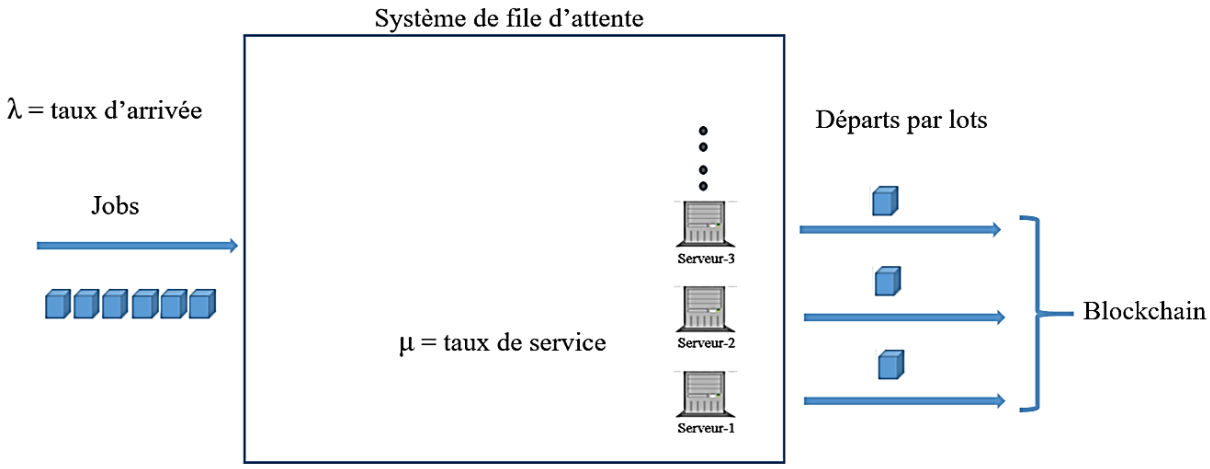


Figure 15 - Modèle de file d'attente avec serveurs infinis et départs par lots

4.6 PRINCIPALES MESURES DE PERFORMANCE

A partir du système de file d'attente précédent, il est possible d'évaluer certaines mesures de performance en simulant le modèle. Toutes ces mesures seront évaluées en régime stationnaire, en prenant une période de simulation suffisamment longue. Les principales mesures de performance qui sont étudiées dans le cadre de ce travail sont les suivantes :

- **Le nombre de jobs dans le système :**
  C'est le nombre total de jobs dans le système à un instant donné.

- **Le temps de séjour d'un job dans le système :**
  C'est le temps que passe un job dans le système depuis son arrivée jusqu'au moment de son départ.

- **La période occupée :**
  Représente la fraction de temps durant laquelle le système de file d'attente n'est pas vide. C'est aussi le complément de la période de temps durant laquelle le système est vide.

- **La probabilité d'avoir zéro jobs dans le système :**
  Représenté par la proportion de temps occupée par le système dans un état avec zéro job.

- **La probabilité d'avoir N jobs dans le système :**
  Représenté par la proportion de temps occupée par le système dans un état avec N jobs.

- **Le nombre de serveurs actifs dans le système :**
  C'est le nombre total de serveurs actifs à un instant donné. Cela correspond au nombre total de jobs au même instant puisque ces derniers sont directement servis à leur arrivée dans le système.

CHAPITRE 4

5. DÉVELOPPEMENT DE LA RECHERCHE

Ce dernier chapitre présente la méthodologie utilisée pour développement du modèle informatique tel qu'il a été spécifié dans le chapitre précédent. Le modèle sera ensuite simulé et les données recueillies seront utilisées pour l'analyse des performances.

Les différentes sections qui structurent ce chapitre présentent les parties suivantes :

- Une présentation générale de la simulation par événements discrets avec les caractéristiques d'un modèle et les principales approches de la simulation
- L'approche utilisée pour l'analyse du modèle informatique avec ses différentes étapes
- L'implémentation du modèle avec la présentation des choix techniques
- La vérification du modèle par un test d'hypothèse

5.1 SIMULATION PAR ÉVÉNEMENTS DISCRETS

5.1.1 Présentation générale

Les propriétés qui caractérisent le comportement de certains systèmes permettent de subdiviser la modélisation de ces derniers en différentes catégories comme le montre la figure suivante (Fig. 16).

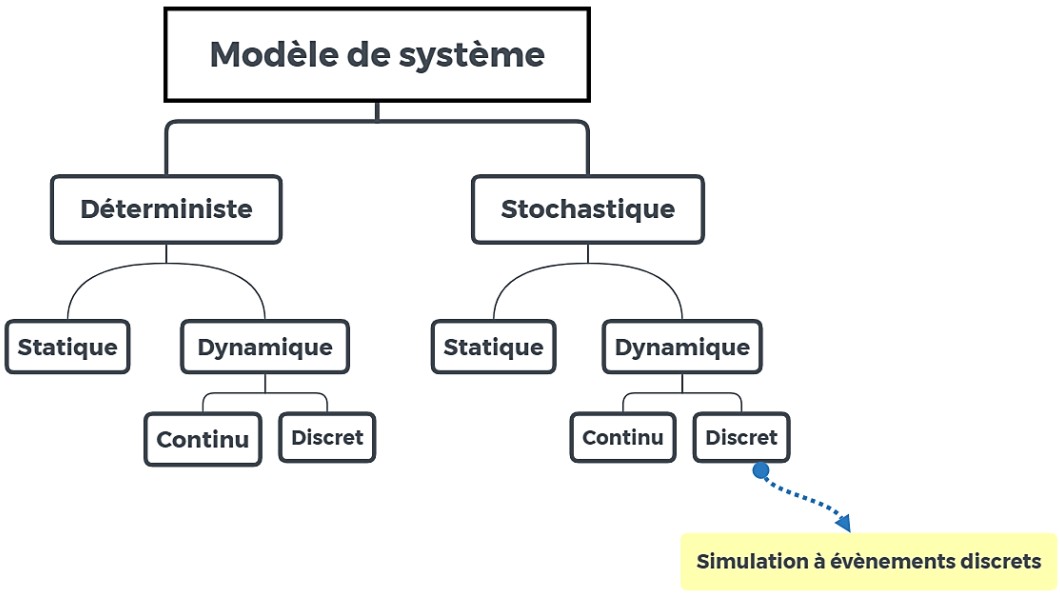


Figure 16 - Catégories de modèles de systèmes. [29, Fig. 1.1.1.]

5.1.2 Caractéristiques d'un modèle à événements discrets

Le modèle de simulation qui est développé dans notre étude est un modèle de simulation à événements discrets. Ce modèle, comme le montre la figure précédente, possède les caractéristiques suivantes [29, p. 3] :

- Stochastique
- Dynamique
- Discret

Un système est dit stochastique lorsque certaines des composantes qui déterminent son état sont aléatoires. Ces composantes sont les variables d'état du système. A l'opposé, dans un système déterministe, les composantes ne sont pas aléatoires.

Un système est dynamique lorsque l'évolution du temps a une influence importante sur les composantes qui déterminent son état. Autrement dit, l'état du système évolue au cours du temps.

Un système discret est un système dont les composantes changent d'état en fonction d'événements qui se produisent à des instants donnés au cours du temps. Dans le cas d'un système continu, les composantes changent d'état de manière continue.

5.1.3 Approches de la simulation à événements discrets

Les principales approches de la simulation à événements discrets sont les approches par événements et par processus [1, p. 183].

- **Approche par événements**

  Dans cette approche, ce sont les événements qui provoquent des changements d'état dans le système. Les différents événements qui caractérisent le système doivent être clairement identifiés et les actions à exécuter au moment de leur déclenchement doivent également être spécifiées. Ces événements sont ordonnés en fonction des instants de leurs occurrences et s'exécutent, dans un ordre de temps croissant au cours de la simulation, provoquant ainsi des changements d'état dans le système.

  Les trois types d'événements identifiés dans le modèle du système étudié et qui provoquent des changements d'état sont les suivants :

  - Les arrivées : l'arrivée d'un nouveau client augmente le nombre de jobs dans le système.

- Les départs : le départ d'un ou plusieurs clients réduit le nombre de jobs dans le système.

- La fin de la simulation : provoque l'arrêt de l'activité dans le système.

- **Approche par processus**

  Dans cette approche, un processus, qui est une suite d'états, décrit la progression complète d'une entité, telle qu'un client, qui évolue dans le système. Il contient la séquence ordonnée des événements relatifs à cette entité. L'ensemble de ces événements est réparti dans le temps en étant séparés par des intervalles. Il peut y avoir plusieurs processus concurrents et à chaque processus correspond une routine qui décrit sa progression à travers le système.

Dans le cas de l'approche par événements, l'action qui provoque le changement d'état dans le système est exécutée de manière instantanée, lorsque l'événement correspondant survient. Dans le cas de l'approche par processus, par contre, l'action évolue dans le temps et les événements s'exécutent en suivant l'ordre de leurs instants de survenance.

Dans le cadre de ce travail, c'est l'approche de simulation par événements qui est utilisée. Le choix de cette approche est principalement justifié pour des raisons de performance. En effet, l'approche par processus doit gérer la synchronisation entre les différents processus en plus de traiter les événements. Par conséquent, les temps d'exécution peuvent être plus longs. Et dans le cas des processus java, langage utilisé pour l'implémentation de ce modèle, l'approche par processus est environ douze fois plus lente que l'approche par événements [30]. C'est pour cette raison que notre choix s'est orienté vers l'approche par événements.

5.2 ANALYSE DU MODÈLE INFORMATIQUE

L'analyse du modèle informatique qui est utilisée, dans le cadre de ce travail, pour le développement du modèle de simulation à événements discrets s'appuie sur l'approche « Next-Event » [29, Chap. 5]. Cette approche, très générale, convient également pour les systèmes de files d'attente et permet de répondre facilement à l'extension des modèles de simulation.

L'approche *Next-Event* se compose des éléments suivants [29, Chap. 5]:

- **L'état du système**
  C'est ce qui caractérise le système, dans son entièreté, à un instant donné. C'est, en quelque sorte, une vue complète du système à un certain moment. Ce sont les valeurs prises par les variables d'état du système qui caractérisent son état à un moment donné. Lorsque ces variables prennent d'autres valeurs au cours du temps, l'état du système change également.

- **Les événements**
  Ce sont des occurrences qui peuvent produire des changements dans l'état du système. Ces événements peuvent être de différents types comme l'arrivée d'un client, le départ d'un client ou encore la fin de la simulation. Le changement d'état du système ne peut avoir lieu que lorsqu'un de ces événements se produit.

- **L'horloge de simulation**
  L'horloge de simulation est une variable qui représente le temps simulé. En effet, Il est important de faire une distinction entre le temps que dure l'exécution d'une simulation et la période de temps qui est simulée, il n'y a pas de correspondance entre les deux. Pour simuler un système, il est nécessaire d'utiliser un mécanisme qui permet d'avancer l'horloge de simulation entre chaque événement, sans devoir attendre. Pour cela, deux approches sont proposées. La première approche consiste, chaque fois, à avancer l'horloge jusqu'à l'instant du prochain événement tandis que la seconde approche consiste à avancer l'horloge en l'incrémentant par intervalles de temps fixes. En pratique, c'est la première approche qui est la plus répandue dans les programmes de simulation [1, p. 7]. C'est également cette approche qui est utilisée pour le développement de notre modèle informatique.

- **La planification des événements**
  Dans un modèle de simulation *Next-Event*, les événements sont planifiés à l'avance et ordonnés en fonction des instants de leurs occurrences. Chaque avancée de l'horloge de simulation permet l'exécution d'un événement dans l'ordre planifié. De cette manière, l'horloge ne risque pas de retour en arrière et l'exécution d'un événement qui précède ne peut pas se produire.

- **La liste des événements**
  La liste des événements ou l'échéancier est simplement une structure de données, un tableau ou une liste chaînée par exemple, qui contient l'ensemble des événements à exécuter tels qu'ils ont été planifiés.

5.3 ETAPES D'UNE SIMULATION NEXT-EVENT

Le développement d'un modèle de simulation à événements discrets basé sur l'approche *Next-Event*, se compose des quatre étapes suivantes [29, p. 189] :

- **Initialisation :**
  Cette première étape permet d'initialiser l'horloge de simulation, la plupart du temps à zéro, et de planifier le premier événement en l'insérant dans la liste des événements, ce qui permet d'initialiser cette dernière également.

- **Traitement de l'événement courant :**
  L'événement courant est le prochain événement qui a été récupéré dans la liste des événements. Son traitement consiste alors à avancer l'horloge de simulation jusqu'à l'instant de son occurrence, tel qu'il a été planifié, et à modifier l'état du système en fonction de l'action correspondante à cet événement.

- **Planification des nouveaux événements :**
  L'événement courant peut, lui-même, générer d'autres événements. Ces nouveaux événements doivent être, également, planifiés et insérés dans la liste des événements.

- **Fin de la simulation :**
  L'horloge de simulation avance en effectuant des sauts dans le temps entre les différents événements successifs jusqu'à ce qu'une condition d'arrêt, sous la forme d'un événement, mette fin à la simulation. Un événement particulier, qui se produit qu'une seule fois en fin de simulation, permet également de mettre fin à la simulation.

5.4 ALGORITHME DE LA SIMULATION SELON L'APPROCHE NEXT-EVENT

L'algorithme de la simulation, basé sur l'approche *Next-Event*, est représenté par le diagramme de la figure 17.

Pour effectuer une simulation, telle qu'elle est décrite par l'algorigramme (Fig. 17), il est nécessaire d'introduire les valeurs de certains paramètres d'entrée comme le taux d'arrivée, le taux de service et la période de simulation. La simulation démarre après avoir validé les valeurs de ces paramètres. L'horloge et la liste des événements sont d'abord initialisés, puis la fin de la simulation et l'arrivée des premiers événements sont planifiés. Ensuite, la simulation entre dans une boucle qui consiste à récupérer le prochain événement dans la liste des événements et à avancer l'horloge de simulation jusqu'à l'instant de cet événement, lequel devient alors l'événement courant. Ce dernier est alors traité après avoir, éventuellement, planifié de nouveaux événements. La simulation s'arrête lorsqu'il n'y a plus d'événements à traiter dans la liste des événements ou lorsque la période de simulation se termine. Finalement, les résultats des statistiques sont affichés sous forme de graphiques.

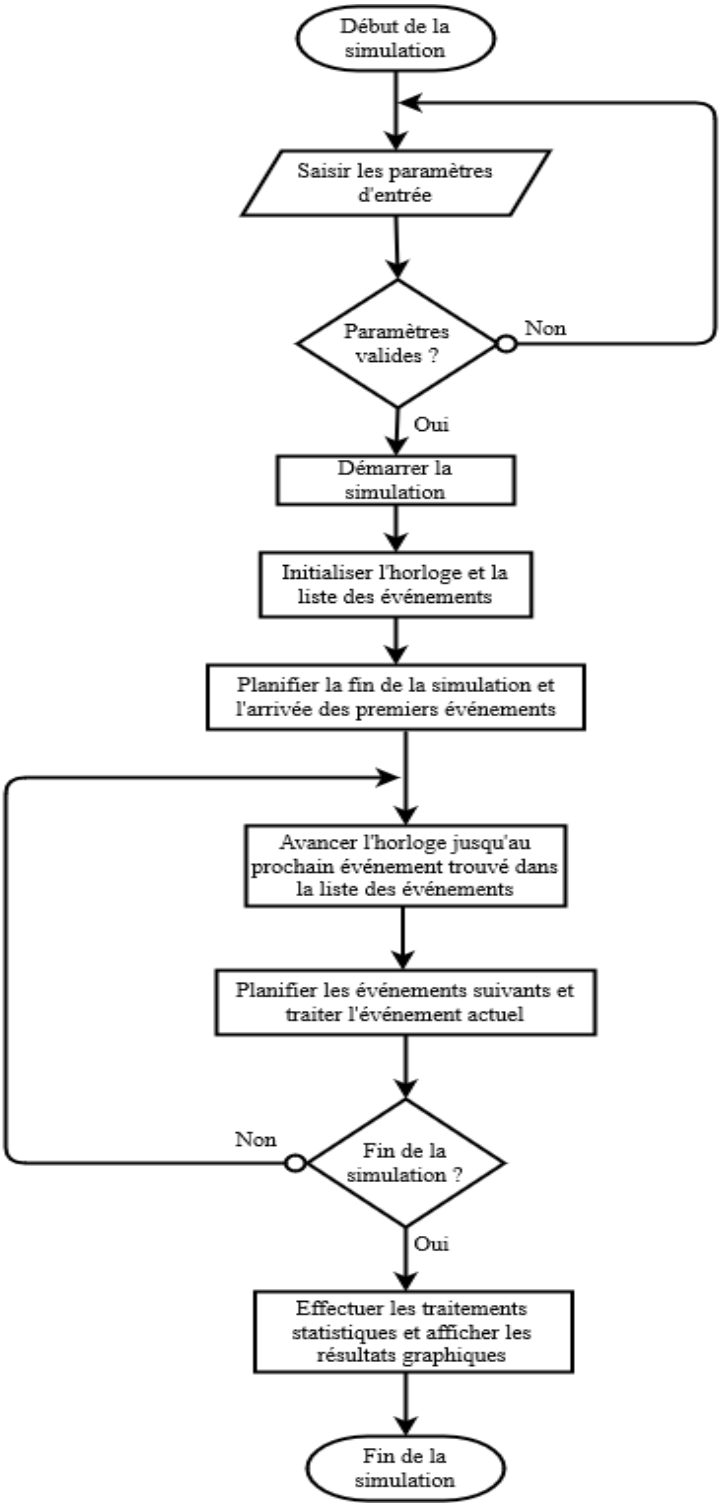


Figure 17 - Algorigramme de la simulation selon l'approche Next-Event

5.5 IMPLÉMENTATION DU MODÈLE

Cette section décrit la mise en œuvre du modèle informatique, c'est-à-dire la mise en place de l'environnement technique et le développement du code.

5.5.1 Choix techniques

Des choix techniques ont été effectués dans le but de permettre le développement du modèle informatique. Parmi ces choix :

- **Le langage de programmation Java**
  Des logiciels de simulation, avec des environnements graphiques, sont disponibles dans le commerce. Ils permettent de faire des simulations sans nécessiter de compétences en programmation. Il suffit de paramétrer un modèle puis de démarrer la simulation correspondante. Mais l'inconvénient est qu'ils sont destinés, dans la plupart des cas, à un groupe de modèles relativement restreint, avec une logique déjà établie. Par conséquent, il peut être nécessaire et parfois plus intéressant d'utiliser un langage de programmation général pour un développement plus flexible, permettant de répondre à des besoins plus spécifiques [30].
  Pour le développement du modèle informatique, notre choix s'est orienté vers le langage de programmation Java. C'est un langage puissant et très répandu, qui offre des possibilités d'extension en incluant de nouveaux outils de développement comme des libraires par exemple.

- **La librairie SSJ**
  La librairie SSJ, qui sera détaillée dans la section suivante, est une librairie qui fournit des fonctionnalités importantes pour le développement de simulations en Java.

- **La librairie FreeChart**

  La librairie FreeChart[1] offre des fonctionnalités intéressantes pour une visualisation graphique des résultats de simulation. La plupart des graphiques usuels, comme les histogrammes, les diagrammes circulaires, les nuages de points ou les courbes, sont pris en charge par cette librairie.

[1] https://www.jfree.org/jfreechart/

5.5.2 Présentation de la librairie SSJ

La librairie SSJ, littéralement *Stochastic Simulation in Java*, est une librairie composée d'un ensemble de fonctionnalités, organisées en paquets Java, destinées à faciliter le développement de simulations en Java [30]. Dans cette section, nous présentons les principales fonctionnalités utilisées pour la mise en œuvre du modèle informatique.

Gestion des événements

La gestion des événements est principalement assurée par les classes *Sim* et *Event* du paquet *simevents*. Elles contiennent toutes les fonctionnalités utiles pour le traitement d'événements discrets [31] [30].

La classe *Sim* permet de gérer l'ordonnancement des événements. Elle contient l'horloge de simulation et un lien vers la liste des événements [30].

La classe *Event*, quant à elle, est une classe abstraite qui permet de créer différents types d'événements et de planifier les instants de leurs occurrences. Elle contient une méthode abstraite *actions()* dont l'implémentation permet de spécifier le comportement de l'événement au moment de son déclenchement [30].

Collecte des statistiques

Les observations réalisées durant une simulation sont collectées dans le but d'effectuer des études statistiques. Le paquet *stat* fournit plusieurs classes permettant de collecter des observations et d'effectuer des calculs statistiques de base [30]. La principale classe utilisée dans notre développement est la classe *Tally*. Chaque instance de cette classe peut collecter des données en lui fournissant chaque nouvelle observation au moyen de sa méthode *add()*. Les résultats statistiques de base comme la moyenne, la variance ou l'intervalle de confiance, peuvent alors être calculés, par d'autres méthodes, sur les données collectées [30].

Génération de nombres aléatoires

La génération de nombres aléatoires, ou plutôt pseudo-aléatoires, est un élément important de la simulation par événements discrets. En effet, pour effectuer une simulation, il est nécessaire de déterminer, de manière aléatoire, un instant d'arrivée et une durée de service pour chaque nouveau client. Ces données peuvent être générées au moyen d'un algorithme, c'est-à-dire par un traitement logique. Le problème est qu'après une certaine période, qui dépend de l'algorithme utilisé, la même séquence de nombres aléatoires se reproduit. C'est pour cette

raison qu'on parle de nombres pseudo-aléatoires, contrairement à certains phénomènes naturels qui, eux, peuvent être complètement aléatoires. Le choix d'un algorithme avec une période suffisamment longue est donc important.

Un générateur de nombres aléatoires dispose d'un algorithme permettant de générer des nombres aléatoires uniformément répartis sur l'intervalle [0,1]. Le paquet *rng*, de la librairie SSJ, fournit différents générateurs de nombres aléatoires au moyen de l'interface *RandomStream* et de ses différentes implémentations qui se distinguent par les longueurs de leurs périodes, allant de 2^{113} à 2^{19937}, et les vitesses de génération des nombres. Chaque instance de ces classes doit fournir, comme spécifié par l'interface *RandomStream*, un flux de nombres aléatoires sous forme de plusieurs sous-flux [30]. Pour le développement de notre modèle informatique, c'est le générateur *MRG32k3a* qui a été utilisé. Il fournit plusieurs flux et sous-flux, ce qui permet d'utiliser la même instance pour les instants d'arrivée des clients et leurs durées de service. Enfin, la longueur de sa période est largement suffisante pour les besoins de notre développement. De plus, son utilisation très répandue constitue, également, une preuve de son efficacité.

Génération de variables aléatoires

Dans un modèle de file d'attente, les instants d'arrivée et les temps de service sont distribués selon des lois de probabilité. Il est donc nécessaire d'utiliser une technique permettant de produire des données aléatoires selon des distributions de probabilité spécifiques. Pour cela, une des méthodes utilisée est l'inversion de fonction de distribution [29, p. 236]. C'est la technique utilisée par défaut dans la librairie SSJ. Le paquet *randvar* fournit un ensemble de classes permettant de générer, facilement, des variables aléatoires à partir des distributions de probabilité classiques [30]. Le paquet *probdist*, quant à lui, propose un ensemble de fonctionnalités très intéressantes, dont le calcul des fonctions de distribution et de leurs inverses, pour la plupart des distributions de probabilité [30]. Un générateur de variables aléatoires s'obtient par la mise en relation de deux éléments : un flux de nombres aléatoires uniformes, qui est une instance de type *RandomStream*, et une distribution de probabilité obtenue par une instance de classe du paquet *probdist*. A partir de ces deux instances, la classe *RandomVariateGen*, du paquet *randvar*, permet de construire un générateur de variables aléatoires très facilement. Le générateur est obtenu par une instance de cette classe et il suffira simplement d'appeler sa méthode *nextDouble()* chaque fois qu'une variable aléatoire discrète, correspondante à une distribution donnée, doit être générée [30].

5.6 VERIFICATION

La vérification est une étape importante dans le développement d'un modèle de simulation. Elle consiste à s'assurer que le modèle fonctionne correctement, c'est-à-dire que son comportement est bien celui qui est attendu [32]. Il y a différentes manières d'effectuer cette vérification comme, par exemple, l'analyse de la trace d'exécution, la réalisation de tests avec des données d'entrée dont les résultats sont connus, etc.

Pour effectuer la vérification de notre modèle, un test d'hypothèse a été réalisé. L'objectif de la démarche étant de vérifier la validité de la simulation. A cet effet, une routine pour le test d'égalité de deux proportions a été développée. La description complète de ce test et les résultats obtenus sont présentés ci-dessous.

Pour ce test, deux simulations distinctes ont été réalisées, ce qui permet d'obtenir deux échantillons indépendants. Le premier échantillon, appelons-le A, correspond à une simulation sur une longue période : 10000 unités de temps. Le second échantillon, appelons-le B, correspond à la répétition d'une simulation, 1000 fois dans ce cas-ci, avec les mêmes paramètres d'entrée. Ce qui est observé sur ces deux échantillons, ce sont les instants durant lesquels le système se retrouve avec zéro client. Autrement dit, la proportion de temps durant laquelle il n'y a plus de clients dans l'échantillon A, et la proportion des simulations qui se terminent avec zéro client dans l'échantillon B.

Le processus stochastique simulé étant ergodique, il doit en effet présenter la propriété que la distribution stationnaire de l'état du système doit correspondre à la distribution à un instant quelconque (suffisamment grand par rapport au démarrage de la simulation). La distribution stationnaire est ici estimée dans la première simulation par la proportion de temps passé dans l'état 0 et la distribution à un instant quelconque est estimée par la proportion de simulations où à l'instant 1000, le système était dans l'état 0.

L'hypothèse à tester, appelée hypothèse nulle et notée H_0, est l'égalité des deux proportions, c'est-à-dire tester si les échantillons proviennent de populations ayant des proportions identiques. L'hypothèse alternative, notée H_1, est celle où les proportions ne sont pas les mêmes.

Les hypothèses sont alors formulées de la manière suivante :

$$\begin{cases} H_0 : P_A = P_B \\ H_1 : P_A \neq P_B \end{cases}$$

Où P_A et P_B représentent les proportions d'individus, dans chaque échantillon, qui prennent la valeur zéro, par rapport au nombre total d'individus.

Il faut également fixer un niveau de risque quant à la décision qui sera prise, sur base du résultat des échantillons, d'accepter ou de rejeter l'hypothèse nulle, H_0, qui est testée.

Généralement, ce risque est fixé à 5%. Il correspond au risque de rejeter l'hypothèse nulle, H_0, alors que celle-ci est vraie. Ce risque est également appelé risque de première espèce et noté α.

La valeur utilisée pour décider de rejeter ou pas l'hypothèse testée est calculée, à partir des échantillons observés, par la routine utilisant la statistique de test appropriée au test de proportions. La règle de décision sera de rejeter l'hypothèse H_0, en acceptant un faible risque d'erreur, si la valeur de la statistique de test qui est calculée n'est pas comprise dans un certain intervalle. Ce dernier est obtenu à partir des tables statistiques appropriées au test. Et pour α = 0.05, l'intervalle donné par les tables statistiques est le suivant : [-1.96, 1.96].

Les résultats du test sont calculés, par la routine de test, pour différentes valeurs de λ. Ils sont mis en forme, à partir de la trace d'exécution, dans le tableau suivant (Tab. 1) :

Table 1 - Test d'égalité de deux proportions

	Echantillon A		**Echantillon B**		**Statistique calculée**
	Taille	**Proportion**	**Taille**	**Proportion**	
λ = 0.5	10000	0.6687234527408364	1000	0.665	0.23846193209849464
λ = 5	10000	0.16631546876926398	1000	0.172	-0.45971861658204727
λ = 30	10000	0.03177152342357992	1000	0.035	-0.5525383655548164

Les résultats obtenus montrent que le test n'est pas significatif. La valeur de la statistique de test calculée est comprise dans l'intervalle de non rejet. Par conséquent, il n'est pas possible de rejeter l'hypothèse H_0 sans contrôler l'erreur commise. Les résultats laissent donc à penser que les deux proportions sont bien égales. La différence observée n'est pas le résultat de la simulation mais probablement due à l'échantillonnage.

5.7 VALEURS DES PARAMÈTRES D'ENTRÉE POUR LA SIMULATION

Les paramètres d'entrée sont la période de simulation, le taux d'arrivée λ et le taux de service μ. Chaque simulation est réalisée avec trois valeurs différentes du taux d'arrivée :

- $\lambda = 5$
- $\lambda = 15$
- $\lambda = 30$

Les valeurs des deux autres paramètres restent constantes pour toutes les simulations et sont fixées de la manière suivante :

- Période de simulation : T = 10000
- Taux de service : $\mu = 1$

Pour la période de simulation, la valeur à considérer doit être évaluée à l'avance. En effet, il est important que cette valeur soit suffisante pour que le système puisse atteindre un régime stationnaire correspondant à un état de fonctionnement normal. Si cette période de simulation est trop courte, le système risque d'être encore dans un régime transitoire correspondant à une phase de démarrage. Dans ce dernier cas, les résultats observés ne refléteraient pas le fonctionnement normal du système mais plutôt sa phase de démarrage.

La valeur de cette période peut être évaluée à partir des résultats observés au moyen de tests préliminaires. Le tableau suivant (Tab. 2) résume les résultats observés durant cette étape :

Table 2 - Choix d'une valeur convenable pour la période de simulation

Mesures pour λ = 5 / **Périodes de Simulation**	Nombre d'observations	Temps moyen de séjour [unités de temps]	Variance	Ecart type
T = 100	518	0.459	0.093	0.305
T = 1000	4981	0.430	0.092	0.304
T = 10000	**50048**	**0.438**	**0.099**	**0.314**
T = 100000	501170	0.438	0.098	0.313
T = 1000000	5002005	0.438	0.099	0.314

Le tableau (Tab. 2) montre qu'à partir d'une certaine période de simulation, les fluctuations des résultats obtenus restent très faibles même si le nombre d'observations continue d'augmenter. Le fait de simuler encore plus longtemps n'influencera plus les résultats de manière sensible. Par conséquent, le choix de la période de simulation est fixé à T = 10000.

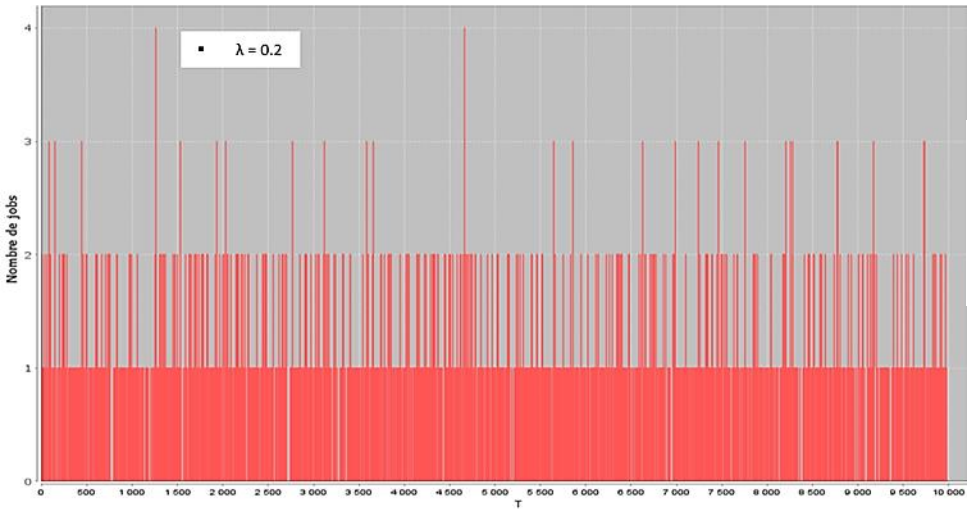
5.8 ANALYSE DES RÉSULTATS

Cette section présente une analyse des résultats obtenus par les différentes simulations. Les performances évaluées par les différentes simulations sont les suivantes :

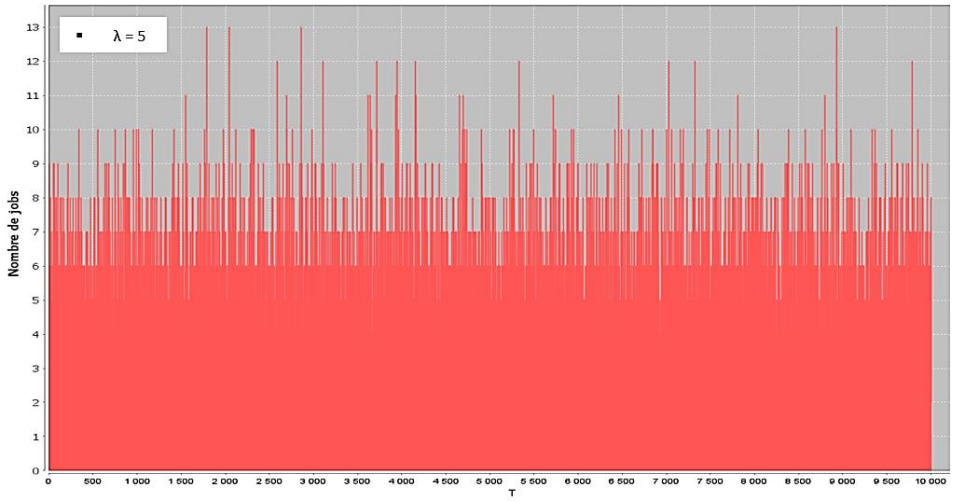
- Evolution du nombre de jobs dans le système
- Le délai observé pour un client dans le système
- Fractions de temps occupées par le système dans un état avec n jobs
- Fréquences par nombre de jobs résiduels sur une répétition de 1000 simulations
- Nombre moyen stationnaire de serveurs occupés dans le système
- La période occupée (busy period)
- Comparaison des différents graphiques

5.8.1 Evolution du nombre de jobs dans le système

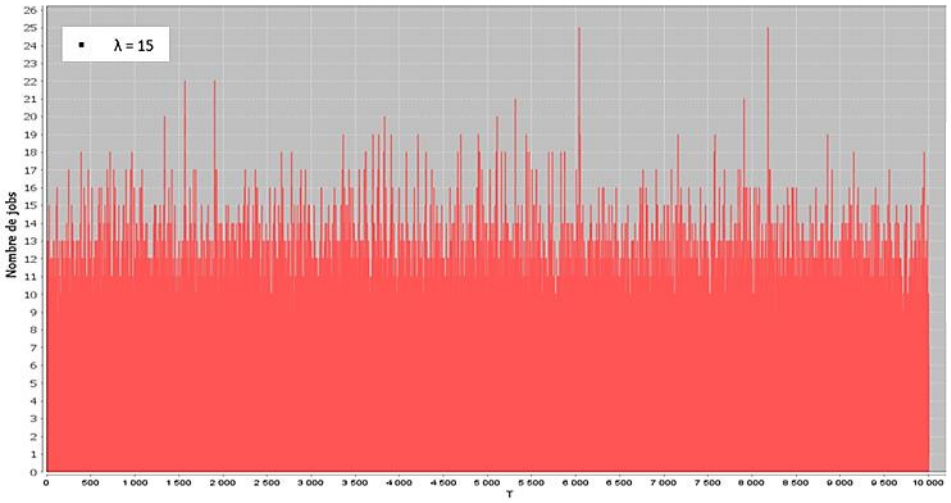
Les premiers graphiques (Fig. 18) montrent l'évolution du nombre de jobs dans le système pour différentes valeurs du taux d'arrivée (λ).



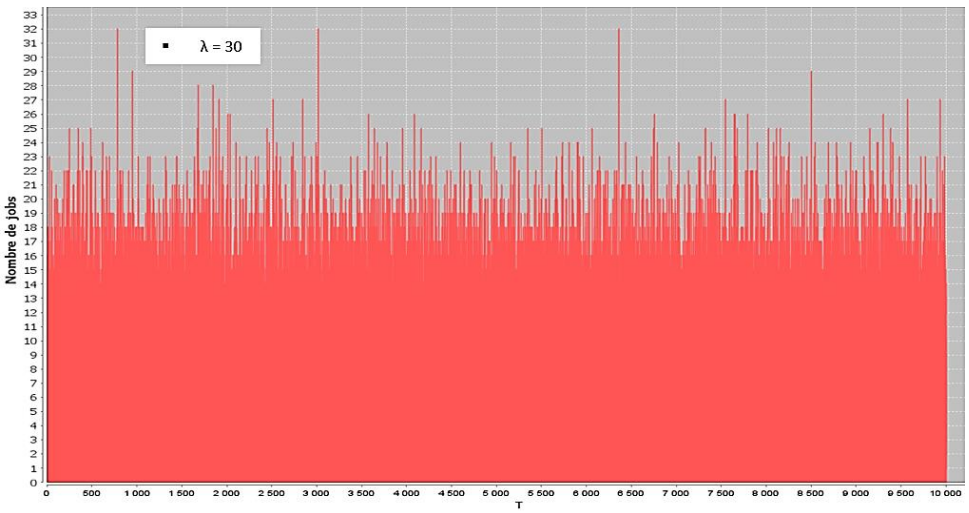
(a) . Evolution du nombre de jobs dans le système pour un taux d'arrivée λ = 0.2



(b) . Evolution du nombre de jobs dans le système pour un taux d'arrivée λ = 5



(c) . Evolution du nombre de jobs dans le système pour un taux d'arrivée λ = 15



(d) . Evolution du nombre de jobs dans le système pour un taux d'arrivée λ = 30

Figure 18 - Evolution du nombre de jobs dans le système pour différentes valeurs du taux d'arrivée λ

Ces graphiques (Fig. 18) permettent d'avoir une vue sur l'activité du système et de sa réponse en fonction du taux d'arrivées. Ils montrent que pour $\lambda<\mu$ les observations ne sont pas très intéressantes car le système se trouve trop souvent dans un état avec zéro client. Lorsque $\lambda>\mu$, le nombre de clients dans le système avant une synchronisation est très variable et augmente avec la valeur de λ.

5.8.2 Délai observé pour un client dans le système

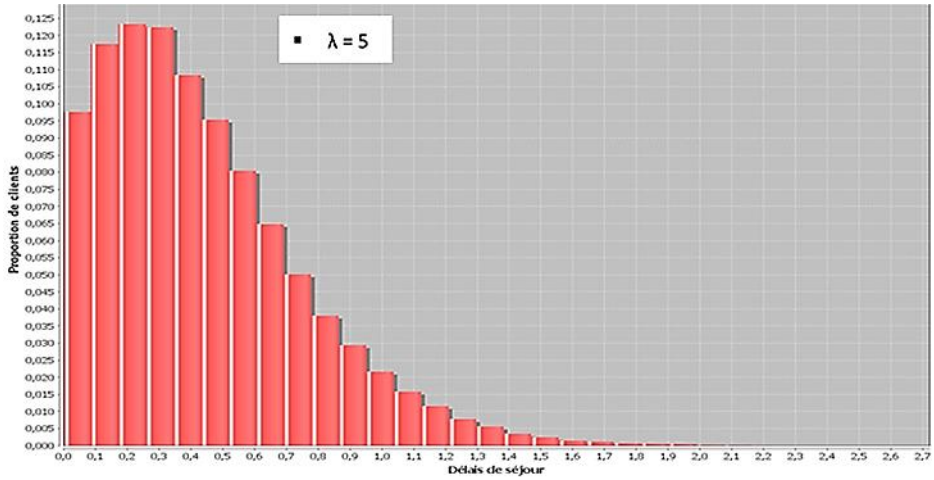
Les graphiques précédents (Fig. 18) fournissent, également, des informations intéressantes concernant le temps de séjour d'un client dans le système. Les résultats sont résumés dans le tableau suivant (Tab. 3) pour chaque graphique et valeur de λ.

Table 3 - Temps de séjour moyen d'un client dans le système en fonction du taux d'arrivée λ

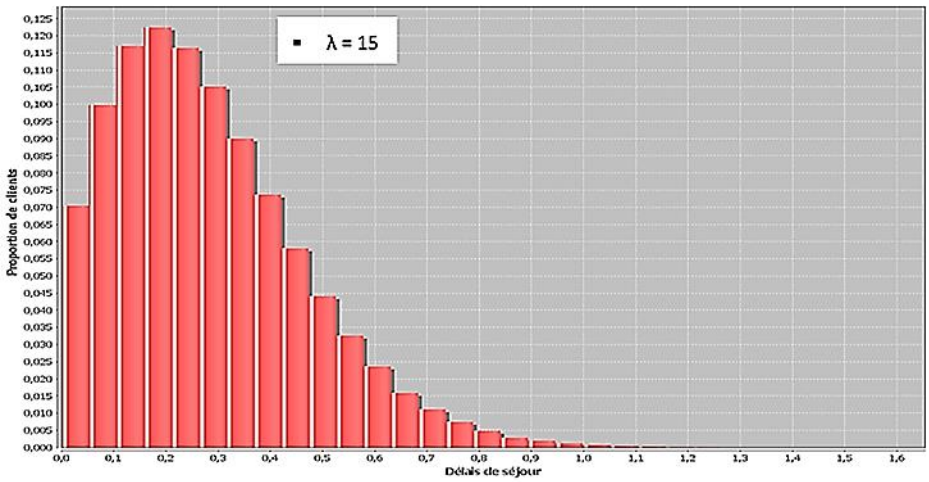
Mesures / **Valeurs de λ**	Nombre d'observations	Temps minimum	Temps maximum	Temps moyen	Variance	Ecart type
Graphe(b) : $\lambda = 5$	50048	2.5×10^{-5}	2.593	**0.438**	0.099	0.314
Graphe(c) : $\lambda = 15$	150141	2.5×10^{-5}	1.576	**0.280**	0.033	0.182
Graphe(d) : $\lambda = 30$	300545	1.3×10^{-5}	0.964	**0.208**	0.016	0.127

Les résultats montrent que plus le taux d'arrivée augmente, plus le temps de séjour d'un client dans le système diminue.

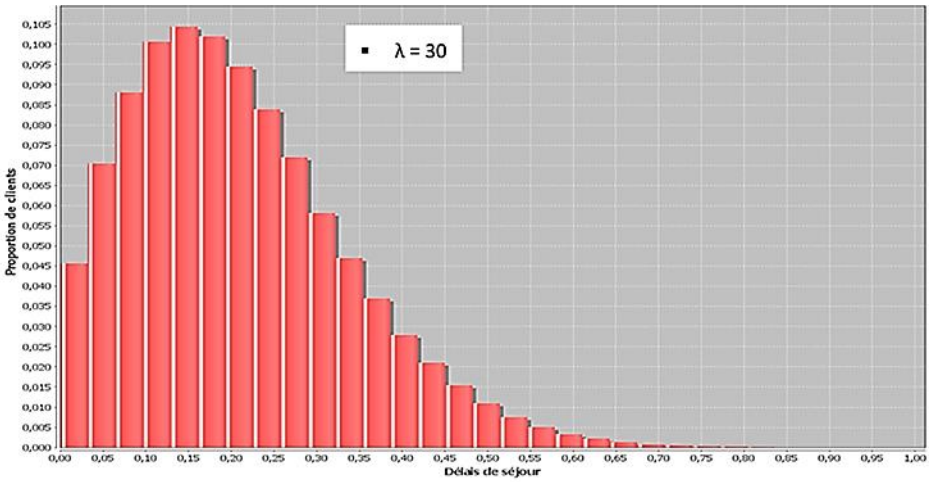
Les graphiques de la figure 19 fournissent également une information relative au délais de séjour des clients dans le système, présentée sous forme de proportions (Fig. 19).



(a) . Délai observé pour un client dans le système pour $\lambda = 5$



(b) . Délai observé pour un client dans le système pour $\lambda = 15$



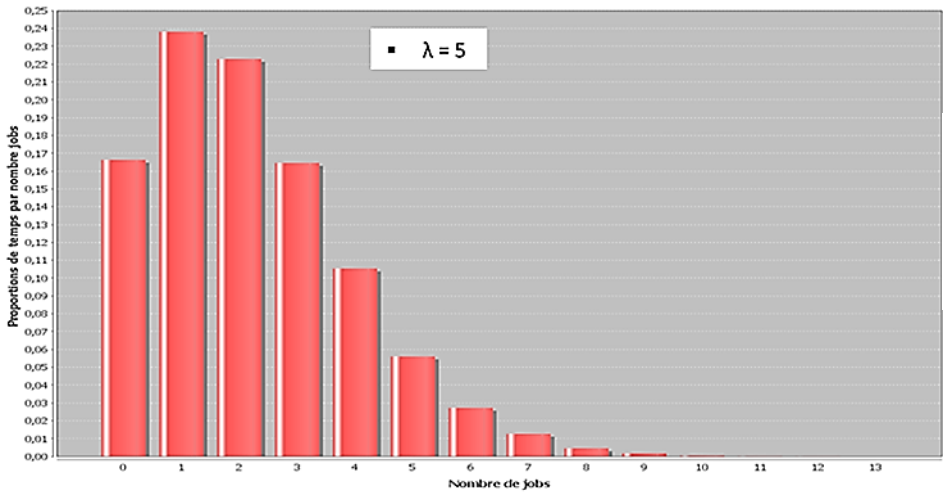
(c) . Délai observé pour un client dans le système pour $\lambda = 30$

Figure 19 - Délais de séjour des clients dans le système

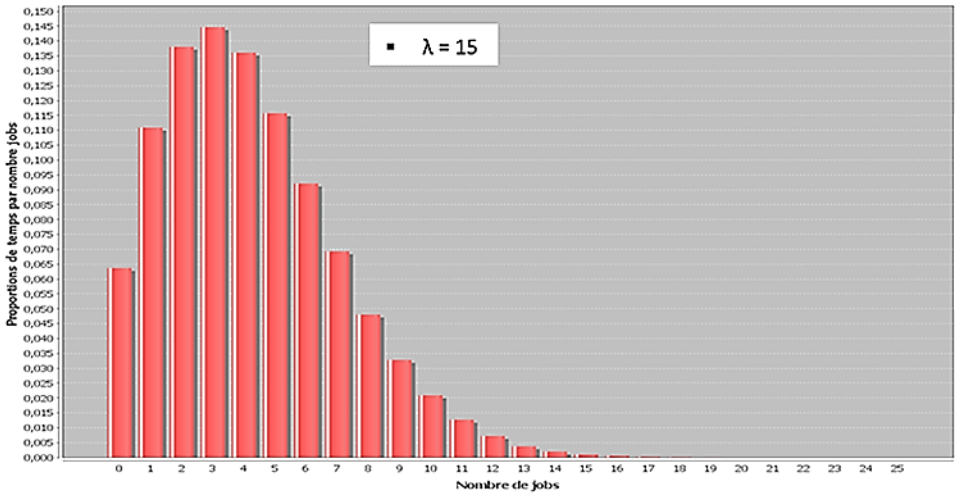
Ces graphiques montrent également que le délai de séjour des clients dans le système diminue très sensiblement lorsque le taux d'arrivée λ augmente.

5.8.3 Fractions de temps occupées par le système dans un état avec n jobs

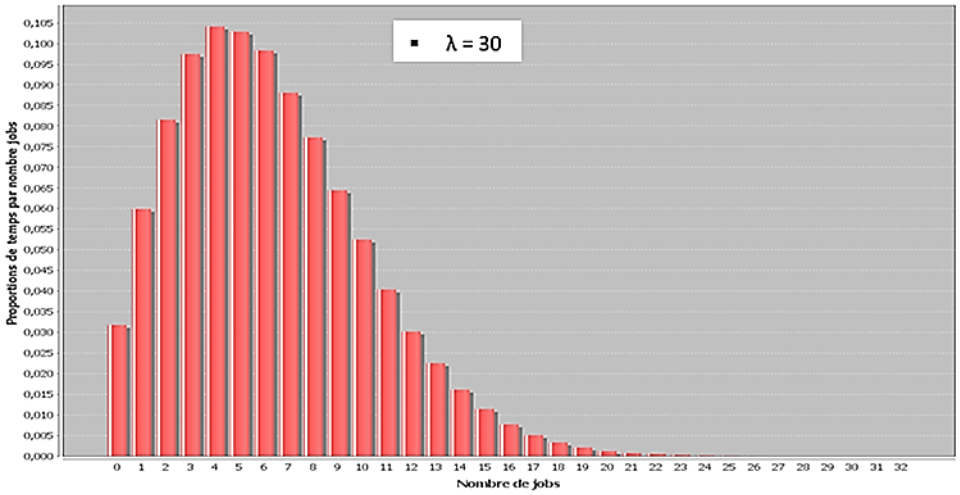
Une autre observation intéressante est la fraction de temps occupée par le système dans un état avec n jobs. Elle est représentée par les graphiques suivants (Fig. 20).



(a) . Fractions de temps occupées par le système dans un état avec n jobs pour λ = 5



(b) . Fractions de temps occupées par le système dans un état avec n jobs pour λ = 15

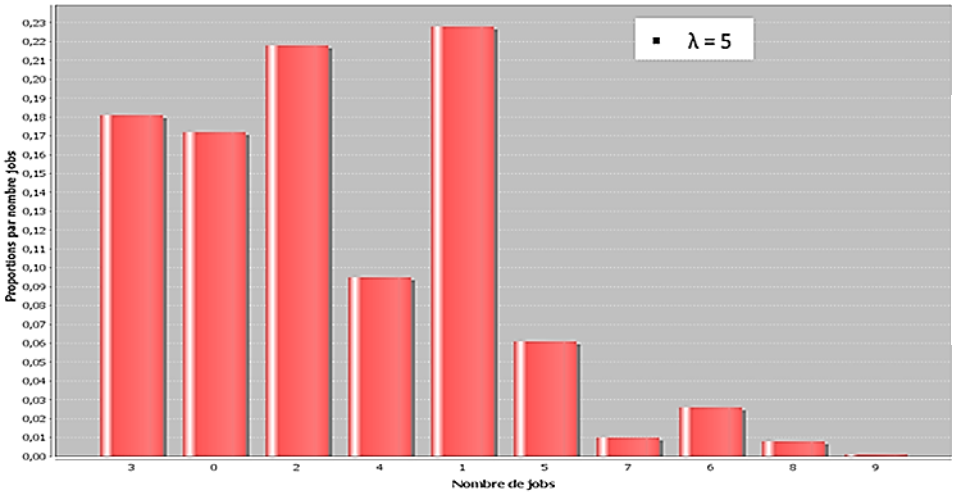


(c) . Fractions de temps occupées par le système dans un état avec n jobs pour λ = 30

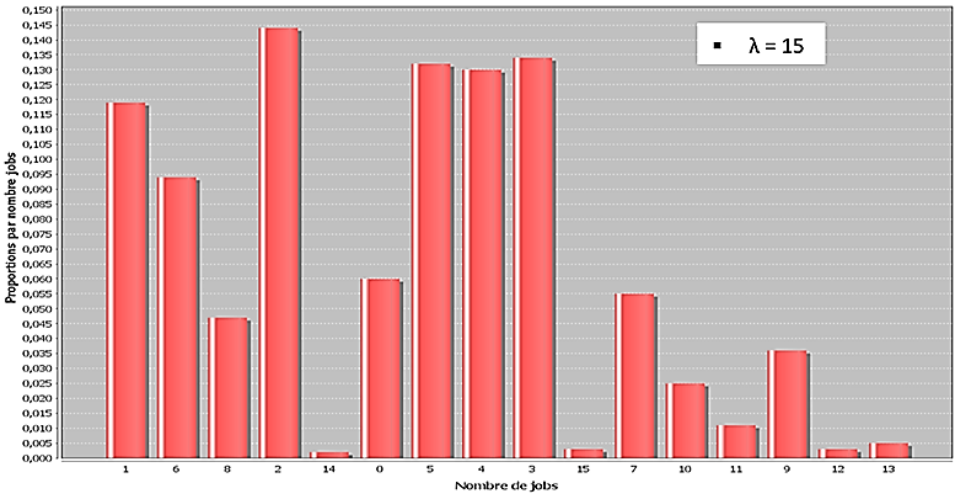
Figure 20 - Fractions de temps occupées par le système dans un état avec n jobs

5.8.4 Fréquences par nombre de jobs résiduels sur une répétition de 1000 simulations

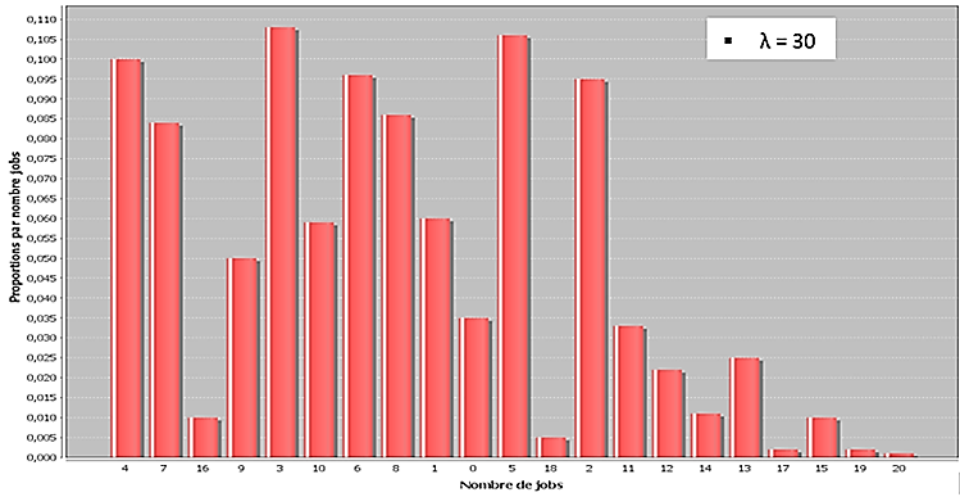
Les jobs résiduels représentent le nombre de jobs restants dans le système à la fin d'une simulation. Les fréquences de ces jobs sont calculées sur 1000 simulations (Fig. 21).



(a) . Fréquences par nombre de jobs restants dans le système sur une répétition de 1000 simulations pour λ = 5



(b) . Fréquences par nombre de jobs restants dans le système sur une répétition de 1000 simulations pour λ = 15



(c) . Fréquences par nombre de jobs restants dans le système sur une répétition de 1000 simulations pour λ = 30

Figure 21 - Fréquences par nombre de jobs restants dans le système sur une répétition de 1000 simulations

5.8.5 Nombre moyen stationnaire de serveurs occupés dans le système

A partir des graphiques précédents (Fig. 20 et Fig. 21), nous pouvons calculer le nombre moyen stationnaire de serveurs occupés dans le système. Les résultats sont repris dans le tableau suivant (Tab. 4).

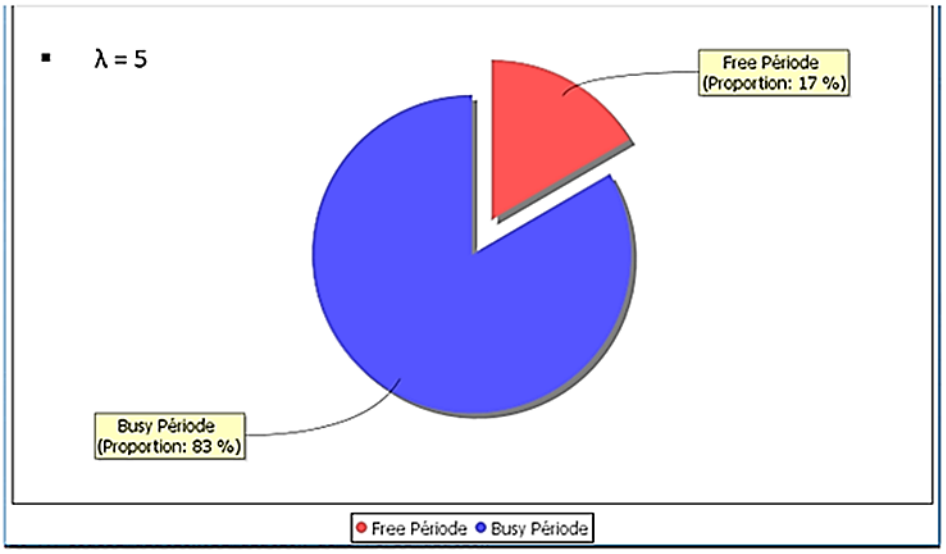
Table 4 - Nombre moyen stationnaire de serveurs occupés dans le système

	Série 1 : une seule simulation sur une période T = 10000	Série 2 : répétition sur 1000 simulations avec T = 10000
$\lambda = 5$	2.190	2.191
$\lambda = 15$	4.207	4.183
$\lambda = 30$	6.245	5.994

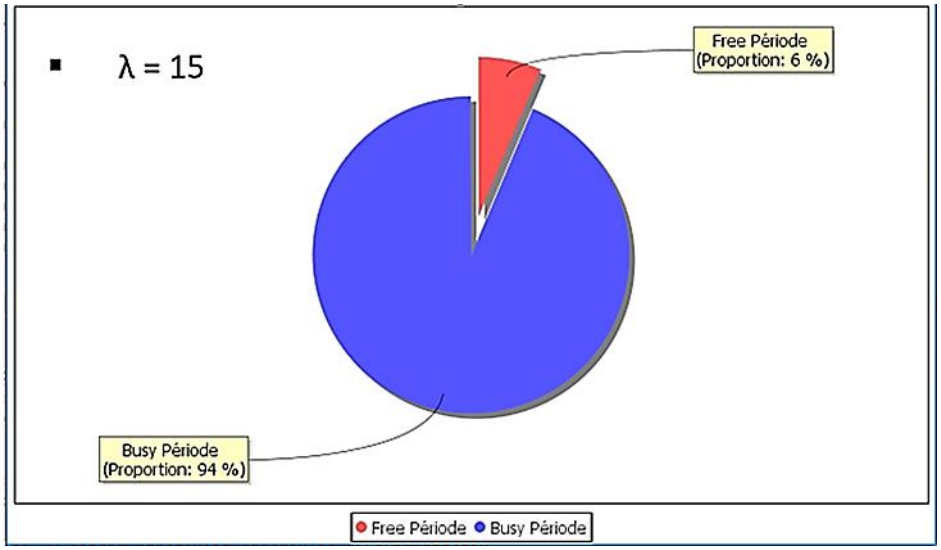
Les résultats obtenus à partir de la première série de graphiques (Fig. 20) sont très proches de ceux obtenus à partir de la deuxième série (Fig. 21). Cela montre encore une fois que la simulation du modèle est correcte.

5.8.6 La période occupée (busy period)

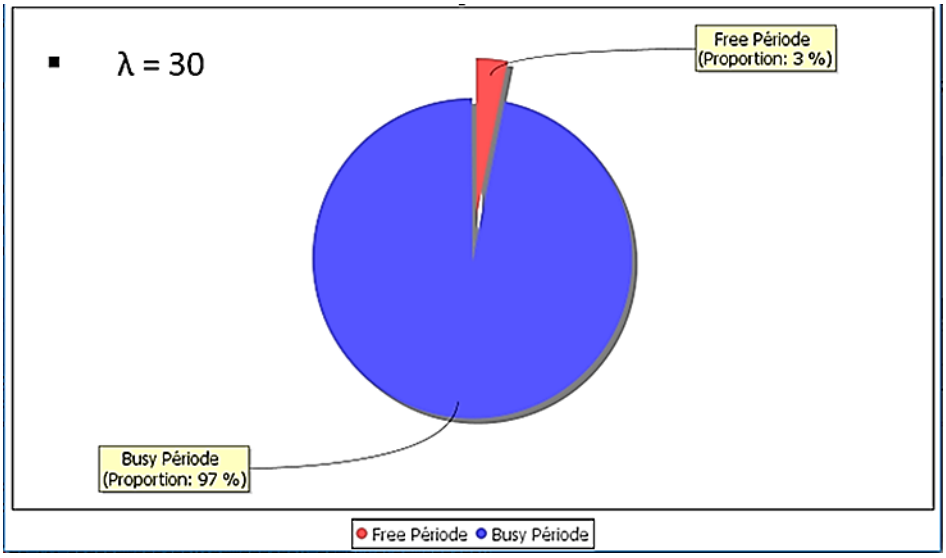
La période occupée correspond à une période de temps qui sépare deux instants où le système se retrouve dans un état avec zéro client.



(a) . La période occupée pour λ = 5



(b) . La période occupée pour λ = 15



(c) . La période occupée pour λ = 30

Figure 22 - La période occupée

Les graphiques (Fig. 22) montrent que plus le taux d'arrivée augmente, plus la période occupée augmente. Essayons de vérifier si les paramètres d'entrée, λ et μ, qui influencent la période occupée, peuvent être liés par la relation suivante :

$$\frac{\boldsymbol{\mu}}{\boldsymbol{\lambda + \mu}}$$

- Vérification de la relation pour **λ = 5** :

$$\frac{\mu}{\lambda + \mu} = \frac{1}{5+1} = \frac{1}{6} = 0.1666667 \quad ou \quad 16{,}67\ \%$$

Résultat obtenu par simulation : 17 %

Résultats très proches

- Vérification de la relation pour **λ = 15** :

$$\frac{\mu}{\lambda + \mu} = \frac{1}{15+1} = \frac{1}{16} = 0.0625 \quad ou \quad 6{,}25\ \%$$

Résultat obtenu par simulation : 6 %

Résultats très proches

- Vérification de la relation pour **λ = 30** :

$$\frac{\mu}{\lambda + \mu} = \frac{1}{30+1} = \frac{1}{31} = 0.0322581 \quad ou \quad 3{,}22\ \%$$

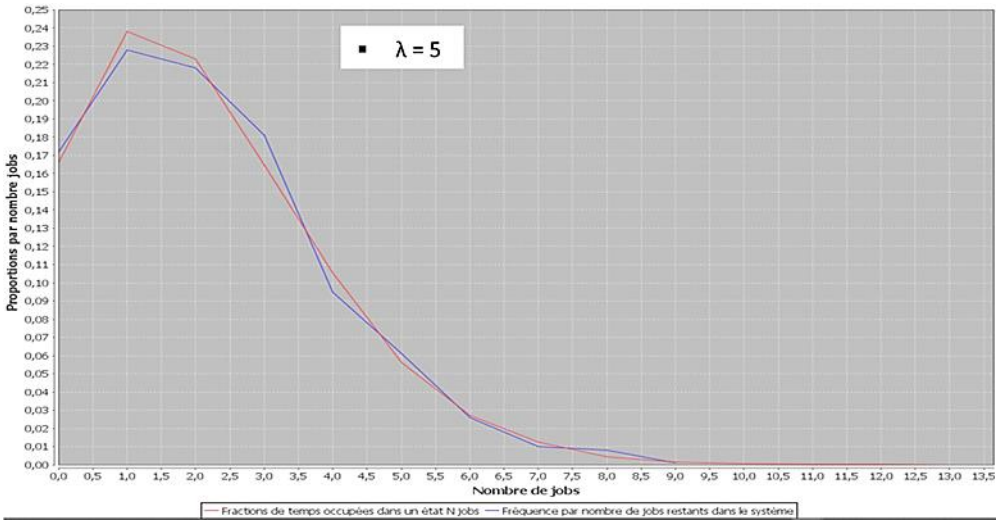
Résultat obtenu par simulation : 3 %

Résultats très proches

Les vérifications montrent que les résultats obtenus par la simulation et ceux obtenus par la relation théorique sont presque identiques. Ces résultats constituent un autre moyen de vérifier la précision de la simulation.

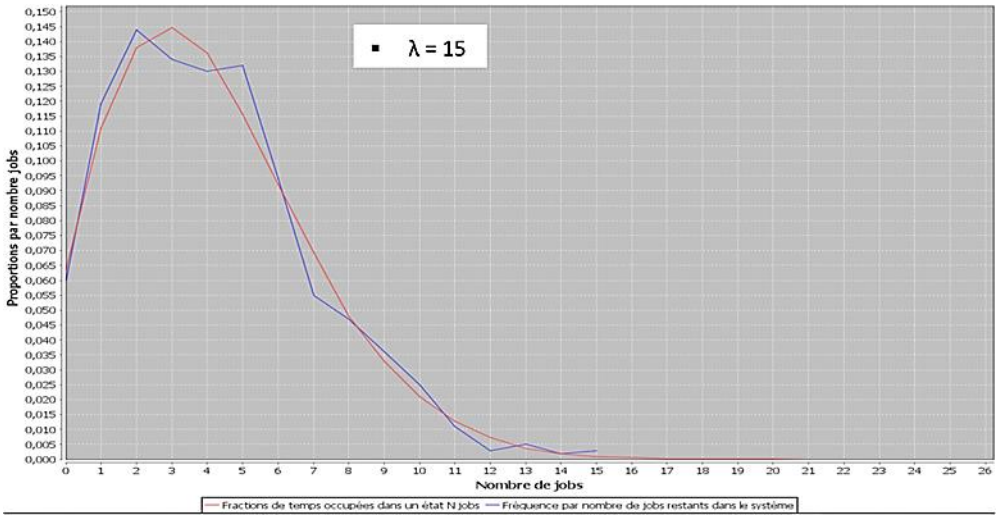
5.8.7 Comparaison des différents graphiques

Les graphiques suivants (Fig. 23) montrent la comparaison entre les graphiques des fractions de temps occupées par le système dans un état avec n jobs et ceux des fréquences par nombre de jobs résiduels sur une répétition de 1000 simulations.



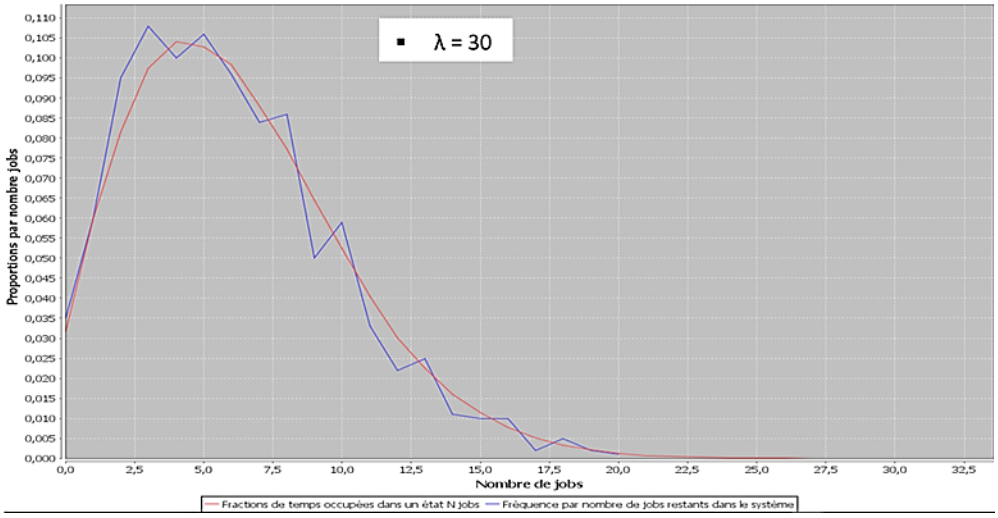
(a) . Comparaison entre graphiques des fractions de temps occupées par le système dans un état avec n jobs et ceux des fréquences par nombre de jobs résiduels sur une répétition de 1000 simulations

pour $\lambda = 5$



(b) . Comparaison entre graphiques des fractions de temps occupées par le système dans un état avec n jobs et ceux des fréquences par nombre de jobs résiduels sur une répétition de 1000 simulations

pour $\lambda = 15$



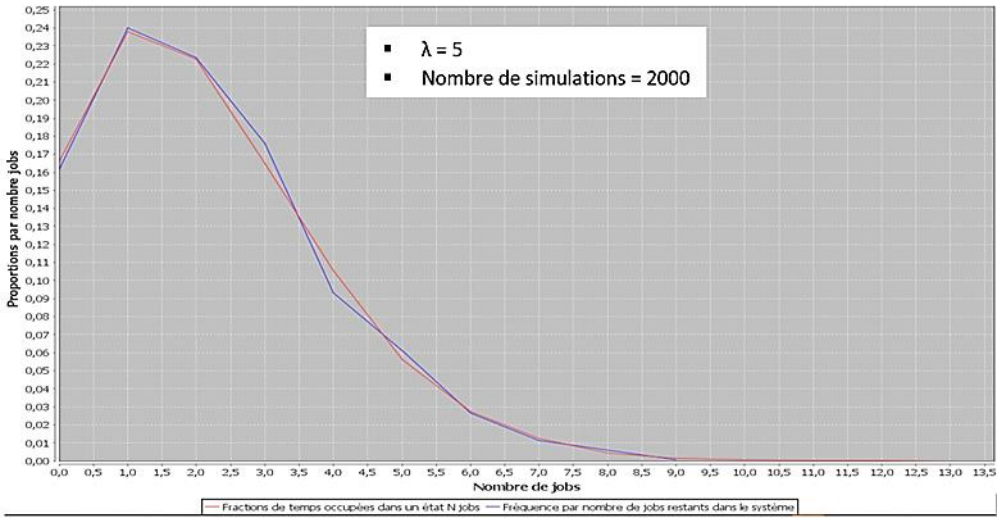
(c) . Comparaison entre graphiques des fractions de temps occupées par le système dans un état avec n jobs et ceux des fréquences par nombre de jobs résiduels sur une répétition de 1000 simulations

pour λ = 30

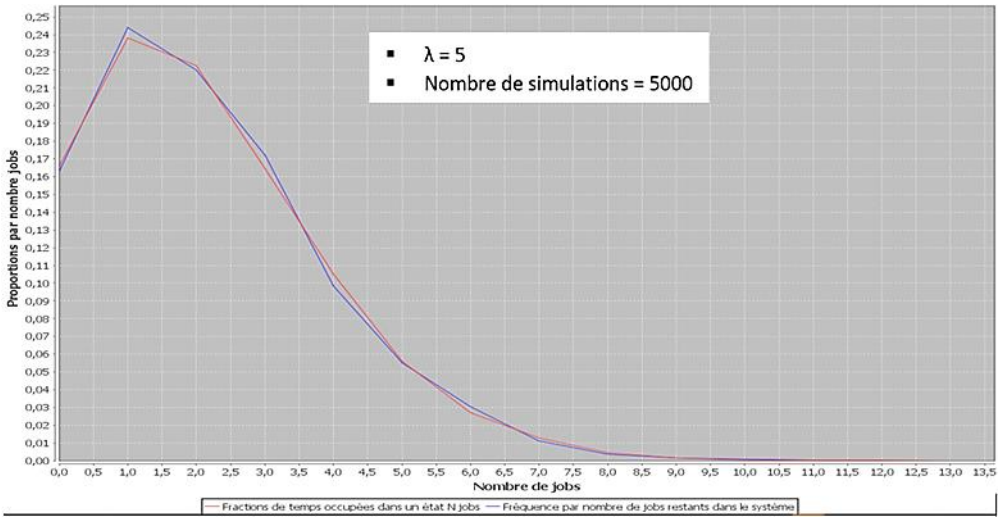
Figure 23 - Comparaison des graphiques précédents

Ces graphiques (Fig. 23) montrent que les courbes relatives aux fractions de temps occupées par le système dans un état avec n jobs et celles des fréquences par nombre de jobs résiduels sur une répétition de 1000 simulations ont presque la même allure.

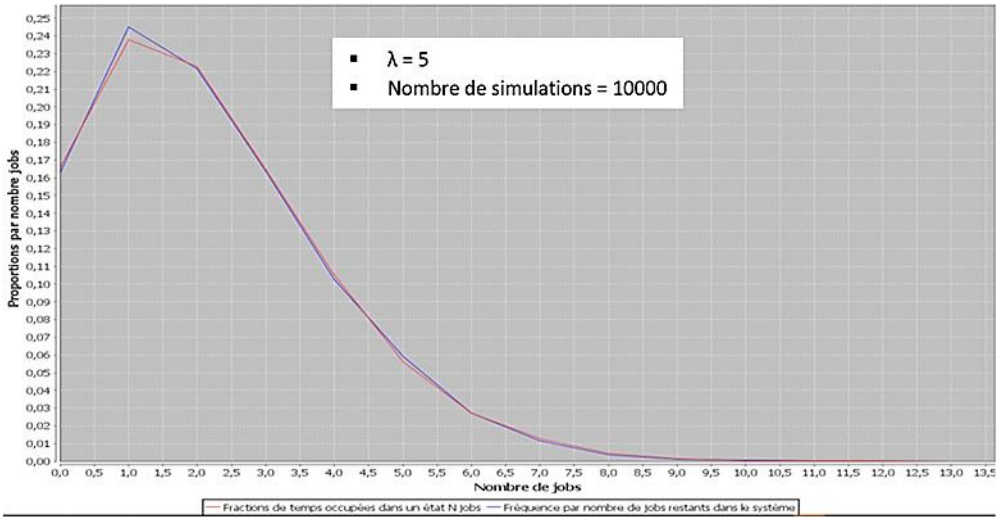
Les graphiques de la figure suivante (Fig. 24) permettent d'observer le rapprochement des courbes lorsque le nombre de simulations augmente. Ils montrent le rapprochement qui s'opère entre les graphiques des fractions de temps occupées par le système dans un état avec n jobs et ceux des fréquences par nombre de jobs résiduels sur des répétitions de simulations de plus en plus importantes (seulement pour λ=5).



(a). Rapprochement entre les graphiques des fractions de temps occupées par le système dans un état avec n jobs et ceux des fréquences par nombre de jobs résiduels lorsque le nombre de simulations augmente à 2000



(b). Rapprochement entre les graphiques des fractions de temps occupées par le système dans un état avec n jobs et ceux des fréquences par nombre de jobs résiduels lorsque le nombre de simulations augmente à 5000



(c). Rapprochement entre les graphiques des fractions de temps occupées par le système dans un état avec n jobs et ceux des fréquences par nombre de jobs résiduels lorsque le nombre de simulations augmente à 10000

Figure 24 - Rapprochement des courbes lorsque le nombre de simulations augmente

La figure montre (Fig. 24) que plus le nombre de simulations augmente, plus les courbes se rapprochent et finissent par se confondre.

DISCUSSION

L'objectif de ce travail est de simuler un modèle mathématique, proposé dans la littérature scientifique, afin de déterminer son niveau de précision par rapport à l'étude du comportement et l'évaluation des performances du mécanisme de synchronisation de la blockchain sur laquelle s'appuie le système Bitcoin.

La revue de la littérature a permis de mettre en évidence que la plupart des études s'intéressent plutôt aux différentes applications s'appuyant sur la blockchain qu'aux fondements de ses différentes implémentations. Quelques modèles sont proposés, dans la littérature, pour étudier le processus de confirmation des transactions, mais les modèles qui s'intéressent aux mécanismes de synchronisation de la blockchain sont, quant à eux, plus rares. Les recherches effectuées n'ont pas permis de trouver d'autres modèles, utilisant une approche par files d'attente, traitant de la problématique de synchronisation de la blockchain utilisée dans le système Bitcoin. Par conséquent, il n'a pas été possible de confronter le modèle étudié avec d'autres modèles similaires. Les éléments de réflexion sont principalement basés sur les résultats de la simulation.

L'analyse des résultats montre que le modèle étudié traduit bien le comportement du mécanisme de synchronisation de la blockchain, mais les simplifications effectuées semblent très importantes et ne permettent pas d'atteindre un bon niveau de précision. En d'autres termes, le modèle de base permet de bien décrire le fonctionnement du mécanisme, mais le niveau de détail ne permet pas d'évaluer, spécifiquement, les performances liées à la synchronisation de la blockchain sur laquelle s'appuie le système Bitcoin.

Les graphiques relatifs à l'évolution du nombre de jobs dans le système montrent une activité correcte du système. Les passages successifs, du graphique, par zéro indiquent des départs par lots correspondant à une synchronisation de la blockchain. Ils montrent, également, que plus le taux d'arrivée augmente, plus la taille des lots de départ augmente, ce qui correspond au fonctionnement réel. Si nous mettons ce graphique en lien avec le temps moyen de séjour d'un job dans le système, on constate que plus le taux d'arrivée augmente, plus le temps de séjour d'un job dans le système diminue. De plus, la diminution de ce temps de séjour est clairement visible sur le graphique relatif aux délais de séjour des clients. En d'autres termes, plus le nombre d'arrivées par unité de temps augmente, plus le temps nécessaire à la synchronisation de la blockchain diminue. Ceci est confirmé dans la littérature, concernant l'insertion des blocs dans la blockchain, qui montre que plus le nombre de blocs augmente, plus le temps d'attente pour l'insertion d'un bloc dans la blockchain diminue.

Les graphiques relatifs aux fractions de temps occupées par le système dans un état avec n jobs montrent également que la taille des lots de départ augmente lorsque le taux d'arrivée augmente. Le pic maximum glisse vers la droite au fur et à mesure que ce taux augmente. Cela démontre une cohérence dans les résultats obtenus et reflète, également, la représentation correcte du mécanisme de synchronisation par le modèle étudié.

La période occupée, qui est le reflet de l'activité du système, tend à augmenter avec le taux d'arrivée. Si nous mettons ce graphique en lien avec les fractions de temps occupées par le système dans un état avec n jobs, nous pouvons constater que la fraction de temps occupée par le système dans un état avec zéro job diminue lorsque le taux d'arrivée augmente. Cela signifie, en tenant compte de ce qui a été dit précédemment, que le système passe encore plus souvent par l'état zéro job mais qu'il reste moins longtemps dans cet état.

Le rapprochement des courbes, lorsque le nombre de simulations augmente, montre que le fait d'effectuer plusieurs observations sur une seule simulation est équivalent à effectuer une seule observation sur plusieurs simulations dans les mêmes conditions. Les résultats se rapprochent lorsque le nombre de simulations observées se rapproche du nombre d'observations effectuées dans une seule simulation.

Le modèle étudié a été simulé en considérant que les clients arrivent dans le système selon un processus de Poisson et que les temps de service suivent une distribution exponentielle. Cela a permis d'obtenir des résultats intéressants permettant de comprendre et d'évaluer le modèle. Cependant, la précision des résultats ne peut pas être prise en considération, et ce pour diverses raisons. La première raison est que le modèle est simulé avec des données d'entrée obtenues par des générateurs de variables aléatoires qui ne correspondent pas aux données réelles du système. L'autre raison est que la description du modèle semble relativement simple pour permettre d'étudier correctement les performances du système réel qu'il modélise.

Une proposition d'amélioration possible du modèle serait d'envisager une autre discipline de service, dans laquelle l'ordre de départ des clients est indépendant de leur ordre d'arrivée. En effet, dans la description du modèle actuel, l'ordre d'arrivée des clients dans le système conditionne leur ordre de départ, ce qui permet de simplifier considérablement le modèle, mais s'éloigne un peu de la réalité du système. Il est possible d'envisager une autre discipline de service où le premier client qui arrive dans le système sera le premier à être servi, mais ne sera pas nécessairement le premier à quitter le système. En d'autres termes, les départs et les arrivées de clients deviennent complètement indépendants.

CONCLUSION

Le travail réalisé, dans le cadre de ce mémoire, consistait à simuler un modèle mathématique, proposé dans la littérature scientifique, pour lequel une solution analytique demeure trop complexe. L'objectif était d'obtenir des résultats numériques, qui n'étaient pas envisageables avec une solution analytique, afin d'évaluer le niveau de précision du modèle par rapport au système qu'il représente et dont il propose d'étudier le comportement.

L'intérêt du modèle étudié est d'analyser le mécanisme de synchronisation de la blockchain, qui est un aspect fondamental du système Bitcoin. Le modèle permet d'étudier le comportement de cette partie du système et d'évaluer ses performances. Bien que le modèle soit inspiré du système Bitcoin, il peut, également, être utilisé pour étudier d'autres systèmes dont le comportement est similaire.

Pour permettre d'évaluer le niveau de précision du modèle proposé, il était important, dans un premier temps, de comprendre le système réel, qu'il représente, dans sa globalité. Une recherche a donc été menée pour tenter de faire une synthèse, aussi complète que possible, du système Bitcoin. Ensuite, la problématique, qui est à la base de la construction du modèle étudié, a été identifiée de manière plus précise en localisant la partie du système concernée. Cette étape importante a permis de mieux comprendre le modèle et de cerner la problématique à laquelle il s'intéresse.

Une autre étape importante, dans la revue de la littérature, était de faire une synthèse complète des différentes études relatives à la modélisation du système Bitcoin, qui utilisent une approche par files d'attente. L'information, importante, que nous avons retenu de cette étape, est que la plupart des études s'intéressent aux applications de la blockchain, dans des domaines très variés, et que très peu d'entre elles s'intéressent vraiment aux mécanismes de sa mise en œuvre dans les différents systèmes qui l'utilisent. Par conséquent, les modèles utilisant une approche par files d'attente, pour étudier les mécanismes sur lesquels s'appuie le système Bitcoin, semblent peu nombreux.

La dernière étape, dans la réalisation de ce travail, correspond au développement d'un modèle de simulation à événements discrets. C'est dans cette étape que notre contribution a été la plus importante. Au moyen de la simulation, nous avons obtenu des résultats numériques, pour les différentes performances du système, permettant, ainsi, d'évaluer le niveau de précision du modèle étudié. La méthodologie utilisée a été expliquée de manière très détaillée et les choix techniques ont été clairement justifiés. L'algorithme utilisé a été présenté et son fonctionnement a été décrit de manière très complète. Le modèle de simulation, ainsi développé, a été vérifié par un test d'hypothèse, plus particulièrement par un test d'égalité de deux proportions.

Des résultats intéressants ont été obtenus par la simulation du modèle pour l'évaluation des performances suivantes :

- Evolution du nombre de jobs dans le système
- Le délai observé pour un client dans le système

- Fractions de temps occupées par le système dans un état avec n jobs
- Fréquences par nombre de jobs résiduels sur une répétition de 1000 simulations
- Nombre moyen stationnaire de serveurs occupés dans le système
- La période occupée (busy period)
- Comparaison des différents graphiques

Les résultats, analysés et interprétés dans la partie discussion, montrent une certaine cohérence et restent en accord avec les informations recueillies dans la revue de la littérature. Par conséquent, le modèle traduit correctement le comportement du système et convient donc pour l'évaluation des performances de synchronisation de la blockchain du système Bitcoin. Il est aussi un bon modèle de base pour les systèmes qui présentent un comportement similaire. Cependant, pour tendre vers des résultats plus précis, il faudrait spécifier le modèle de départ en utilisant des données d'entrée issues du système réel et augmenter le niveau de détail du modèle.

Enfin, l'utilisation de données statistiques relatives au système réel, comme valeurs d'entrée pour les paramètres du modèle, permettrait probablement d'obtenir des résultats plus précis. De plus, ce modèle de base, très intéressant, pourrait encore être spécifié par rapport à d'autres systèmes ayant un comportement similaire ou être comparé à d'autres approches de modélisation.

BIBLIOGRAPHIE

[1] A. M. Law, *Simulation modeling and analysis*, Fifth edition. Dubuque: McGraw-Hill Education, 2013.

[2] B. Fralix, « On classes of Bitcoin-inspired infinite-server queueing systems », *Queueing Syst*, vol. 95, nº 1-2, p. 29-52, juin 2020, doi: 10.1007/s11134-019-09643-w.

[3] Blockchain France, *La Blockchain décryptée: les clefs d'une révolution*. 2016.

[4] A. M. Antonopoulos, *Mastering bitcoin*, First edition. Sebastopol CA: O'Reilly, 2015.

[5] Z. Zheng, S. Xie, H. Dai, X. Chen, et H. Wang, « An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends », in *2017 IEEE International Congress on Big Data (BigData Congress)*, Honolulu, HI, USA, juin 2017, p. 557-564. doi: 10.1109/BigDataCongress.2017.85.

[6] R. A. Memon, J. P. Li, et J. Ahmed, « Simulation Model for Blockchain Systems Using Queuing Theory », *Electronics*, vol. 8, nº 2, Art. nº 2, févr. 2019, doi: 10.3390/electronics8020234.

[7] H. Huang, W. Kong, S. Zhou, Z. Zheng, et S. Guo, « A Survey of State-of-the-Art on Blockchains: Theories, Modelings, and Tools », *ACM Comput. Surv.*, vol. 54, nº 2, p. 44:1-44:42, mars 2021, doi: 10.1145/3441692.

[8] S. Smetanin, A. Ometov, M. Komarov, P. Masek, et Y. Koucheryavy, « Blockchain Evaluation Approaches: State-of-the-Art and Future Perspective », *Sensors*, vol. 20, nº 12, Art. nº 12, janv. 2020, doi: 10.3390/s20123358.

[9] H. A. Kalodner, S. Goldfeder, A. Chator, M. Möser, et A. Narayanan, « BlockSci: Design and applications of a blockchain analysis platform », USENIX Security Symposium, 2020.

[10] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, et K.-L. Tan, « BLOCKBENCH: A Framework for Analyzing Private Blockchains », in *Proceedings of the 2017 ACM International Conference on Management of Data*, Chicago Illinois USA, mai 2017, p. 1085-1100. doi: 10.1145/3035918.3064033.

[11] L. Alsahan, N. Lasla, et M. Abdallah, « Local Bitcoin Network Simulator for Performance Evaluation using Lightweight Virtualization », in *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, févr. 2020, p. 355-360. doi: 10.1109/ICIoT48696.2020.9089630.

[12] Q.-L. Li, J.-Y. Ma, Y.-X. Chang, F.-Q. Ma, et H.-B. Yu, « Markov processes in blockchain systems », *Computational Social Networks*, vol. 6, nº 1, p. 5, juill. 2019, doi: 10.1186/s40649-019-0066-1.

[13] Q.-L. Li, J.-Y. Ma, et Y.-X. Chang, « Blockchain Queue Theory », in *Computational Data and Social Networks*, Cham, 2018, p. 25-40. doi: 10.1007/978-3-030-04648-4_3.

[14] C.-H. Ng et B.-H. Soong, *Queueing Modelling Fundamentals*. Chichester, UK: John Wiley & Sons, Ltd, 2008. doi: 10.1002/9780470994672.

[15] Y. Kawase et S. Kasahara, « Transaction-Confirmation Time for Bitcoin: A Queueing Analytical Approach to Blockchain Mechanism », in *Queueing Theory and Network Applications*, Cham, 2017, p. 75-88. doi: 10.1007/978-3-319-68520-5_5.

[16] S. Ricci, E. Ferreira, D. S. Menasche, A. Ziviani, J. E. Souza, et A. B. Vieira, « Learning Blockchain Delays: A Queueing Theory Approach », *SIGMETRICS Perform. Eval. Rev.*, vol. 46, nº 3, p. 122-125, janv. 2019, doi: 10.1145/3308897.3308952.

[17] Y. Shahsavari, K. Zhang, et C. Talhi, « A Theoretical Model for Block Propagation Analysis in Bitcoin Network », *IEEE Trans. Eng. Manage.*, p. 1-18, 2020, doi: 10.1109/TEM.2020.2989170.

[18] Q. Zhou, H. Huang, Z. Zheng, et J. Bian, « Solutions to Scalability of Blockchain: A Survey », *IEEE Access*, vol. 8, p. 16440-16455, 2020, doi: 10.1109/ACCESS.2020.2967218.

[19] S. Kasahara et J. Kawahara, « Effect of Bitcoin fee on transaction-confirmation process », *Journal of Industrial & Management Optimization*, vol. 15, nº 1, p. 365-386, 2019, doi: 10.3934/jimo.2018047.

[20] R. Srivastava, « Blockchain and transaction processing time using M/M/1 queue model », International Journal of Recent Technology and Engineering, vol. 7, p. 399-401, janv. 2019.

[21] RR. Bowden, H. P. Keeler, A. E. Krzesinski, et P. G. Taylor, « Block arrivals in the Bitcoin blockchain », arXiv:1801.07447 [cs], janv. 2018

[22] N. Papadis, S. Borst, A. Walid, M. Grissa, et L. Tassiulas, « Stochastic Models and Wide-Area Network Measurements for Blockchain Design and Analysis », in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, avr. 2018, p. 2546-2554. doi: 10.1109/INFOCOM.2018.8485982.

[23] M. Frolkova et M. Mandjes, « A Bitcoin-inspired infinite-server model with a random fluid limit », *Stochastic Models*, vol. 35, nº 1, p. 1-32, janv. 2019, doi: 10.1080/15326349.2018.1559739.

[24] K. Javier, « A Study of Quasi-Birth-Death Processes and Markovian Bitcoin Models », *All Dissertations*, août 2020, [En ligne]. Disponible sur: https://tigerprints.clemson.edu/all_dissertations/2677

[25] M. Lefebvre, *Probabilités, statistique et applications*. Presses inter Polytechnique, 2011.

[26] V. Delsart et N. Vaneecloo, *Probabilités, variables aléatoires, lois classiques*. Presses Univ. Septentrion, 2010.

[27] R. Gallager, Stochastic Processes: Theory for Applications. 2013. doi: 10.1017/CBO9781139626514.

[28] R. Durrett, « Essentials of Stochastic Processes », 1999. doi: 10.1007/978-1-4614-3615-7.

[29] « Discrete Event Simulation - A First Course - Lemmis Park - Free Download PDF ». https://kupdf.net/download/discrete-event-simulation-a-first-course-lemmis-park_5afe247ae2b6f5886233def4_pdf (consulté le juill. 08, 2021).

[30] P. L'Ecuyer et E. Buist, « Simulation in Java with SSJ », in *Proceedings of the Winter Simulation Conference, 2005.*, Orlando, FL. USA, 2005, p. 611-620. doi: 10.1109/WSC.2005.1574301.

[31] P. L. L'Ecuyer, L. Meliani, et J. Vaucher, « SSJ: a framework for stochastic simulation in Java », in *Proceedings of the Winter Simulation Conference*, San Diego, CA, USA, 2002, vol. 1, p. 234-242. doi: 10.1109/WSC.2002.1172890.

[32] A. Maria, « Introduction to modeling and simulation », in Proceedings of the 29th conference on Winter simulation, USA, déc. 1997, p. 7-13. doi: 10.1145/268437.268440.