MASTER EN SCIENCES INFORMATIQUES

Motivations, défis et fonctionnement du stockage distribué

le cas de l'InterPlanetary File System

CRUTZEN, Pierre-Yves

Award date:
2022

Awarding institution:
Universite de Namur

UNIVERSITÉ DE NAMUR

FACULTÉ D'INFORMATIQUE

Motivations, défis et fonctionnement du stockage distribué: le cas de l'InterPlanetary File System

Pierre-Yves Crutzen

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

Université de Namur
Faculté d'informatique
Année académique 2021–2022

Motivations, défis et fonctionnement du stockage distribué: le cas de l'InterPlanetary File System

Pierre-Yves Crutzen

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Jean-Noël Colin

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Avant-propos

Je tiens avant toute chose à remercier mon promoteur pour l'aide qu'il m'a apportée tout au long de la rédaction de ce mémoire. Merci pour sa disponibilité ; sa gentillesse ; son suivi tout au long du travail de recherche. Les réponses et les conseils obtenus durant les différentes étapes d'avancement du mémoire on permis d'orienter la recherche vers un résultat concret du mémoire.
Je tiens également à remercier mes enseignants pour le savoir qu'ils m'ont inculqué, ayant permis d'aboutir à un travail de recherche.

Résumé

Ce mémoire a pour objectif de mener à comprendre pourquoi et comment mettre en œuvre une solution de stockage décentralisé avec l'InterPlanetary File System. Les types d'infrastructures du web, les systèmes pair-à-pair, les systèmes de fichiers décentralisés, ainsi que le protocole IPFS sont mis en relation dans un contexte de décentralisation du stockage des données du web. Pour ce faire, le protocole IPFS et son système de fichiers sont étudiés en profondeur afin de comprendre leur fonctionnement, leur contexte d'utilisation et ce qui les différencie des autres systèmes afin de fournir une solution interopérable pour l'hébergement et le partage de données en pair-à-pair. Les moyens que la littérature scientifique présente afin d'y arriver sont étudiés dans un état de l'art. Les critères et les défis que soulèvent les systèmes de stockage de fichiers et les solutions de communication pair-à-pair sont mis en perspective afin de comprendre la relation qu'ils peuvent avoir pour mener le web vers une distribution complète des infrastructures grâce aux réseaux pair-à-pair. Une étude de cas d'application du protocole IPFS pour le stockage décentralisé et le partage de contenu sur le Web, ainsi que la mesure de ses performances pour l'échange de données sont étudiés. Les manières dont cela peut être mis en œuvre, en perspective des contraintes d'application liées au contexte du web distribué, sont ensuite soulignées. Enfin, les résultats de la recherche sont discutés et permettent d'aboutir à une conclusion, ainsi qu'à une proposition d'arbre de décision permettant de comprendre dans quel cas l'InterPlanertary File System peut être pertinemment utilisé comme système de stockage dans une infrastructure web décentralisée.

Table des matières

Chapitre 1

Introduction

Le *World-Wide-Web*, communément appelé « le Web », est un réseau de communication qui utilise des liens hypertextes pour partager du contenu consultable à l'aide de logiciels de récupération du contenu tels que les navigateurs web. Internet correspond quant à lui à un ensemble d'infrastructures réseau qu'utilise le World-Wide-Web afin de communiquer à l'aide d'une connexion internet. La combinaison du World-Wide-Web et de l'Internet permet de fournir un ensemble de services d'échange des données de tous types (images, texte, fichiers, vidéos, bandes sonores, etc.).

Le mode de vie actuel de la Société implique qu'il est devenu presque essentiel de vivre avec les moyens de télécommunications tels que la téléphonie, les réseaux internet, les systèmes satellites ou bien les systèmes hybrides. Tout devient désormais ultra-connecté. De manière plus ciblée, l'Internet est une ressource dont il est devenu difficile de se passer au quotidien, car il augmente la productivité, il aide à la communication, à l'information, aux achats en ligne, à la gestion financière, etc. La communauté des utilisateurs dépendant du World-Wide-Web et de son accessibilité s'est étendue avec les années par le déploiement des réseaux et par la réduction des coûts de mise en place. [66, 16, 6, 80]
Pour continuer de garantir l'instantanéité des moyens de communication en proposant tous les services imaginables à portée de main et accessibles directement depuis notre poche, nous avons besoin que le Web soit toujours disponible, performant, sûr et fiable. En l'occurrence, nous voulons que notre identité et que nos données restent en sécurité quels que soient les services en ligne que nous utilisons [9, 80].
Cependant, les entreprises qui contrôlent les technologies de communication ont des intérêts divergents de ceux des utilisateurs. De manière générale un utilisateur lambda d'une technologie désire obtenir un service en toute sécurité sans que les données qu'il fournit ne soient utilisées contre sa volonté. Les entreprises, quant à elles, traitent les données des utilisateurs et en ont un contrôle total en échange du service fourni, ce qui leur permet d'utiliser les informations suivant leur propres intérêts.[9, 103]

Le modèle actuel du web centralisé a souvent été remis en question par des chercheurs au fil des ans afin de le faire évoluer. Ceux-ci identifient et développent de meilleurs moyens d'exploitation de systèmes toujours plus efficaces que ceux que nous connaissons. Qu'il s'agisse de solutions à intégrer au web existant, ou encore de solutions qui tentent de revisiter complètement le modèle du Web et de résoudre des problèmes bien connus par de nouvelles approches plus innovantes, voire qui tentent de combler des besoins non pris en compte ou non ciblés par les protocoles du Web centralisé [14, 42, 16, 36].

La décentralisation du web est un processus de distribution d'éléments distants d'un emplacement ou d'une autorité centrale, tels que des applications, des fonctions, des pouvoirs, etc. [6]. En l'occurrence, les technologies décentralisées et *open source* montrent que des systèmes peuvent être construits en privilégiant une souveraineté individuelle, afin de s'éloigner des systèmes de contrôle centralisés auxquels il est nécessaire de faire confiance en matière de disponibilité des ressources, de confidentialité, ou encore d'intégrité des données échangées. La Web3 Foundation [1], par ailleurs, a été créée afin de gérer et de développer les applications et technologies qui gravitent autour des protocoles Web décentralisés. Elle souligne les intérêts de privatiser les données des utilisateurs à l'aide des systèmes décentralisés pour que ces dernières ne puissent pas être exploitées sans qu'elles ne soient explicitement fournies par les utilisateurs. Elle s'intéresse à des domaines récents, tels qu'aux technologies *blockchains*, aux infrastructures de mise en réseau pair-à-pair ; aux systèmes cryptographiques, aux systèmes de publication de données tel qu'IPFS. [103]
Dans cette même optique, les configurations pair-à-pair (P2P) du web décentralisé permettent d'offrir des plate-formes et services que l'Internet centralisé ne propose pas. La communauté du Web s'intéresse d'ailleurs

1. Web3 Foundation, Site officiel de la Web3 Foundation, `https://web3.foundation/` (2022) (Date d'accès 01/07/2022)

de plus en plus à ce type d'infrastructures depuis plusieurs années [84]. L'arrivée des moyens de paiements décentralisés et des cryptoactifs sur les réseaux pair-à-pair, ainsi que du web distribué, ont exacerbé l'intérêt de la communauté du Web pour ces systèmes distribués. Les systèmes pair-à-pair ont ouvert des perspectives vers de nouveaux protocoles réseau totalement décentralisés et vers des technologies ne nécessitant plus l'intervention d'un serveur central, faisant office d'autorité de confiance pour des domaines d'application tels que la bureautique, l'éducation, le divertissement, la domotique, les transports, la surveillance de l'environnement, la sécurité, la santé [49, 6]. Par ailleurs, la *Decentralized Identity Foundation* (DIF) est une organisation qui soulève des enjeux associés à l'administration centralisée des identités sur le Web. Elle propose un contexte standardisé pour décentraliser l'Identité des utilisateurs, permettant ainsi d'assurer un contrôle de l'identité numérique de chacun par l'emploi d'un système interopérable pour les interactions entre des systèmes informatiques et leurs utilisateurs. Grâce à ces systèmes, chacun peut partager des données sur le web sans devoir s'authentifier à l'aide de données d'identification privées, qui risquent d'être utilisées de manière opaque par les tiers. [2] [9].
Ces organisations tendent donc à participer à l'évolution du web centralisé vers un web complètement décentralisé, où l'utilisateur a le contrôle de son identité et de ses données ; c'est le Web 3.0.

L'InterPlanetary File System (IPFS) est un protocole créé par *Juan Benet*, qui a ensuite fondé *Protocol Labs* en 2014. Il propose un système de stockage distribué qui permet l'hébergement et le partage de contenu de manière partiellement ou complètement décentralisée à l'aide des réseaux P2P [29]. Le système fournit son propre protocole pair-à-pair pour le stockage et la transmission distribués des données de type hypermédia, versionnées et adressables par le contenu. Il fournit une solution de publication de contenu permanente décentralisée semblable au World-Wide-Web [6, 22, 88]. Pour ce faire, il associe des techniques des systèmes les plus performants parmi les résultats éprouvés et répertoriés dans la littérature scientifique [14]. Cela permet d'obtenir un seul système cohérent, évolutif et simple à utiliser. Il doit proposer une vision ambitieuse d'internet par une infrastructure décentralisée et pour laquelle des applications de nombreux types peuvent être construites.

Les nouveaux protocoles de communication fournis par la communauté d'IPFS , tels que le protocole d'échange *Bitswap* et le protocole pair-à-pair *Libp2p* pour la couche réseau sont interopérables et peuvent être utilisés par d'autres systèmes externes à IPFS [32, 14]. Le protocole IPFS vise à combler les lacunes du protocole HTTP en résolvant certaines faiblesses du web centralisé et pourrait être destiné à le remplacer à l'avenir [80] ; en direction d'un web davantage décentralisé : le Web 3.0 [103].

Les solutions proposées par le protocole IPFS paraissent étroitement liées aux principes de décentralisation du stockage pour le Web. En effet, le protocole cherche à se libérer des contraintes de dépendances à des systèmes de stockage centralisés. De plus, les technologies des réseaux pair-à-pair conjuguées avec ce protocole doivent permettre d'apporter de nouvelles solutions destinées à la décentralisation du stockage de données partagées entre des utilisateurs d'un système applicatif, au sein d'architectures web.

Tout ceci amène à se poser la question de recherche suivante : « Pourquoi et comment mettre en œuvre une solution de stockage décentralisé ; le cas d'IPFS ? ». Cette question induit plusieurs sous-questions qui permettront d'y répondre, telles que « Pourquoi faire du stockage décentralisé ? » ; « Quelles solutions de stockage décentralisé existent ? » ; « Quels défis ces solutions soulèvent-elles ? » ; « Quels sont les principes de fonctionnement d'IPFS ? » ; « Quelles sont les performances d'IPFS ? ». Par ailleurs, il convient de comprendre l'étendue de l'IPFS et ce que ce protocole peut apporter de manière générale. La problématique suivante a donc été identifiée : « Quelles sont les solutions apportées par l'IPFS ? ».

Dans ce mémoire, le terme « Web » est utilisé de manière non distinctive pour décrire l'ensemble des infrastructures et des services fournis à la fois par l'Internet et par le World-Wide-Web pour l'échange et de mise à disposition de contenu.

Dans le chapitre 2, l'état de l'art de la littérature scientifique relative à l'IPFS est établi afin de répondre à la problématique identifiée. Il y est décrit les solutions que le protocole IPFS tente d'apporter au web actuel pour le stockage des données, allant vers une décentralisation des systèmes. Les concepts qui y sont associés sont présentés, ses principes de fonctionnement, ses utilisations possibles et actuelles, ses avantages et inconvénients, ainsi que des résultats de mesures de performances du système et de sous-systèmes.
Dans le chapitre 3, une étude est menée en profondeur afin de répondre à la question de recherche. L'étude est menée sur base de la réalisation d'une application *Proof of Concept* décentralisée pour un paiement par

2. Decentralized Identity Foundation, "Presentation Exchange - Decentralized Identity Foundation", `https://identity.foundation/presentation-exchange/` (2022) (Date d'accès 02/07/2022)

transactions bancaires entre deux individus à l'aide d'un système de *scanner* de QR codes embarqué dans le système applicatif. L'expérience doit aider à comprendre comment mettre en œuvre l'IPFS pour un système de stockage décentralisé, à comprendre ses limites, ses cohérences, ses incohérences et ses orientations possibles lors de la réalisation d'une application nécessitant d'assurer la sécurité des données échangées. Il est également introduit la manière d'utiliser le système IPFS et des moyens de déploiement de l'application au sein de noeuds du réseau. Enfin, les performances du système sont évaluées à partir d'un nombre de noeuds d'hébergement de contenu restreint afin de mesurer la latence et le débit du réseau lorsque des données de tailles variables sont demandées avec un taux de parallélisme de requêtes élevé. Les résultats de la recherche effectuée sont présentés.
Dans le chapitre 4, une discussion d'interprétation des résultats de la recherche effectuée (chapitre 3) est faite avec l'état de l'art (chapitre 2). Elle reprend les points essentiels à retenir concernant la réalisation du travail de recherche, ainsi que les réponses aux sous-questions de la question de recherche. Les manques dans la recherche sont abordés, ainsi que ce qui pourrait être réalisé comme travail de recherche futur.
Dans le chapitre 5, la conclusion du travail permet de récapituler l'objectif du travail et de répondre à la question de recherche. Une proposition d'arbre de décision doit permettre d'identifier l'adéquation de l'InterPlanetary File System à son utilisation pour faire du stockage de données décentralisé.

Chapitre 2

Etat de l'art : Quelles sont les solutions apportées par l'IPFS ?

2.1 Mise en contexte

Dans ce chapitre, l'état de l'art est établi à partir de la littérature scientifique afin de mettre en exergue ce qu'apporte le protocole IPFS.
La première section de ce chapitre présente ce qui amène les systèmes à se diriger vers une décentralisation des données et des infrastructures des systèmes web, ainsi que différents moyens permettant de la réaliser. Il est d'abord présenté les principes de centralisation des données sur le web, les enjeux que cela soulève, ainsi que la manière dont cela peut être résolu à l'aide de la décentralisation des systèmes. Plusieurs systèmes décentralisés abordés dans la littérature sont présentés et l'intérêt des systèmes pair-à-pair est étudié. Enfin les systèmes de *blockchains* et plusieurs systèmes de stockage de données qui les utilisent sont abordés afin de comprendre leur intérêt pour la décentralisation des données du web.

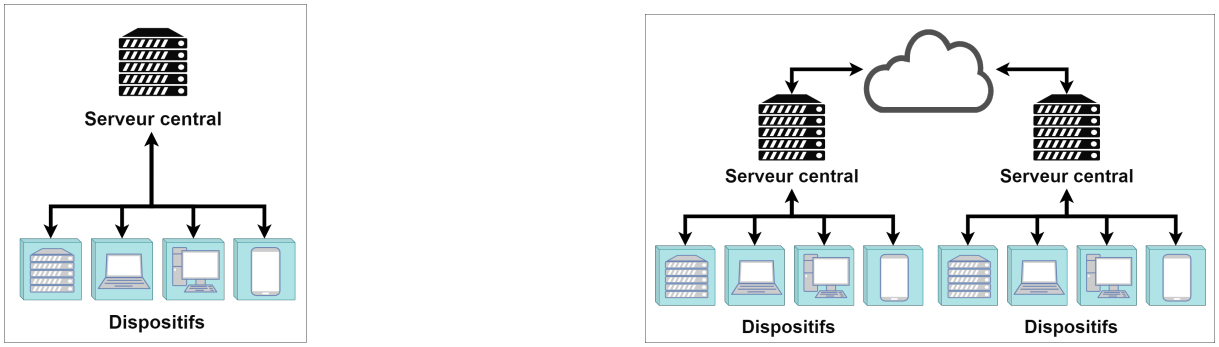
Dans la seconde section, les principes de fonctionnement de l'InterPlanetary File System sont présentés. Est étudiée la façon dont il peut intervenir afin d'amener le web vers une décentralisation complète des données, ainsi que les systèmes qui constituent le protocole IPFS pour constituer un web distribué. Ses objectifs, ses caractéristiques, ses avantages et inconvénients sont présentés et mis en relation avec des solutions associées. Ses utilisations et ses performances provenant de la littérature scientifique sont également présentés.

2.2 Evolution vers une décentralisation du Web

2.2.1 Le Web centralisé

Le Web centralisé correspond à un ensemble de machines dont des ressources sont mises en commun pour former un réseau de communication dont les informations transitent par l'intermédiaire de serveurs centraux. Il peut être localisé (serveurs physiques) ou délocalisé (systèmes cloud) [16, 29]. Ce type de réseau est géré par une seule entité logique, et est caractérisé par la présence de points de défaillances uniques, qui empêchent toute communication entre l'Internet et un réseau dépendant d'un serveur central si celui-ci est indisponible [5, 29]. Le Domain Name System (DNS) est un service qui attribue des noms aux adresses IP permettant de les résoudre lors de leur utilisation sous la forme de noms de domaine en se référant à un serveur DNS. L'organisation ICANN gère le DNS et les serveurs racine des noms de domaine. Ces serveurs agissent en tant qu'autorités de confiance centralisés, ainsi que des points de défaillance unique. [4]
La centralisation de l'information concerne les données échangées, les identités des utilisateurs, ou encore les certificats d'authenticité fournis par des autorités de confiance. Dans ce principe, les bases de données locales des différentes parties sont synchronisées à l'aide d'un système de registre électronique centralisé. Elles sont maintenues et contrôlées par un tiers de confiance. La gouvernance et le contrôle centralisés mènent à l'apparition de silos de données, causant des risques de censure, d'indisponibilité, de failles de sécurité, de perte de confidentialité des données. Cela peut mener à un choix d'utiliser des fournisseurs de stockage *cloud*. La figure 2.1a représente la manière dont des dispositifs communiquent avec un serveur central pour le web centralisé. La figure 2.1b représente la manière dont des dispositifs communiquent avec un système *cloud* pour le web centralisé.

Si les données d'une entreprise ou d'une personne physique sont enregistrées de manière centralisée, l'hôte engage sa responsabilité afin d'assurer la localisation, la conservation, la disponibilité et la sécurité de ces données. Par ailleurs, dans le cas de données stockées à l'aide de service *cloud*, il convient aux systèmes *cloud*



(a) Système centralisé localisé

(b) Système *cloud* centralisé

FIGURE 2.1 – Les dispositifs communiquent avec le web centralisé à l'aide de serveurs centraux. Cela peut être fait de manière localisée (serveurs centraux), ou délocalisée (systèmes *cloud*).

concernés de préserver leur confidentialité en les chiffrant. La présence d'un tiers de confiance, faisant office d'autorité de confiance pour fournir des clés de sécurité de manière centralisée, ou pour certifier l'identité d'un utilisateur est un goulot d'étranglement. En l'occurrence, la sécurité du système est affectée si des données sur un service *cloud* sont déchiffrées par un utilisateur tiers qui n'est pas digne de confiance. [29, 27]
De plus, dans ce type de système, l'envoi de données nécessite potentiellement l'utilisation d'un grand nombre de serveurs intermédiaires pour traiter et transmettre l'information. Ils stockent des données en base de données ; ils analysent les données ; ils envoient une notification à un destinataire pour qu'il récupère les données depuis une base de données dédiée ; ils fournissent des services d'authentification ; etc. [27]

2.2.2 Adressage de localisation

Dans le web partiellement ou complètement centralisé, les machines se localisent afin de pouvoir communiquer ensemble. Les données de toutes formes sont accessibles sur base de liens qui donnent accès à des ressources de manière interactive. L'adressage de localisation permet à des intervenants quelconques de communiquer par l'utilisation d'URL (*Uniform Resource Locator*). Les données peuvent être récupérées grâce à leur centralisation (l'hébergement) par une autorité de confiance (DNS). Les noms de domaine des URL indiquent à quelle autorité de confiance il faut s'adresser pour récupérer les données. [72, 14, 16, 80]

Faiblesses de l'adressage de localisation sur le web centralisé

- **Sûreté :** La fiabilité du contenu auquel un utilisateur accède repose sur des informations ou des suppositions qu'il possède. En effet, rien ne garantit qu'une ressource accessible est plus fiable qu'une autre et que le contenu est sûr. Il n'est pas non plus possible de vérifier avec certitude qu'un contenu disponible depuis une URL spécifique est fiable, car il est facile d'être piégé par des acteurs malveillants. L'utilisateur fait donc souvent confiance aveuglément à la source d'informations, tout en dépendant d'autorités centrales qui répertorient les sources en tant que sources fiables ou malveillantes [6, 72, 14] ;
- **Localisation du contenu :** Rien ne garantit la géolocalisation des adresses des données récupérées sur le réseau de manière fiable. De plus, il n'est pas possible d'obtenir l'URL d'un contenu désiré s'il est disponible sur le web sans connaître le nom de domaine de l'hébergeur, ainsi que le nom du contenu [79, 14] ;
- **Duplication :** Le web centralisé contient une pléthore de données dupliquées enregistrées sur une ou plusieurs sources (URL) différentes, et qui possèdent parfois uniquement un nom différent. Il n'est pas évident de différencier quels sont les objets de données identiques, ou ceux qui sont deux versions différentes et en quoi ils diffèrent [72] ;
- **Disponibilité :** En cas de panne, ou de problème technique sur les serveurs d'un domaine concerné, du contenu peut devenir indisponible temporairement ou définitivement. Cela peut être causé par un problème de maintenance, par une attaque provenant de personnes malveillantes, par l'abandon des services de données, ou autres [79, 14] ;
- **Intégrité :** Malgré un contexte sécurisé, de nombreux événements peuvent survenir ne garantissant pas l'intégrité des données de l'utilisateur [16] ;
- **Confidentialité :** L'utilisateur n'a pas de contrôle absolu concernant l'utilisation des données le concernant qui sont stockées, traitées et échangées par des tiers sur le réseau [26].

IPFS tente de répondre à ces lacunes par l'utilisation de l'adressage de contenu (voir 2.2.5) et la décentralisation partielle ou complète du web [14, 83, 29].

2.2.3 Les systèmes *cloud*

Les systèmes de stockage basés sur le *cloud* ont plusieurs caractéristiques [29, 33] :

1. Un fournisseur de stockage partage des ressources de stockage sur un réseau et les alloue à la demande aux clients ;
2. Le fournisseur de stockage localise les utilisateurs et enregistre, partage, sécurise les données ;
3. Les ressources utiles aux utilisateurs restent accessibles à l'aide de systèmes normalisés et sans le besoin d'une intervention humaine ;
4. L'allocation des ressources réagit dynamiquement en fonction de l'utilisation du système ;
5. Un service de contrôle des ressources est fourni pour pouvoir consulter les mesures de consommation ;
6. Les utilisateurs paient uniquement pour utiliser les ressources ou les services qu'ils désirent utiliser.

Systèmes de stockage *cloud* centralisé

Le modèle de stockage *cloud* centralisé utilise des serveurs dédiés pour le partage et le stockage des données mises sur des serveurs distants des fournisseurs de stockage. L'infrastructure de stockage étant généralement détenue et gérée par une seule entité logique. [16, 29].

Avantages par rapport aux systèmes de stockage localisés

— **Moins coûteux :** n'utilisent pas directement des serveurs dédiés pour le stockage de données, éliminant de coûteux investissements en matériel [16] ;
— Plus fiables [16] ;
— **Moins sujets aux pertes de données :** les données sont réparties sur plusieurs lieux stratégiques [16].

Faiblesses :

— **Sécurité :** La manière dont les données sont sécurisées est inconnue. Rien n'assure l'absence absolue de failles malgré les solutions de sécurité améliorées développées durant ces dernières années. Les données ne sont pas toujours chiffrées. De plus, les objets de données ne sont pas fractionnés et répartis sur plusieurs nœuds du système global [16, 26] ;
— **Disponibilité :** Manque de transparence concernant la disparition ou la compromission de données [26, 16] ;
— **Contrôle des données stockées :** Rien n'assure aux clients que leurs données ne sont pas vendues, copiées, altérées. Il y a un manque de transparence quant au lieu de stockage des données, qui y accède, comment et quand les données sont traitées ;
— **Confiance :** Peu de contrats officiels entre les clients et fournisseurs de services. Il n'existe pas toujours un cadre juridique suffisant permettant aux utilisateurs de réclamer des réparations si leurs données sont compromises, vendues à des tiers, voire endommagées [26] ;
— **Coûts :** Plus élevés que les système distribués. Il est généralement demandé de payer avant la réservation du stockage [26, 16] ;
— **Bande passante :** Plus faible que pour les systèmes distribués, les données téléchargées sont récupérées depuis une connexion auprès d'un serveur rendu disponible par le fournisseur de services [16] ;
— **Confidentialité :** Pas de garanties d'anonymat, car généralement, une personne doit s'authentifier dans le réseau avec ses informations personnelles qui prouvent son identité et un numéro de compte bancaire doit être transmis pour effectuer des paiements [16, 39, 26].

2.2.4 Le Web décentralisé

La décentralisation du Web tend à modifier la mise en oeuvre des systèmes qui composent la pile de protocoles dans les communications réseau des modèles du web.
D'une part, contrairement au web centralisé dans lequel les données sont synchronisées à l'aide de bases de données dépendant de l'architecture des systèmes localisés ou délocalisés, les sources de données sont réparties sur le réseau afin de ne pas dépendre de points de sauvegarde uniques dans les systèmes, ainsi que pour augmenter la disponibilité des données.
Ensuite, les systèmes de vérification des identités pour l'authentification sont externalisés afin de réduire la dépendance à un stockage des données d'identité à un seule entité. [9, 42]
Par ailleurs, les dépendances à des tiers de confiance agissant comme autorité unique dans les systèmes sont également voués à être réduites. En effet, les responsabilités de prise de décisions sont réparties sur le réseau afin de ne pas faire confiance à une seule source de vérité [85, 9]. Un système décentralisé peut être partiellement décentralisé ou complètement décentralisé ; on parle dans ce cas d'un système distribué [16]. Plus le

système décentralisé doit atteindre un taux de décentralisation élevé, moins il convient de faire confiance aux systèmes et utilisateurs externes.
Les données sont donc réparties sur le réseau en fonction d'un système de stockage utilisé, tout en assurant que l'accès reste sécurisé pour combler le besoin d'un tiers de confiance [4] et donc pour continuer à assurer l'authenticité des transactions de communication des données. Chaque nœud de ce type de réseau décide et agit à partir des informations qu'il possède. Le résultat obtenu par l'ensemble du système aboutit à une réponse globale des nœuds. [85]

Le système d'attribution des noms, qui est associé au service DNS pour le web centralisé, est également concerné par la décentralisation du Web. En effet, étant donné que le DNS dépend de serveurs centraux permettant de résoudre les noms de domaine associés aux adresses IP, le système d'attribution des noms est concerné par la décentralisation. L'adressage par le contenu (ou adressage de contenu) est une solution (voir 2.2.5). [14, 4, 67, 104]

Enfin, les applications décentralisées doivent avoir des temps de réponse courts, au moins similaires à ceux des services du web centralisé et une capacité de stockage répondant à leurs besoins [6, 4].

Intérêt du stockage décentralisé

La décentralisation du stockage des données permet de ne pas dépendre de la gouvernance d'une autorité centrale à laquelle faire confiance pour utiliser des services sur le Web. Elle permet donc aux utilisateurs d'un système de garder le contrôle de leurs données privées, qui dépendent cependant du taux de décentralisation des applications utilisées. [9] reprend notamment le terme de « l'identité décentralisée » qui qualifie les données d'identité digitale dédiées à une utilisation sur le Web décentralisé. De plus, le terme « d'identité auto-souveraine » considère que chaque utilisateur doit avoir un contrôle concernant l'interopérabilité et l'administration de son identité numérique, sans qu'elle ne soit dépendante de services tiers et que les données soient enregistrées à différents endroits de manière intraçable, voire incontrôlable par les applications web.

Par ailleurs, dans ce concept de décentralisation du stockage, les données utilisées au sein d'un système ne sont donc pas vouées à rester de manière indélébile dans des bases de données incontrôlables. C'est l'utilisateur qui les conserve, sinon un service tiers à l'application utilisée que l'utilisateur peut plus ou moins contrôler. Par exemple, les systèmes de type *Single-Sign-On* permettent une décentralisation partielle des services d'authentification à l'aide de tiers de confiance qui centralisent les données d'identification [9]. Dans le cas d'une décentralisation totale des données d'une application, les données stockées peuvent être reliées à des systèmes de stockage externes tel que le système de fichiers IPFS pour la distribution de contenu. Elles peuvent également être reliées à des systèmes de stockage basés sur les *blockchains* (voir 2.2.11) afin de décentraliser les informations sans l'intervention de tiers de confiance, tout en garantissant la traçabilité des données en complément d'un portefeuille de cryptoactifs qui externalise les données d'authentification des systèmes et permet aux utilisateurs de s'authentifier à l'aide de services auto-souverains pour l'identité décentralisée [10, 85].

Ainsi, moins les données du système sont centralisées, moins le risque est élevé au sein des systèmes de rencontrer des problèmes de sécurité, de confidentialité, d'intégrité, d'indisponibilité. En effet, les données ont un risque moindre d'être utilisées d'une manière involontaire par des systèmes tiers, d'être modifiées et utilisées de manière détournée, de causer des indisponibilités ou des fautes dans les systèmes car elles sont externalisées. [15]

Systèmes de stockage *cloud* décentralisés

Contrairement au stockage centralisé, les systèmes basés sur le stockage décentralisé n'utilisent pas directement des serveurs dédiés pour le stockage de données. Dans ce modèle de stockage en réseau, chaque machine qui stocke des données (hôte) correspond à un nœud du système. Il utilise un logiciel permettant d'allouer de l'espace de stockage indépendamment des autres nœuds du système global. L'ensemble des nœuds forme un *cloud*. Un système de stockage distribué est un système décentralisé. Le traitement des données y est partagé entre plusieurs nœuds, mais les décisions peuvent être prises de manière centralisée ou décentralisée. Les données peuvent être chiffrées et découpées depuis un hôte, avant d'être ensuite envoyées sur le stockage décentralisé, et distribuées aux nœuds du système (les fournisseurs de stockage). Des algorithmes interviennent pour distribuer géographiquement les données, répartir la charge, ou tout autre élément pertinent. De plus, il peut être nécessaire que les nœuds du système prouvent que les données stockées ne sont pas altérées durant le stockage. Cela peut être fait à l'aide de *smart contracts* spécifiant le coût et la durée pour le stockage [26, 16, 42, 43].

Avantages par rapport aux systèmes *cloud* centralisés :

- **Sécurité :** L'utilisation de la technologie *blockchain* surmonte efficacement les problèmes de sécurité. Le chiffrement des communications de bout en bout entre les clients assure une transmission des données à travers un réseau complètement décentralisé. Cela élimine les risques de défaillance des données provenant des contrôles centralisés [43] ;
- **Sûreté de l'information :** Meilleure que les autres types de stockage de données pour assurer l'authenticité de la provenance des données. Les systèmes de stockage de fichiers complètement distribués sont basés sur la technologie de la *blockchain*. Ils n'ont pas de serveur central servant de tiers confiance pour contrôler le réseau [43, 6] ;
- **Coût :** Capacité de stockage supérieure pour un coût moindre que les systèmes *cloud* centralisés [16] ;
- **Disponibilité :** Optimisation de l'espace de stockage disponible. Les utilisateur peuvent louer leur espace disque inutilisé à un tiers en ayant besoin [16, 6].

Inconvénients :

- Des milliers de nœuds sur le réseau (serveurs) doivent communiquer pour que le réseau soit optimal en termes de disponibilité ou de performances [16] ;
- **Création de silos de données :** Les systèmes de gouvernance et de contrôle centralisés ont une autorité concernant la validation des points d'accès aux données. La disponibilité, l'accessibilité et la confidentialité des données peuvent être impactées en cas de blocage des accès aux sources de données échangées. [29] ;
- **Risque de pertes et d'indisponibilités des données** par manque de nœuds disponibles sur le réseau, sinon, par des défaillances des serveurs. Il faut alors répliquer des données ou utiliser des codes d'effacement (*erasure codes*) afin d'améliorer la résilience du stockage [16, 18, 29].

2.2.5 Adressage de contenu

L'adressage de contenu tend à résoudre des faiblesses de l'adressage de localisation en permettant aux nœuds d'un réseau de communiquer sans qu'ils ne doivent se localiser [83, 14]. Il peut être utilisé pour tous les types de données, telles que les données structurées (objets de données) et les données non-structurées (fichiers). Il consiste à calculer un *hachage cryptographique* de taille fixe depuis des données de toute taille et de tout type. Le hachage provenant du protocole web décentralisé utilisé peut être dérivé du contenu des données, ce qui signifie que toute personne utilisant le même algorithme sur les mêmes données arrivera au même hachage. Si les hachages cryptographiques de deux objets de données sont comparés et que l'on confirme qu'ils sont identiques, il est garanti que chaque bit de ces deux fichiers est identique. Pour toute modification de cet objet de données, telle qu'un caractère dans un fichier texte, le hachage cryptographique de l'objet initial sera différent de celui du nouvel objet de données. [79, 14, 80, 29]

Garantie du contenu

Contrairement à l'adressage de localisation du web centralisé, dont le contenu d'une adresse source (URL) peut changer avec le temps, la récupération des données à l'aide d'une adresse de contenu assure de fournir la version prévue de ces données. Pour récupérer du contenu avec cette méthode, une requête est faite à tout le réseau connecté pour récupérer la ressource désirée à l'aide du hachage cryptographique qui y correspond. Si une source possédant la ressource désirée est en ligne, le fichier peut être récupéré. [80, 57]

Cadre d'application de l'adressage de contenu

IPFS [14], Git et des systèmes basés sur les *blockchains* tels que Blockstack [4], Ethereum [104], ou encore Bitcoin [67] utilisent l'adressage de contenu [14, 32]. Ils diffèrent cependant dans la manière d'interpréter les données et dans la fonction cryptographique qu'ils utilisent pour le hachage [83].

Avantages de l'adressage de contenu

La sureté du contenu varie en fonction des URL, car il n'est possible de faire confiance qu'à un nombre d'autorités limitées. Certains acteurs malveillants utilisent les lacunes de l'adressage de localisation pour tromper un utilisateur à propos de contenu des fichiers qu'il récupère. Le hachage des données utilisé par l'adressage de contenu permet de faire confiance aux informations partagées sans devoir se soucier de la suspicion de malveillance des pairs qui hébergent les données. L'adressage de contenu permet de s'affranchir des inconvénients de l'adressage de localisation, tels que les pannes d'un serveur donnant accès au contenu, le changement de format, de nom, ou de répertoire d'un objet de données, le blocage de l'accès à un DNS. [80, 29, 14]

2.2.6 Systèmes de stockage distribués

Les systèmes de stockage distribué (DFS) sont des systèmes fournissant une communication de type client-serveur dans un réseau de nœuds. Ils fournissent un système de partage de fichiers répondant à certains critères de disponibilité, de tolérance aux pannes, d'évolutivité et de performances [36, 80]. Le moyen d'accès aux fichiers se fait à l'aide d'un espace de nom (*namespace*) au sein d'une arborescence des fichiers dans le système. La répartition de la charge de travail peut varier en fonction du type de modèle réseau. [95, 89, 16]

Le tableau 2.1 répertorie une liste non-exhaustive d'applications offrant un système de stockage ou de fichiers distribués, chacune ayant des contextes d'application distincts.

Stockages de données		
Nom	**Objectifs**	**Caractéristiques**
Bigtable File System [21]	Système de stockage de données compressées distribué	Il s'appuie sur un système de fichiers distribué pour augmenter sa durabilité et il fournit à la fois une distribution et des données structurées. Il est conçu pour gérer de très grands volumes de données (des pétaoctets sur des milliers de serveurs). Il utilise le Google File System (GFS) pour conserver ses fichiers de données et de journalisation. [21, 54, 95]
Cassandra [54]	Système de stockage de données distribué compatible avec HDFS qui gère de grands volumes de données structurées réparties sur de nombreux serveurs de base.	Conçu pour du matériel bon marché et pour gérer un débit d'écriture élevé sans diminuer la qualité de la lecture. Il requiert des centaines de nœuds pour proposer un service hautement disponible et sans point de défaillance unique. Il fournit aux clients un modèle de données simple, qui prend en charge le contrôle dynamique du format des données et de la présentation. Il ne supporte pas des modèles de données complètement relationnels.
Stockages de fichiers		
Nom	**Objectifs**	**Caractéristiques**
Andrew File System (AFS)	Système d'archivage de fichiers distribué inspiré de NFS [56]	Encore largement utilisé dans le monde [18]. Offre aux utilisateurs l'abstraction d'un espace de noms unique cohérent et partagé par plusieurs clients [95]. Fait la distinction entre les clients et les serveurs dédiés. Les clients stockent les *namespaces* et les serveurs stockent les données. Les serveurs fédérés sont disposés en *clusters* inter-connectés pour permettre l'évolutivité. Les ressources de stockage ont un répertoire commun. Un système de réplication crée des copies de fichiers en lecture. Fournit une mise en cache des fichiers locaux. [56]

Ceph	Système de fichiers distribué.	Propose des politiques de distribution de données redondantes sur des étagères de disques et *racks* afin de tolérer les risques de fautes. Analyse la disposition physique d'un centre de données. L'emplacement des données est calculé au lieu d'être stocké explicitement. Cela réduit la charge sur les serveurs de méta-données. Les méta-données sont stockées dans un journal persistant à l'extérieur des serveurs de méta-données. Rééquilibrage des méta-données sans avoir à déplacer de données. Un client peut extraire les parties d'un fichier de différents serveurs en parallèle, permettant un débit élevé. Non tolérant aux partitions. Coût élevé de l'ajout et de la suppression de nœuds en raison du rééquilibrage. [57, 18]
Google File System (GFS)	Système de stockage de fichiers distribué propriétaire développé par Google.	Optimisé pour fonctionner fiablement et efficacement sur de grands *clusters* de calcul avec du matériel peu coûteux. Plusieurs répliques de chaque fichier sont maintenues pour une plus grande fiabilité et disponibilité. Les fichiers traités sont volumineux et sont divisés en fragments de 64 Mo. Les fichiers sont en lecture seule après leur écriture. Il est nécessaire de les écraser pour les modifier. [21, 18, 54, 95]
Hadoop File System (HDFS)	Système de stockage de fichiers distribué pour fournir une surcouche de stockage optimisée pour le *framework* de calcul distribué MapReduce. [18]	Adéquat pour un *cluster* composé de milliers de serveurs. Les serveurs exécutent les tâches de calcul des applications clientes et hébergent le stockage. La quantité des ressources dédiées au calcul et au stockage varient à la demande, tout en restant économique. Tolérant aux pannes. Les fichiers ne peuvent être modifiés que par un seul utilisateur à la fois. Adapté au stockage de fichiers de données volumineux. [89, 95]
Lustre	Espace de travail optimisé pour les applications en coopération sur des superordinateurs [18, 95]	Destiné aux *clusters* de calcul modulaires de grandes tailles. Compatibilité avec les les composants réseau et le matériel de stockage inter-connectés. Disponible pour Linux uniquement. [95]
OceanStore [51]	Système de stockage de fichiers distribué évolutif pour fournir un accès sécurisé et hautement disponible aux objets persistants partout dans le monde. [16, 36, 7, 51]	Prend en charge des données nomades qui permettent l'accès à l'information, en la rapprochant de l'endroit où elle est demandée. Elles sont analysées et permettent une réponse rapide aux attaques de déni de service et à des pannes régionales. Les opérations côté serveur telles que la résolution de conflits doivent être effectuées avec des informations chiffrées afin d'éviter une manipulation des serveurs par un individu malveillant. [51, 36]

Self-Certified Filesystem (SFS) [60]	Propose une implémentation de chaînes de confiance distribuées (*distributed trust-chains*) [14]. Propose une technique de création de systèmes de fichiers auto-certifiés, qui ne nécessitent pas d'autorité centrale pour gérer les *namespaces*.	Toutes les instances SFS partagent égalitairement un seul *namespace* global, permettant de gérer une politique de clés cryptographiques dont les noms sont alloués arbitrairement. Le nom d'un système de fichiers certifie son serveur. L'utilisateur peut vérifier la clé publique du serveur, sécuriser le trafic, partager un message sercret. Les fichiers sont accessibles à travers un chemin tel que $/sfs/ < adresse >:< hachage(public\_key/emplacement) >$. [61, 14, 60] Le système à lui seul comprend des vulnérabilités de sécurité à son implémentation. [60]
XrootD	accès à haut débit (tolérant aux pannes et évolutif) à des *repositories* de données de toutes sortes pour la physique des hautes énergies.	L'utilisation typique est de donner accès à des *repositories* basés sur des fichiers. Le routage arborescent utilisé affiche de meilleures performances de recherche que les tables de hachage distribuées (DHT). [18]

TABLE 2.1 – Applications offrant un système de stockage de données ou de fichiers distribués.

[95] propose une comparaison entre plusieurs systèmes de fichiers distribués.

2.2.7 Couche Communication : Les réseaux pair-à-pair

Les systèmes pair-à-pair utilisent des approches de stockage de données qui s'appuient sur un réseau d'égal à égal [16]. Ces dernières comprennent des aspects de stockage, de réplication, de distribution et d'échange de données [29, 84]. Entre autres, les réseaux de données pair-à-pair aident à réduire les hypothèses de confiance et à s'affranchir des silos de données créés par les systèmes *cloud* centralisés [29].

Protocoles pair-à-pair

Les réseaux pair-à-pair (P2P) ont été architecturés comme un moyen de créer des réseaux résilients permettant aux pairs toujours connectés de continuer à communiquer malgré la déconnexion du réseau de certains d'entre eux. Les systèmes pair-à-pair sont un modèle des réseaux dont l'architecture entre les nœuds du réseau est comparable à des communications de type client-serveur pour les réseaux distribués.
Dans ces réseaux, les nœuds sont égaux (des pairs) et construisent un réseau superposé. Les utilisateurs finaux (les pairs) partagent les ressources par un échange direct entre ordinateurs et peuvent prendre des décisions locales autonomes. Chacun alterne donc entre les rôles de client et de serveur car l'information est distribuée entre les nœuds membres au lieu d'être concentrée sur un seul serveur. La charge de travail est partagée entre les ressources des pairs. Chacun peut partager, gérer et héberger conjointement du contenu complet ou partiel avec d'autres pairs. Lorsqu'un pair demande du contenu, tout pair qui possède des portions des données demandées peut participer à l'envoi de ce contenu. Après réception, le réseau de partage se renforce car le contenu récupéré peut être partagé par l'hôte de contenu supplémentaire. [29, 36, 22, 80, 18].

Les réseaux pair-à-pair peuvent soit avoir une architecture partiellement décentralisée, soit complètement décentralisée (réseaux pairs-à-pairs complets). Dans ce cas, on parle alors d'une architecture distribuée [16].

Les réseaux pair-à-pair sont utilisés dans de nombreux cadres applicatifs de logiciels qui utilisent généralement leur propre protocole de partage [95, 36, 7]. Le tableau 2.2 propose un recensement non-exhaustif de protocoles de transfert de données utilisés par les systèmes de stockage de fichiers distribués, ceci afin de recenser leurs caractéristiques principales, leurs avantages et leurs inconvénients.

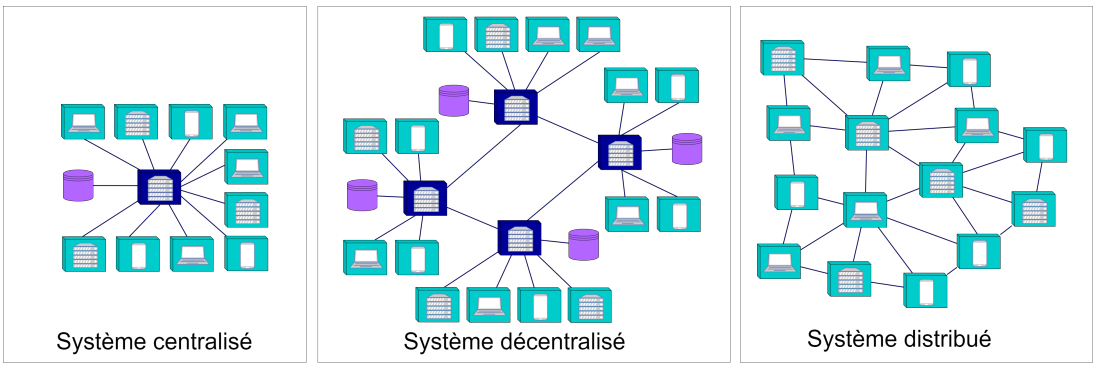


FIGURE 2.2 – Trois types d'architectures réseau. Les nœuds communiquent avec d'autres nœuds afin de former un réseau de communication. Les feuilles du réseau correspondent aux clients qui fournissent et/ou réceptionnent des données. Les nœuds font transiter des données et/ou gèrent leur stockage. Dans le cas d'une architecture distribuée, les pairs ont un rôle à la fois de nœuds et de clients.

Nom	**Objectifs**	**Caractéristiques**
BitTorrent	Transférer des fichiers volumineux de manière distribuée entre pairs [98].	Le protocole utilise les tables de hachage distribuées (DHT) [98] Kademlia [59, 14] pour suivre des pairs faisant partie d'un essaim de *torrents* [14]. Les pairs annoncent au réseau ce qu'ils ont en leur possession avec une consommation de bande passante négligeable avant de se connecter entre eux. Ils échangent des données dans les deux sens et utilisent toute la capacité de téléchargement disponible avec une fiabilité élevée. Plusieurs milliers de téléchargements sont effectués simultanément. Le système favorise les pairs pour lesquels le taux de téléchargement est élevé et qui ne téléchargent rien d'autre que le contenu en cours.[23, 98] **Avantages :** Robuste, plusieurs dizaines de millions de nœuds s'activent quotidiennement durant quelques heures sous la forme d'essaims [98, 23, 14]. Contrairement à un serveur informatique, lorsque le protocole pair-à-pair BitTorrent est utilisé par un client pour récupérer des données, celui-ci devient serveur à son tour et il partage également le contenu téléchargé précédemment. Cela renforce la capacité de partage du réseau.[78] **Inconvénients :** Un goulot d'étranglement se trouve au niveau de la surcharge de la bande passante des *trackers*.[78, 23] Le protocole ne couvre pas toutes les fonctionnalités que le web fournit avec HTTP [22]. Pas de stockage persistant à long terme garanti sur le réseau [29].

BitSwap	Echanger des blocs de données de manière distribuée entre pairs par adressage de contenu dans IPFS. Il parallélise les recherches dans les arborescences de Merkle et vérifie l'intégrité du contenu. [30]	Les nœuds pairs essayent d'obtenir un ensemble de blocs dont ils ont besoin sans savoir de quels fichiers ces blocs font partie. Ils fournissent en échange un autre ensemble de blocs en fonction du besoin des pairs. Les blocs peuvent provenir de fichiers complètement indépendants dans le système de fichiers. Cela s'inspire de BitTorrent. [30, 98, 14] **Avantages :** Améliore le système de routage de contenu de Kademlia DHT [59]. Il accélère la résolution du contenu et rend tout système de routage de contenu comparable à la vitesse du DNS. Le système ne se limite pas aux blocs de données d'un torrent. Il peut être utilisé avec une *blockchain* pour organiser une stratégie d'échange de données moyennant une récompense. [30, 14]
Gnutella	Permettre de rechercher et transférer des fichiers en pair-à-pair	Des pairs voisins sont connectés à un serveur et une méthode de *propagation par inondation* est utilisée pour obtenir les résultats des requêtes. [81] Utilise les tables de hachage distribuées (DHT) Kademlia. [59, 14] **Avantages :** Le réseau combine à la fois une distribution presque constante des nœuds, avec des caractéristiques d'une structure de *loi de puissance*. La connectivité des nœuds suit donc une distribution multimodale, rendant le réseau résistant aux attaques de personnes malveillantes, mais aussi aux défaillances aléatoires de nœuds. [81, 62] **Inconvénients :** Peu de précautions pour repousser les attaques, telles que le déni de services [81]. Dépendance à une consommation efficace du trafic réseau, car la consommation en données est très élevée pour connecter et diffuser les requêtes d'utilisateurs [81]. Pas de stockage persistant à long terme garanti sur le réseau [29].
HTTP (Hypertext Transfer Protocol)	Protocole de transfert hypertexte parmi les plus populaires et efficace dans de nombreux contextes du Web centralisé. [72, 22]	Communication de type client-serveur. Utilise l'adressage de localisation. Un unique serveur héberge de nombreux fichiers et l'information est récupérée en accédant à ce serveur. [80] Requiert de spécifier un emplacement sur le réseau pour l'adressage de localisation. Les coûts de téléchargement sont à la charge de la machine hôte du contenu. **Avantages :** Peu coûteux pour le transfert de fichiers de données de faible volume, même avec un trafic élevé. Gère difficilement les gros volumes de données. [23, 79, 72, 36, 80] **Inconvénients :** L'infrastructure du web n'est pas facilement modifiable en raison de sa grande échelle [22]. La sécurité et la présence de points de défaillances uniques sont des points faibles. [80]

PPSPP	protocole de diffusion de contenu en direct (streaming) sur un réseau pair-à-pair [72]	Adapté aux flux continus de vidéo et pour la demande continue de méta-données d'ensembles de données volumineux (*streaming*). Il tente de réduire le temps d'envoi et de réponse, ainsi que le temps de lecture en autorisant de nombreux types d'optimisations. **Avantages :** Son utilisation des arbres de Merkle permet de la transmission en direct afin de réduire le temps de lecture des données par les utilisateurs. [72]

TABLE 2.2 – Comparatif des protocoles de transfert de données

Architectures réseaux distribuées pair-à-pair

Les architectures réseaux distribuées pair-à-pair ont pour objet de rendre des ressources disponibles à long terme à travers un réseau par des échanges directs entre pairs, telles que du contenu, du stockage, de la puissance de calcul. [29, 33] Ces réseaux ont des algorithmes de localisation et de routage qui offrent la capacité de s'adapter et de s'auto-organiser sans devoir recourir à un serveur central comme intermédiaire (architecture client-serveur), et sans nécessiter la présence de tiers de confiance. [7, 14, 6]
Un système pair-à-pair pur est un système distribué sans contrôle centralisé, où le logiciel exécuté sur chaque nœud est équivalent en termes de fonctionnalité. Certains systèmes de fichiers prennent en compte ce type d'architecture et leurs fonctionnalités [36].

Dans la littérature scientifique, [36] présente certains points essentiels quant à la prise de décision de l'utilisation d'une architecture pair-à-pair pour un système de stockage de fichiers. [57] fournit un état de l'art des choix décisifs des styles architecturaux de systèmes de fichiers distribués.

Systèmes de stockage pair-à-pair purs

Les domaines des technologies de distribution de contenu et des systèmes de fichiers distribués on été affectés par la popularité et les progrès des technologies pair-à-pair [29]. Les systèmes de stockage décentralisés peuvent être partiellement, complètement décentralisés (distribués), voire être pair-à-pair purs. Ils peuvent être destinés au stockage et à l'organisation d'objets de données (données structurées), ou encore au stockage de fichiers de données (données non-structurées) à l'aide d'un espace de noms hiérarchique (*namespace*) et d'une interface permettant aux applications d'utiliser un stockage persistant à grande échelle. [36, 54, 21, 18]
Entre autres, [36, 57, 7] comparent des caractéristiques de plusieurs systèmes de stockage pair-à-pair purs.

Avantages des systèmes pair-à-pair purs :

- Tolérance aux pannes et aux défaillances réseau [36, 18] ;
- Garanties de disponibilité [7, 36] ;
- Evolutivité [7, 36, 26] ;
- Performance [7, 36, 26] ;
- Capacité d'auto-organisation en présence de nœuds transitoires [7, 36] ;
- Indépendance d'un serveur central [7] ;
- Equilibre efficace de la bande passante et des ressources [36].

Inconvénients des systèmes pair-à-pair purs :

- Des milliers de nœuds sur le réseau (serveurs) doivent communiquer pour que le réseau soit optimal [16] ;
- Risque de perte et d'indisponibilité des données par manque de nœuds disponibles sur le réseau, ou par des défaillances [95].

Plusieurs solutions sont possibles pour réduire les risques de perte et augmenter la disponibilité des données [16, 18, 29] :

- répliquer les données (les fichiers répartis sur plusieurs domaines) ;
- utiliser des codes d'effacement (*erasure codes*).

Combiner des systèmes de stockage de fichiers pair-à-pair purs avec des systèmes de *blockchains* peut aider à résoudre ces faiblesses. La combinaison avec les *blockchains* 2.2.9 offre des perspectives permettant d'augmenter la robustesse des systèmes distribués (intégrité, disponibilité, tolérance aux pannes byzantines) et les cryptoactifs incitent les utilisateurs à s'impliquer. [5, 33, 80, 29]

2.2.8 Performances des systèmes pair-à-pair et perspectives

[84] mesure le trafic des messages échangés sur un réseau pair-à-pair, à partir d'estimations analytiques et de la simulation informatique, sous la forme de deux approches probabilistes. Le protocole Gnutella est pris pour exemple afin de mesurer la charge engendrée par les messages sur les réseaux pair-à-pair distribués. Il ressort que le pair-à-pair change bien mieux de taille que ce que prédisent des théories se basant sur des hypothèses simplificatrices stipulant la création d'une structure arborescente exponentielle.

Trois mécanismes permettent de limiter ce problème de croissance :

- la taille finie de tout réseau.
- l'effet qui punit un utilisateur trop agressif en lui renvoyant un grand nombre de réponses sur sa machine.
- la probabilité de propagation des pings dans une structure arborescente diminue rapidement pour un nombre élevé de sauts dans un réseau fini.

[84] conclut que l'on peut s'attendre à ce que le pair-à-pair dans son état actuel ne remplacera pas Internet, mais qu'il pourrait concerner davantage d'utilisateurs et d'opérateurs réseaux qui apprendront à l'utiliser.

[11] compare de manière approfondie l'efficacité du trafic Web et les protocoles pair-à-pair BitTorrent et Gnutella.

Trois types de mesures sont prises au niveau de l'hôte [11] :

- **La concurrence des flux de données :** Les hôtes Web maintiennent un haut degré de simultanéité des flux de données, alors que de nombreux hôtes pair-à-pair ne maintiennent qu'un seul flux à la fois ;
- **La répartition géographique :**
  - **Web :** Hypothèse que les connexions sont variables en fonction du site web et probablement des habitudes culturelles de chaque pays (la langue, le type d'application accessible ou utilisées dans le pays, la notoriété, etc.). Les résultats ne peuvent pas être interprétés de manière fiable [11] ;
  - **Pair-à-pair :** La connectivité entre les hôtes pair-à-pair semble plutôt dépendre de la disponibilité des ressources pendant la phase d'établissement de la connexion plutôt que de la localisation des hôtes ;
  - **Gnutella :** Hypothèse que soit les pairs se connectent à des hôtes proches, soit le système est davantage utilisé dans certaines zones ;
  - **BitTorrent :** Les hôtes se connectent aux pairs à partir d'une liste de pairs fournie par les trackers. La liste des trackers serait créée en fonction de la disponibilité de la bande passante des hôtes dans un essaim et nous constatons donc un biais en faveur des régions à forte pénétration de la large bande [11]. La plupart des transferts réussis proviennent de pairs situés dans la même région géographique, même si les pairs se connectent à d'autres pairs distants pour obtenir du contenu ;
- **Le volume de transfert :** La symétrie des transferts de données est une préoccupation majeure des systèmes pair-à-pair en encourageant un partage équitable entre les pairs participants. Les transferts des hôtes Web et pair-à-pair sont asymétriques. De plus, BitTorrent utilise un mécanisme de *tit-for-tat* pour encourager le partage équitable entre les pairs en partageant des données pour avoir le droit de télécharger [23, 14]. Il y a donc une meilleure équité dans BitTorrent que dans Gnutella. Enfin, les gros consommateurs de Gnutella sont beaucoup moins importants en quantité d'octets échangés que ce que représentent les gros consommateurs de BitTorrent.

Trois types de mesures sont prises au niveau du flux de données [11] :

- **La taille :** Les applications P2P génèrent de nombreux flux de petite (le corps de la répartition) et de très grande taille (dans la queue de la répartition) par rapport au Web. La taille de flux pair-à-pair a donc une moyenne plus élevée et une taille de flux médiane plus faible que les flux Web. De plus, la taille des flux Gnutella est plus grande et plus dispersée que celle des flux BitTorrent. Enfin, la queue de la distribution de la taille des flux Web décroît plus rapidement que la distribution pair-à-pair correspondante ;
- **Les temps de latence :** les temps de latence des réseaux pair-à-pair sont beaucoup plus longs et plus dispersés que ceux des flux Web ;
- **La durée :**

— De manière générale, la probabilité de flux de longue durée est plus élevée pour le pair-à-pair que pour le Web ;
— Dans BitTorrent, il y a une proportion plus faible de transferts de très longue durée, ainsi qu'une corrélation positive entre la taille et la durée des flux. De plus, en raison de sa fonction de segmentation des fichiers, les transferts de fichiers importants n'y sont pas courants ;
— BitTorrent présente initialement un pourcentage plus élevé de flux de longue durée que Gnutella, et avec le temps la probabilité de la présence de flux de longue durée dans Gnutella est plus élevée que dans BitTorrent ;
— Les durées des flux BitTorrent diminuent plus rapidement que celles des flux Gnutella à la queue de la distribution.

L'outil Libp2p

Les systèmes pair-à-pair ne proposent pas de modules extractibles et réutilisables. Ces derniers sont généralement mis en oeuvre pour répondre à des environnements d'exécution spécifiques d'entreprises utilisant leurs propres protocoles pair-à-pair. Ils sont donc réalisés avec certaines caractéristiques qui les rendent difficiles à adapter à d'autres applications que leurs applications cibles initiales. L'outil Libp2p a été développé par la communauté open source du projet IPFS comme couche réseau générique. Son objectif est de fournir des modules logiciels qui comblent les faiblesses des protocoles réseau existants en proposant un système (fournissant un protocole) pair-à-pair modulaire inter-opérable destiné aux protocoles réseaux pour des cas d'applications non spécifique. Il a été inspiré à partir d'applications réseau existantes et de la recherche scientifique. [32, 29, 72, 42]
La solution se compose de [32, 29, 72, 42] :
— une licence open source ;
— un code source maintenu ;
— un code source libre ;
— solutions à des problèmes contemporains ;
— une documentation claire et exhaustive ;
— API conviviales supportant plusieurs langages de programmation ;
— implémentations peu spécifiques, voire génériques ;
— code évolutif pour les protocoles à venir ;
— une batterie de suites de tests.

2.2.9 La technologie *blockchain*

La technologie *blockchain* est un système réseau de registre distribué pair-à-pair qui conserve toutes les transactions effectuées sur le réseau *blockchain*. Elle permet de vérifier les données stockées avec une intégrité, une résilience, une crédibilité et une traçabilité élevées. Le réseau s'auto-organise pour changer de taille, supporter les défaillances et il n'a pas pour objectif de dépendre d'un serveur central. Elles sont adaptées aux systèmes de gestion de souscriptions, tels que la gestion d'identité et d'authentification, ainsi que des systèmes transactionnels financiers. [45, 7, 46, 39, 5]

La figure 2.3 représente les étapes de l'ajout de données de transactions dans une *blockchain*.
Un système de *blockchains* dépend de plusieurs éléments clés :
— **Type :** La *blockchain* peut être soit publique, de consortium, ou privée. Le concept de crypto-monnaie est commun à la *blockchain* publique [16, 6] ;
— **Réseau pair-à-pair :** Les *blockchains* s'exécutent de manière décentralisée. Les pairs contribuent au système de stockage et au traitement pour maintenir le réseau [49, 67] ;
— **Blocs :** Ils se composent chacun de données d'une transaction, du hachage cryptographique du bloc parent, d'un *nonce*, et de l'horodatage [49, 67, 80] ;
— **Transactions :** Regroupement en blocs d'un enregistrement d'événements vérifiés ordonnés et sécurisés cryptographiquement par une signature numérique ;
— **Système cryptographique :** La fonction de hachage inhérente aux systèmes *blockchains* relie les blocs dans un ordre chronologique de manière définitive (immuables).[33, 5]. Le hachage est une forme cryptée de la transaction d'origine à l'aide d'un algorithme de hachage cryptographique SHA (exemple : SHA1, SHA128, SHA256, SHA512) [26, 6] ;
— **Contrats intelligents (*Smart contracts*) :** Beaucoup d'applications décentralisées dépendent des contrats intelligents. Ils sont établis entre deux parties et certaines informations et les conditions essentielles de l'accord y sont écrites afin que les deux parties respectent leur accord. Les contrats sont signés numériquement. [26, 16, 6].

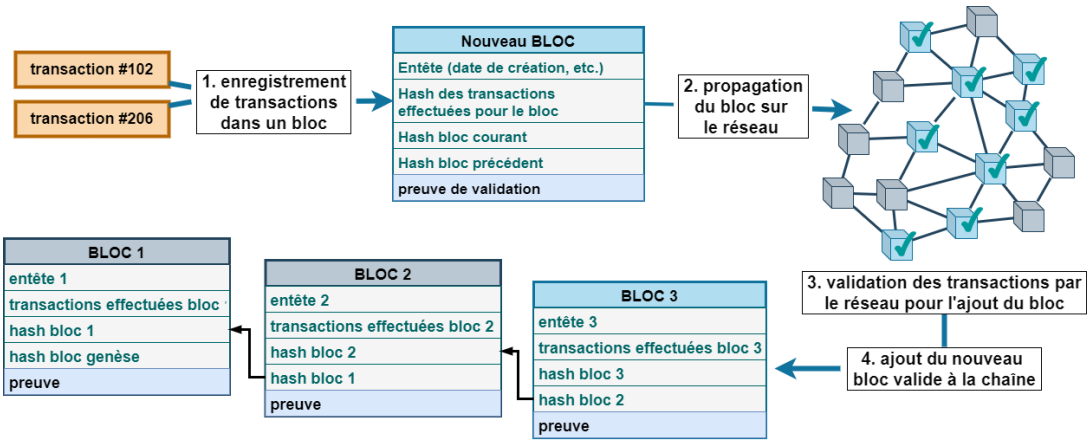


FIGURE 2.3 – Etapes de l'ajout d'un nouveau bloc dans une *blockchain*. (1) Des données de transactions sont enregistrées dans un nouveau bloc.(2) propagation du bloc sur le réseau. (3) validation des transactions par le réseau pour l'ajout du bloc. (4) ajout de nouveau bloc valide à la chaîne.

Dans le contexte des échanges réseaux, un contrat intelligent peut être mis en oeuvre et s'exécute en surcouche des réseaux de *blockchains* [26, 86]. Ils permettent d'éviter la présence d'intermédiaires en définissant et en appliquant automatiquement les règles et les obligations formulées par les pairs [46] ;

— **Protocoles de consensus :** Ils sont utilisés dans un réseau distribué d'une *blockchain* pour empêcher des nœuds malveillants de falsifier l'état des données, à l'aide d'un accord (un consensus) sur la garantie d'un état unique des données partagé dans tous les nœuds du système à un moment donné (une version, un état de la *blockchain*) [5, 26].

Types d'algorithmes de consensus

Plusieurs types de consensus sont possibles [5, 26], et ils se subdivisent en algorithmes de consensus [26, 15, 104, 97].

Par exemple, le protocole *Vote-based* organise une élection pour choisir un nœud (ou validateur) qui décide quel est le prochain bloc à ajouter au registre de la *blockchain*. Le protocole *Lottery-based* organise une élection sous la forme d'un tirage au sort afin de réduire la probabilité qu'un nœud malveillant valide un bloc falsifié [26, 67].
En l'occurrence, les algorithmes associés à ces deux mécanismes sont nombreux et diffèrent par une variation du modèle de confiance utilisé. Un exemple d'algorithme est le *Proof of Work* (PoW) [67], qui crée de nouveaux blocs en résolvant un casse-tête cryptographique complexe, mais facile à vérifier par d'autres pairs du réseau. Il garantit que le travail de calcul de hachage cryptographique a été fait et qu'un bloc ne peut pas être modifié sans refaire tout le calcul de recherche d'une collision de hachage dans le registre *blockchain*. Il existe beaucoup d'autres algorithmes, tels que *Proof of Elapsed Time* ; *Proof of Stake* (Ethereum) ; *Proof of Authority* ; *Proof of Deposit* ; *Proof of Burn* ; *Proof of Luck* ; *Proof of Spacetime* ; *Proof of Replication* ; *Proof of Space/Storage* ; *Proof of Capacity* ; etc. [49, 67, 26]

Une autre catégorie de protocole de consensus est le type *Byzantyne Fault-Tolerance* (BFT), où les systèmes qui l'utilisent fonctionnent malgré la présence d'une panne volontaire ou involontaire (nœud défaillant, nœud attaqué, nœud hors-service) de certains participants au protocole. Les algorithmes BFT diffèrent par l'utilisation d'un modèle de confiance distinct [20]. Par exemple, l'algorithme *Simplified Byzantine Fault Tolerance* (SBFT) est une version implémentée de l'algorithme *Practical Byzantine Fault Tolerant* (PBFT) [20]. Le système est tolérant aux pannes lorsqu'un consensus (un taux de tolérance) est atteint [26, 29].

Dans [97] des types de d'algorithmes sont comparés selon leurs caractéristiques, leurs taux de tolérance aux fautes, leurs temps de latence, le nombre de transactions par seconde, le temps de génération de blocs.

Types de *blockchains*

— **La *blockchain* publique/sans-autorisation :** Ce système de blockchain peut être déployé sans la présence d'une autorité, car les acteurs du système ne sont pas connus et tout le monde peut rejoindre ou quitter le réseau blockchain à tout moment. Cela peut augmenter les risques de sécurité dans le réseau [5]. Par exemple, Bitcoin [67] ou Ethereum [104]) sont deux sytèmes de *blockchains* publiques ;

— **La *blockchain* de consortium :** Groupe de nœuds sélectionnés (par exemple, la jonction de plusieurs organisations), dont les autorisations sont publiques ou restreintes [16] ;
— **La *blockchain* privée/autorisée :** L'utilisation de la *blockchain* privée peut permettre la gestion des autorisations et de l'authentification des utilisateurs qui interviennent sur un réseau [5, 39] pour des domaines non financiers, tel que le stockage *cloud* pair-à-pair. Seuls les participants identifiables authentifiés et autorisés sont admis dans le réseau blockchain. Cela réduit le risque de la présence d'acteurs malveillants au sein du réseau [76]. Le nombre de nœuds et de menaces de sécurité est moins élevés que pour les *blockchains* publiques [16]. Un exemple d'un système de *blockchain* privée est l'*Hyperledger Project* par la *Linux Foundation*. [5].

La figure 2.4 représente la différence de principes entre trois types de systèmes de registre : les systèmes de registres centralisés ; les systèmes de registres privés/de consortium ; les systèmes de registres publics.

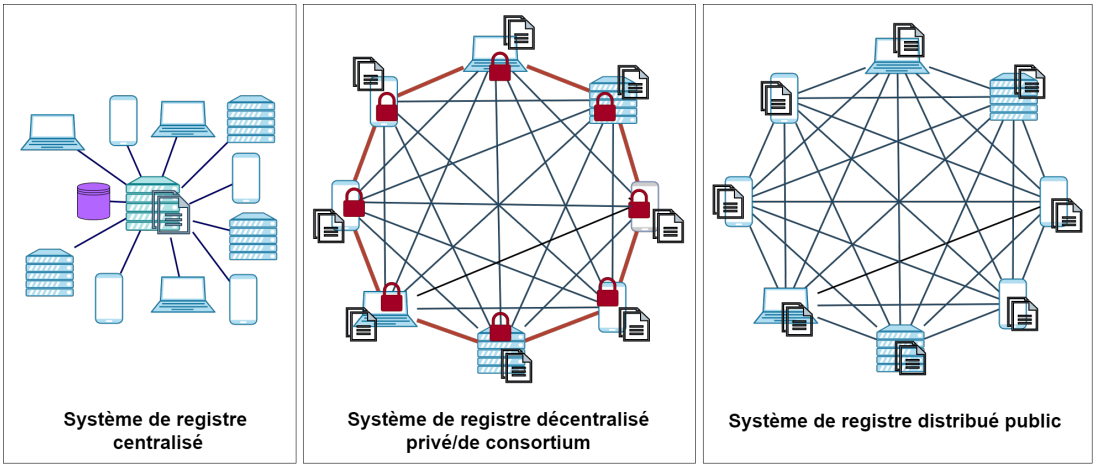


FIGURE 2.4 – Les systèmes de registres centralisés synchronisent les données des différents intervenants, où un tiers de confiance vérifie et entretient le système de registre ; les systèmes de registres privés/de consortium nécessitent un contrôle d'accès de l'identité des intervenants, nécessitant l'intervention d'une entité de contrôle centralisée afin de modifier le registre et répercuter des changements auprès des autres nœuds d'une architecture formant un réseau décentralisé ; les systèmes de registres publics ne requérant pas de contrôle d'accès. Chaque nœud du réseau conserve une copie du registre et le met à jour à chaque modification sur le réseau. Les modifications locales du registre sont communiquées à l'ensemble des nœuds du réseau, qui valident collectivement les modifications à l'aide d'un algorithme de consensus, puis qui les répercutent sur chaque nœud.

2.2.10 La *blockchain* dans les systèmes de stockage décentralisés

Malgré le fait que la *blockchain* s'est développée pour supporter des applications centralisées [86, 39], son utilisation dans les systèmes décentralisés permet de s'affranchir de tiers de confiance servant d'intermédiaires lors de transactions de communications. [16, 6]

Avantages des systèmes de stockage basés sur la *blockchain* :

— **Confidentialité :** Il n'est pas nécessaire de s'authentifier avec une identité réelle au sein du réseau afin de partager des données. Chacun peut cacher son identité à l'aide d'un identifiant *blockchain* et la gestion des comptes se fait de manière anonyme. L'identité d'une personne ne peut pas être révélée sur la *blockchain* à moins de donner sa clé privée et les transactions payantes sont effectuées à l'aide d'un portefeuille numérique utilisant les cryptoactifs [82, 67, 39, 26] ;
— **Système de réputation :** Les nœuds évaluent la confiance des nœuds envers un autre nœud à partir des interactions et transactions précédentes avec ce dernier. Cela permet de réduire les risques d'actions malveillantes dans les réseaux pair-à-pair [31, 28] ;
— **Sécurité des transactions :** Garantie d'authenticité du propriétaire légitime d'une transaction auprès de son destinataire avec l'utilisation d'une signature numérique [5, 39, 67] ;
— **Sécurité des données :** Les données doivent être chiffrées avant d'être envoyées sur la *blockchain*, et uniquement, les pairs ayant accès à la clé de déchiffrement peuvent accéder au contenu des données. Les données sont divisées en fragments, qui sont ensuite répartis sur le réseau vers les nœuds de stockage. Les données ne sont donc pas entièrement lisibles si un individu malveillant déchiffre un fragment. Les hôtes des données sont susceptibles de devoir prouver l'authenticité des données à l'aide des racines de Merkle [83]. Si elles ont été manipulées, comme expliqué plus tôt, il est possible de les récupérer à l'aide d'un *code d'effacement* [15, 6] ;
— **Disponibilité :** Chaque fragment est stocké à plusieurs endroits afin de garantir l'accessibilité aux données ;

- **Intégrité des transactions :** transactions immuables et traçables sur le réseau [33, 5] ;
- **Performances :** La répartition des données sur le réseau distribué augmente les performances [16] ;
- **Résistance aux pannes :** augmente en fonction de la quantité de points de répartition des données, et donc, des points de restauration [16], ainsi que par l'utilisation de fonctions de hachage cryptographique [5, 6] ;
- **Résilience :** Pas un point d'attaque ou de défaillance unique, car le système est décentralisé [5] ;
- **Persistance de données :** données immuables dans le temps [16] ;
- **Coût :** Plus faible que les systèmes centralisés. Les frais sont calculés en fonction de l'utilisation des services par l'utilisateur [102] ;
- **Matériel :** Choix du type de support physique utilisé [102] ;
- **Licences :** Open source [16] ;
- **Bande passante :** Les données sont récupérées en parallèle sur le réseau depuis plusieurs nœuds fournisseurs de stockage. La bande passante disponible augmente au maximum afin de réduire le plus possible le temps de téléchargement. Il est également possible d'augmenter la bande passante à l'aide de certains type de canaux de communication (la 5G par exemple) [45, 96].

Faiblesses des systèmes de stockage basés sur la *blockchain* :

- **Sécurité :** Les objets de données doivent être chiffrés et déchiffrés à chaque envoi sur les réseaux distribués. Ils ne peuvent alors pas être lus par toute personne tentant de les consulter. Malgré cela, les systèmes *blockchains* ne sont pas complètement sécurisés, mais le risque est bien inférieur à celui des réseaux centralisés. [37, 39, 82, 26] discutent de certaines des attaques auxquelles ils sont sujets, ainsi que les applications qui y sont liées par effet boule de neige ;
- Pas de support pour le stockage *cloud* au format d'applications type client-serveur permettant de faire de l'analyse ou du traitement de données [16]. [76] aide à analyser si l'application de la *blockchain* est pertinente en fonction du cas d'utilisation du système distribué ;
- **Evolutivité :** Difficile [16], car il faut assurer la rapidité et la sécurité de la *blockchain* à mesure de son nombre de nœuds croissant, ainsi que du nombre de transactions par blocs [45]. Il peut y avoir des problèmes de débit [47] et de capacité causant des latences [39] ;
- **Risque de ballonnements :** la *blockchain* ne doit pas être considérée comme une base de données, même si l'enregistrement des transactions précédentes est toujours accessible dans la *blockchain* et qu'une énorme quantité de données sont répliquées sur chaque nœud. Seules des méta-données doivent y être enregistrées et les données structurées et non-structurées doivent être stockées hors-chaîne [45, 16, 80] ;
- **Stockage des données :** un manque de mémoire de stockage peut survenir lorsque le nombre de références augmente pour le stockage dans le système des grands réseaux décentralisés [6] ;
- **Coût du stockage de données :** Elevé s'il est fait directement sur la *blockchain* car elle n'y est pas adaptée. Il convient donc de l'utiliser comme registre uniquement [46]. De plus, si la *blockchain* a une forte croissance, il peut y avoir des frais plus élevés que la transaction elle-même [6, 39] ;
- **Pas de cadre juridique clair** en cas de fraude, d'escroquerie ou d'irrespect des contrats intelligents [16, 79] ;
- **Contrôle d'accès :** les réseaux de stockage basés sur la *blockchain* ne permettent pas le partage de données entre les utilisateurs. Une solution basée sur les contrats intelligents peut aider à gérer ce problème [90] ;
- **Logique basée sur les données :** Les données des systèmes de stockage basés sur la *blockchain* sont chiffrées et partitionnées avant d'être stockées par un pair fournisseur de stockage. Elles ne peuvent pas être traitées ou analysées comme sur les systèmes de stockage centralisés pour orienter des décisions. Afin de résoudre ce problème, [77] propose une solution utilisant les contrats intelligents dans lequel les entreprises créent leur système de stockage basé sur la *blockchain* privée, et où des agents certifiés ont les clés et les autorisations pour extraire et analyser les données ;
- **Durée des transactions :** Le processus de validation d'une transaction en fonction du nombre de blocs concernés peut être long, car chaque transaction approuvée nécessite une vérification de différentes fonctions en pair-à-pair, telles que : l'algorithme de consensus ; l'ajout d'un nouveau bloc ; l'ajout d'informations de la transaction au bloc concerné ; la distribution de la réplique à tous les utilisateurs ; l'attente que d'autres utilisateurs la valident. [6]

2.2.11 Systèmes de stockage basés sur la *blockchain*

Les systèmes de stockage qui se basent sur la *blockchain* permettent de stocker des données hors-chaîne. Elles ont la capacité de conserver les avantages des systèmes décentralisés, tout en surmontant les problèmes d'évolutivité et de confidentialité des *blockchains* [47]. Les informations d'horodatage, ainsi que les références

des données non-structurées (fichiers réels) sont conservées dans la *blockchain*. Les objets de données sont, quant-à-eux, enregistrés dans un système hors-chaîne décentralisé pair-à-pair. [79, 15, 92, 80] Le tableau 2.3 décrit plusieurs systèmes similaires. Ils ont des éléments d'intérêt et certaines exigences qui engendrent des différences de décisions de conception pour la recherche, la décentralisation, la redondance, la confidentialité des contenus, ainsi que l'organisation du réseau [29].

Nom	**Description**	**Caractéristiques**
BigchainDB [17]	Base de données pair-à-pair évolutive orientée *Big Data* et basée sur la *blockchain*. Elle permet de déployer des plate-formes et des applications *blockchain*.	Basé sur la base de données MongoDB et supporte les requêtes MongoDB. Les données stockées sont immuables. Tolérance au pannes byzantines. Peut être déployée sur des réseaux publics ou privés. Faible latence. Supporte tout type de cryptoactif ou jeton de transaction. Développement Open source.
Dat (Hypercore Protocol) [72]	Protocole d'application qui se concentre sur la synchronisation et le partage de fichiers volumineux ou d'une taille variant sans cesse. Les données sont continuellement chiffrées à l'aide d'une paire de clés publique et privée [79, 72]	Il utilise son propre protocole de stockage avec un chiffrement de bout en bout. [79, 72] Il utilise l'adressage de contenu. Il permet à un groupe de clients de se connecter pour former un réseau public ou privé décentralisé afin d'échanger des données. Il utilise un registre de modifications versionnées pour prouver que la version des données demandée est distribuée. Les pairs peuvent choisir de répliquer ponctuellement le contenu partiel ou complet provenant d'un référentiel Dat distant, ainsi que de synchroniser les changements en continu [72].
Filecoin [15]	Plate-forme *open source* en surcouche du réseau IPFS basée sur le stockage décentralisé. Elle incite les utilisateurs à partager leur espace de stockage inutilisé pour héberger ou épingler des objets non structurés IPFS. [79, 15] Tous les périphériques informatiques sont connectés en un réseau pair-à-pair formant un système de fichiers distribué. Il évite un point de défaillance unique sur le réseau [16]. Les contrats intelligents offrent la possibilité aux utilisateurs de rédiger des programmes à états utilisant des jetons. Ils conditionnent le stockage ou la récupération de données et ils valident des preuves de stockage. [15]	Les fournisseurs de stockage proposent de l'espace de stockage disponible et des clients y stockent des données sans requérir l'intervention d'un tiers de confiance. Une *preuve de l'espace-temps* ou une *preuve de réplication* [15] doit assurer aux clients que leurs données sont en sécurité dans le temps, comme c'est le cas sur les autres réseaux décentralisés. [16] Le réseau devient robuste en partageant des répliques des données aux pairs par l'utilisation de l'IPFS [79]. Le réseau détecte et en répare automatiquement les objets répliqués défaillants [15]. Le jeton utilisé est le FileCoin (FIL). Son minage est proportionnel au stockage actif afin d'inciter les mineurs à accumuler du stockage et à le louer aux clients. Le contenu est chiffré de bout en bout. La plate-forme supporte les contrats intelligents spécifiques au stockage de données (par exemple, les contrats de fichiers : contrat de mineurs, stratégies de paiement, services de billetterie, création de contrats qui permettent la mise à jour des données) et les contrats intelligents plus génériques, où les utilisateurs peuvent associer des programmes à leurs transactions qui ne dépendent pas directement de l'utilisation du stockage [15]. Les contrats intelligents prennent en charge les opérations propres à Filecoin, telles que la vérification des preuves et les opérations de marché. [79, 15, 16]

Sia [96]	plate-forme open source [79] proposant un système de stockage *cloud* distribué tout public utilisant la *blockchain*. Les pairs ne se font pas confiance. Les fournisseurs (hôtes) proposent leur capacité de stockage disponible aux pairs (clients) sur le réseau à l'aide de contrats de stockage entre pairs stockés sur la *blockchain*. Les données doivent être chiffrées et signées numériquement avant d'être mises sur le réseau. [96]	Chaque fichier est lié à un contrat, et chaque contrat est lié à une racine de Merkle [63, 96] pour chacun des fichiers utilisés qui fournit des preuves par les hôtes. Le client et l'hôte sont liés par un contrat spécifiant la durée de stockage, la récompense pour chaque preuve de travail valide ou invalide, ainsi que le nombre de preuves invalides maximum. [16] Les preuves de stockage sont vérifiables et disponibles sous la forme d'une liste publique de hachages du fichier, ainsi qu'une fraction du fichier original. La *blockchain* de Sia est open source. Le cryptoactif utilisé est le SiaCoin (SC). [79, 96] Utilise des codes d'effacement dans la stratégie de stockage afin de garantir la haute disponibilité des données [29].
Storj 3.0 [92]	plate-forme open source [79] proposant un système de stockage distribué tout public utilisant la *blockchain* [92] et destiné aux grands ensembles de données [29]. Elle permet à un hôte (ou *farmer*) de partager son espace de stockage matériel inutilisé en toute confiance avec un client par l'utilisation de contrats intelligents Ethereum [16]. Ethereum permet aux utilisateurs de récupérer leurs informations dans leur intégralité autant de fois que désiré [26]. Les métadonnées des clients sont chiffrées avant d'être transmises sur le réseau. Le réseau est résistant à l'indisponibilité (résilience). Cela est possible avec réplication automatique des nœuds lorsqu'un nœud est indisponible. [101, 92]	— Un contrat intelligent entre le client et le *farmer* stipule un montant en StorjCoin (SJCX) pour le stockage des données [79, 92, 16] ; — Pour éviter le *ballonnement*, seules des métadonnées de chaque bloc de données sont stockées sur la *blockchain*, avec comme informations l'emplacement et le hachage [16] ; — Les fichiers sont stockés par un minimum de trois hôtes, sinon davantage afin d'augmenter la répartition des points d'accès [101] ; — Les fichiers envoyés sont partitionnés à l'aide d'un processus de partitionnement. Un seul nœud n'a pas accès à la copie complète des données envoyées sur le réseau lors de son envoi, et les fragments sont rassemblés lors de la récupération des données. Ils ne sont pas lisibles séparément ; — Une preuve de redondance permet de garantir que les nœuds demandés sont toujours disponibles dans la quantité spécifiée [101, 92] ; — La plate-forme utilise l'application autonome open source *Metadisk* [102] pour s'assurer régulièrement que les données sont intègres et disponibles sur le réseau [101]. Cela est fait par preuve de stockage (Proof of Space/Storage, ou PoS), sinon, par preuve de capacité (Proof of Capacity, ou PoC) [16] ; — Compatibilité avec le *cloud* public (tel que AWS) [16, 92] ; — A pour objectif de supporter d'autres cryptoactifs pour les échanges [92] ; — Chaque opérateur de nœud de stockage doit répondre à certaines exigences (exemple : bande passante) afin de rejoindre le réseau [92] ; — Utilise des codes d'effacement dans la stratégie de stockage afin de garantir la haute disponibilité des données [29].

Swarm [73]	plate-forme basée sur l'écosystème Ethereum pour fournir des infrastructures dont les pairs partagent le stockage et la bande passante [79, 16]. L'ensemble du système offre des services *cloud* (*streaming* de données, base de données, des canaux de paiement) [73].	— Fournit un stockage décentralisé pour le code et les données d'application Ethereum décentralisées (DApps) [79] ; — Il utilise son propre protocole de stockage [79] ; — S'appuie sur le réseau pair-à-pair Ethereum directement pour communiquer à l'aide de son sous-protocole ; le protocole DevP2P basé sur LibP2P 2.2.8 établit des sessions entre les nœuds et échange de messages ; RLPx (*Recursive Length Prex*) découvre des nœuds et transmet des données de manière sécurisée [42] ; — une chaine de contrats intelligents est configurée pour maintenir les opérations de base [42] ; — Le système se base sur le principe de *liquid democracy* [73] ; — **Swarm Core (couche 1) :** fournit la couche réseau et les contrats intelligents nécessaires à la création de nouveaux projets sur la plate-forme [73] ; — **Swarm Services (couche 2) :** facilite la création d'applications d'investissements par les participants, ainsi que la fourniture de services sur la *marketplace* [73] ; — **Swarm Apps (couche 3) :** Applications frontend et backend indépendantes destinées à des utilisations précises de la plate-forme [73].
Threefold [74]	Exploite un cloud pair-à-pair autonome et une technologie de blockchain publique (TFChain) afin de fournir une infrastructure web distribuée. Cette dernière propose un partage de ressources de calcul dans des conteneurs hautement optimisés, ainsi que des ressources de stockage et des ressources réseau pour l'exécution de services.	Un système d'exploitation open source (Zero-OS) permet aux pairs (textitFarmers) de mettre des serveurs à disposition d'autres pairs pour former un cloud distribué, en l'échange d'une récompense. Le jeton utilisé est le TFT. Tout type de matériel de type serveur AMD ou Intel peut être enregistré dans la blockchain pour fournir de la capacité de stockage ou/et de la puissance de calcul au réseau. Des contrats intelligents permettent à chaque utilisateur de gérer son infrastructure cloud distribuée au niveau du système d'exploitation. Utilise un algorithme de *Proof-of-Capacity* vérifiant que les fermes de stockage garantissent la capacité de stockage en continu sur le réseau, ainsi qu'un *Proof-of-Usage* pour vérifier l'utilisation de la capacité des données. Les fichiers stockés sont divisés en fragments et répartis sur le réseau. Intègre des technologies de conteneurisation en périphérie du réseau afin de permettre des architectures de micro-services avec des *frameworks* d'automatisation pour le calcul distribué.

TABLE 2.3 – Systèmes de stockage distribués basés sur la *blockchain*

[29] fournit une comparaison de certains de ces systèmes de stockage, les compare avec d'autres systèmes de stockage similaires et propose également une vue d'ensemble des relations entre ces systèmes.

2.3 En direction d'un Web distribué avec l'IPFS

Le web distribué (Web 3.0) privilégie la souveraineté individuelle des utilisateurs, plutôt que de se baser sur la confiance que fournissent les systèmes tiers [9, 42]. Comme expliqué plus tôt (voir 2.2.4à-), le web distribué repose sur une décentralisation complète des systèmes du web centralisé et repose en conséquence sur les systèmes pair-à-pair purs (voir 2.2).
En l'occurrence, le web distribué tend à combiner des systèmes tels que des infrastructures de mise en réseau pair-à-pair, des systèmes de stockage de données ou de fichiers distribués, des protocoles de messagerie cryptographiques, des mécanismes crypto-économiques, des systèmes de publication de données tel qu'IPFS, des technologies *blockchains* [103]. La combinaison d'un ensemble de ces types de systèmes peut fournir une solution Web distribuée sans recourir à des tiers de confiance, ceci afin d'assurer l'intégrité et la disponibilité des systèmes, ainsi qu'une sécurité et une confidentialité des données.

L'IPFS peut aider à réaliser ce type d'infrastructures distribuées pour le stockage et la distribution de données. Les modèles des systèmes de stockage de fichiers du Web et d'IPFS diffèrent principalement par leurs paradigmes d'adressage de localisation et d'adressage de contenu pour stocker les données. Le modèle d'adressage de contenu par blocs à haut débit (*content-adressed block storage model*) fourni par l'IPFS utilise un système de liens hypertextes uniques, générés à partir d'un calcul de hachage cryptographique [14, 83]. Les résultats de ce calcul sont enregistrés dans un Graphe Acyclique Dirigé de Merkle (ou Merkle DAG) [83], un système ayant inspiré Dat [72], Git, ou encore Camlistore pour optimiser le contrôle de versions de données [14, 32]. Le protocole intègre alors les arbres de Merkle dans un système de stockage de fichiers partagé afin de construire un Web permanent qui propose un système de fichiers versionnés, avec la possibilité d'intégrer des *blockchains* [83, 79].

Le système IPFS peut être combiné à des *blockchains* pour le web 3.0 en utilisant des plateformes de stockage [79], ou encore en les utilisant nativement afin de stocker les fichiers réels hors-chaîne tout en conservant uniquement les références du hachage de ces fichiers dans la *blockchain* [6, 14, 98, 79].

2.3.1 Enjeux du protocole pair-à-pair de l'IPFS

IPFS a pour objectif de fournir un système de publication de fichiers global distribué pair-à-pair (Web permanent), à faible latence, à l'aide de son propre protocole de communication [79, 14]. Il doit permettre de stocker du contenu numérique distribué sur un réseau pair-à-pair [46] ; versionner les fichiers disponibles sur le réseau afin d'éviter la duplication de données ; connecter de nombreux nœuds informatiques avec le même système de fichiers et les gérer en suivant leurs versions au fil du temps ; fournir une possibilité d'intégrer des *blockchains* et différentes plate-formes de stockage [79]. Il se distingue d'autres protocoles de communication pair-à-pair par la conjonction de plusieurs caractéristiques inhérentes, telles que :

- Une communication décentralisée pair-à-pair. Les fichiers sont distribués sur le réseau ;
- L'utilisation de l'adressage de contenu pour répondre aux inconvénients de l'adressage de localisation ; chaque fichier est stocké par blocs et adressé par son hachage cryptographique en fonction de son contenu ;
- L'utilisation de LibP2P pour gérer la couche réseau [42] ;
- La combinaison d'une table de hachage distribuée (DHT) avec un espace de noms auto-certifiant et un échange incitatif de blocs, l'ensemble de la DHT est distribuée à différents endroits du réseau ;
- Une capacité de combinaison aux *blockchains* afin de stocker les fichiers réels hors-chaîne tout en conservant uniquement les références du hachage de ces fichiers dans la *blockchain*. [6, 14, 98, 79]

Avec l'utilisation de Libp2p 2.2.8, l'IPFS peut transférer des données avec tous les protocoles réseau. Il en change librement en utilisant une combinaison d'adresses et de protocoles correspondants (technique formatée multiaddr) [14]. Lorsque le réseau sous-jacent n'est pas stable, IPFS peut choisir UTP ou SCTP [42].

2.3.2 Adressage de contenu dans l'InterPlanetary File System

IPFS propose une solution d'adressage de contenu vérifiable entre pairs qui est partagé sur le web décentralisé afin de répondre aux inconvénients de l'adressage de localisation [80]. Cette forme d'adressage de contenu utilisée est appelée *Content Identifier* (CID). C'est un identifiant unique universel généré à l'aide d'un algorithme de hachage cryptographique. Il exprime des structures de données éligibles à l'adressage de contenu pour le web distribué, créant alors des réseaux de données entrelacées qui référencent les données du réseau de manière sécurisée et vérifiable [83, 29, 14].

Ce CID correspond au contenu des objets de données spécifiées sous la forme de hachage cryptographique unique. Il garantit l'authenticité du contenu indépendamment de son emplacement [79]. Il peut être considéré à la fois comme un nom et comme un lien et il contient un codec contenant les informations sur la manière d'interpréter les données, ainsi qu'un *multihash* qui est un hachage auto-descriptif [80]. Les codecs encodent et

décodent les données dans certains formats, tandis que le *multihash* indique quel type de fonction de hachage a été utilisé pour créer ce *hash* à l'aide de ses méta-données. Les CID multiformats (*multihash*) sont à l'épreuve du temps car ils utilisent le multihash pour prendre en charge plusieurs algorithmes de hachage. [29, 14]

Un CID supporte pratiquement n'importe quel algorithme puissant de hachage cryptographique (SHA2-256 utilisé par défaut).

Il n'est pas requis de spécifier un emplacement sur le réseau pour partager des données de toutes formes sur le réseau, contrairement à l'adressage de localisation. En revanche, il convient de spécifier le CID correspondant aux objets de données que l'on désire récupérer sur le réseau. Il est alors nécessaire de connaître le format de données et utiliser l'outil approprié pour qu'un hachage de contenu soit fonctionnel. [29, 79, 14]

La figure 2.5 représente la comparaison des principes de l'adressage de localisation avec l'adressage de contenu utilisé avec IPFS.

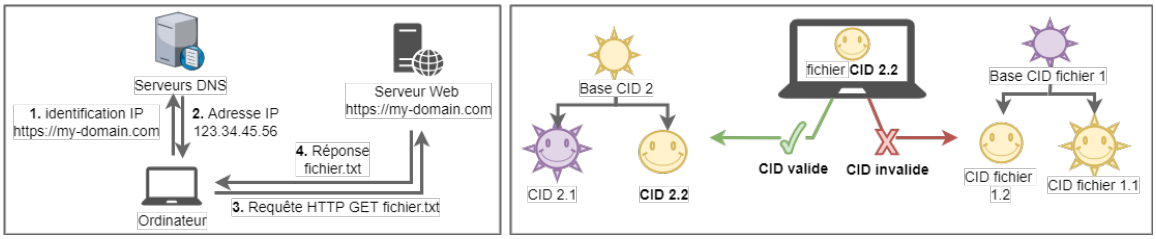


FIGURE 2.5 – Comparaison des principes de l'adressage de localisation utilisant le protocole HTTP avec l'adressage de contenu basé sur les arbres de Merkle de l'IPFS

Défis de mise en place

L'IPFS tente de se confronter à plusieurs difficultés de mise en place de l'adressage de contenu qui peuvent apparaître dans le contexte des systèmes de communication pair-à-pair [22, 80, 57] :

- **Censure :** Bloquer l'accès à du contenu dangereux ou illégal est utile, mais cela est problématique en cas de rétention de l'information liée à un régime politique autoritaire. Cela est courant dans de nombreux pays ;
- **Confidentialité :** Le entreprises ne respectant pas la vie privée des utilisateurs de leurs services sont de plus en plus courantes ;
- **Environnement d'exécution :** Il existe de plus en plus d'environnements d'exécution, comme l'IoT (l'Internet des objets). Dans ce contexte les ressources matérielles permettant de les construire sont limitées et les protocoles utilisés se basent sur des hypothèses d'environnements d'exécution [66, 80] ;
- **Fiabilité :** De nombreux utilisateurs de réseaux sont peu robustes : partout dans le monde, ils sont confrontés à de faibles débits d'échange des données qui découlent d'une connectivité instable ;
- **Innovation :** Le temps peut être long entre le développement de nouveaux protocoles et leur déploiement ;
- **Itinérance (adressage mobile) :** Les appareils des utilisateurs doivent être garantis comme étant identifiables de manière unique lorsqu'ils naviguent sur les réseaux internationaux ;
- ***Network Address Translation* (NAT) :** Le fonctionnement des routeurs Wi-fi domestiques qui traduisent une adresse IP locale d'une machine en une adresse IP unique et auquels des réseaux externes peuvent se connecter ;
- **Pare-feu** Un pare-feu peut restreindre ou bloquer des connexions sur un réseau ;
- **Latence élevée des réseaux :** Les temps de réponse des réseaux à latence élevée peuvent paraître longs pour leurs utilisateurs.

2.3.3 Solutions proposées par l'InterPlanetary File System (IPFS)

Le système de stockage fourni peut amener vers un web pair-à-pair pur, proposant plusieurs fonctionnalités principales :

- une publication d'informations n'imposant pas l'hébergement à l'éditeur, mais à ceux qui le désirent [14] ;
- une sûreté du contenu téléchargé, ne requérant pas de faire confiance aux hôtes [14] ;
- créer un web permanent pour conserver des données importantes [79, 14] ;
- l'hébergement et le partage de flux multimédias de grand volumes (haute définition) en temps réel [14] ;
- l'hébergement et la distribution de pétaoctets de données en toute confiance [14] ;
- un calcul distribué pour des données d'organisations [14] ;
- un système de stockage hors-chaîne pour les *blockchains* en les couplant avec des contrats intelligents [29].

Forces de l'IPFS

Le système est caractérisé par une salve d'avantages qui ont été relevés lors de son utilisation. Ces caractéristiques sont les suivantes :

- un système de distribution avec un espace de noms global, avec une gestion de versions et de liaison d'ensembles de données massifs [14] ;
- une plate-forme permettant l'écriture et le déploiement sur le réseau [6, 14] ;
- un système de partage de fichiers proposant un paradigme d'adressage de contenu, concurrent à l'adressage de localisation [14]. Ajouté à l'immuabilité des données, le système est plus résistant à la corruption des fichiers par un pair malveillant [57] ;
- une réduction considérable du coût de bande-passage utilisée avec l'adressage de contenu [80] ;
- une réduction considérable du coût du stockage des données [88] par un stockage efficace du contenu numérique [46] ;
- la possibilité de créer des architectures de réseaux distribuées en utilisant la *blockchain*, combinée à un stockage hors-chaîne de données chiffrées des utilisateurs dans l'IPFS. Les CID sont gardés comme références du contenu dans la *blockchain* afin d'y enregistrer uniquement des références vers des objets de données [14, 79] ;
- la possibilité de mettre à jour et versionner les données privées, et de ne pas surcharger les systèmes de *blockchain* en y enregistrant des données volumineuses engendrant des coûts élevés. [79, 14, 88] ;
- la garantie d'une haute disponibilité des données [88] fournissant un stockage du contenu permanent à l'aide d'un système d'épinglage [79] ;
- un modèle de distribution réduisant les risques de censure du contenu. Les informations stockées restent disponibles indépendamment de suppressions par censure ou d'attaques par point de défaillance unique. [79, 57] ;
- un déplacement dynamique et précis des données. Les fichiers de données sont accessibles à proximité de leur lieu d'utilisation. Les plus populaires sont davantage répliqués. Cela renforce la résilience et les performances du réseau. [88, 57] ;
- une garantie de l'absence d'un point de défaillance unique lorsqu'il est utilisé de manière distribuée [14, 46] ;
- une non nécessité de faire confiance aux tiers [6, 14].
- L'amélioration des performances du téléchargement des données [88]. Le nombre de nœuds épinglant un fichier spécifique renforcent le réseau en permettant de l'obtenir facilement et rapidement [79] ;
- la possibilité de créer des réseaux IPFS privés depuis des *clusters* IPFS coordonnant la connexion entre des pairs partageant une clé secrète [79].

Faiblesses de l'IPFS

Le système comporte certaines faiblesses qui ont été relevées lors de son utilisation. Ses caractéristiques sont les suivantes :

- Globalement, les systèmes de stockage distribués basés sur IPFS proposent un mécanisme de sauvegarde centralisé à l'aide de nœuds officiels qui garantissent la haute disponibilité des données. Les pairs ont alors un point de défaillance unique, et cela contrarie l'objectif initial d'IPFS en créant un risque de diminution des performances du système à la place de distribuer les données sur de nombreux nœuds. Ces nœuds officiels ont une capacité de stockage limitée et ont des risques de faibles performances d'accès en parallèle [88, 16, 14] ;
- Les nœuds du réseau IPFS sont libres d'être connectés ou déconnectés du réseau à tout moment. Les données envoyées sur un nœud qui se déconnecte du réseau deviennent donc inaccessibles si elles n'ont pas été sauvegardées par d'autres nœuds [88, 16, 14] ;
- Lorsqu'un objet de données non épinglé (uniquement gardé en cache dans le système IPFS) est récupéré par le *garbage collector* du système IPFS, il est supprimé d'un nœud. Rien ne garantit qu'il ait également été supprimé de tous les nœuds qui avaient précédemment accédé à ce fichier et l'avaient donc mis en cache [79] ;
- Le système IPFS actuel permet de résoudre une caractéristique à double tranchant qu'est l'immuabilité des données sur un réseau des systèmes de *blockchains* en stockant les données hors-chaîne. Il ne permet cependant pas de répercuter la suppression de données sur tous les nœuds du réseau lorsque cela est demandé, conformément au règlement européen du droit à l'oubli. Car tant qu'un nœud garde en ligne un objet de données épinglé, ce dernier reste disponible en permanence pour les autres pairs du réseau [79] ;
- Pas d'incitation de garantie du stockage par les pairs dans le système original. Pour cela, il est nécessaire d'utiliser une plate-forme telle que Filecoin 2.2.11 pourvu d'un système de récompense [15, 79, 14] ;
- IPFS ne tolère pas les *attaques byzantines*. Chaque pair peut accéder à chaque fichier stocké sur l'IPFS

tant qu'il se connecte au système. Une solution est d'utiliser des mécanismes de contrôle d'accès basés sur des contrats intelligents et les technologies de chiffrement pour les données stockées basés sur la *blockchain* [29] ;
— La technologie *blockchain* est sûre pour conserver les données de transaction, mais le risque d'attaques de sécurité persiste malgré son utilisation avec l'IPFS en tant que système de stockage hors-chaîne [43].

2.4 Fonctionnement de l'InterPlanetary File System

L'IPFS propose un système de fichiers global (Web permanent) à faible latence [79, 14] qui permet le stockage et l'échange du contenu numérique sous la forme de données structurées et non structurées distribuées sur un réseau pair-à-pair [46]. Cela comprend la présence d'un système de gestion de version intégré au réseau qui permet d'éviter la duplication de données.
IPFS est constitué d'un réseau de nœuds informatiques interconnectés qui peuvent chacun héberger des blocs de données correspondant aux contenus qui composent les fichiers de données. Ils sont reliés au système de fichiers afin de pouvoir suivre leurs versions au fil du temps.

Il est composé de sous-systèmes co-dépendants qui intègrent et exploitent des propriétés. Cela forme une pile de sous-protocoles responsables de certaines fonctionnalités disposés en couches.
La figure 2.6 représente la pile protocolaire de l'IPFS. Les sept couches interagissent entre elles à l'aide d'un ensemble de trois technologies principales. IPNS pour l'attribution des noms ; IPLD pour la gestion des fichiers et objets de données dans le protocole ; Libp2p comme protocole pour les couches d'échange, de routage et réseau. [27]

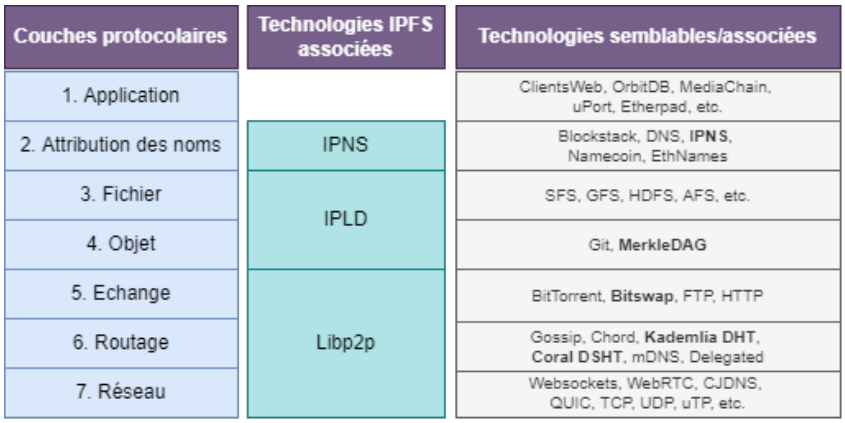


FIGURE 2.6 – Pile protocolaire de l'InterPlanetary File System. (1) La couche application tend à utiliser le système IPFS à travers des applications. Des systèmes tels que OrbitDB [1], ou les Clients web utilisent IPFS. D'autres systèmes sont concurrents dans le domaine du web 3.0 ; (2) la couche d'attribution des noms est IPNS. Elle concurrence notamment le système DNS du web centralisé ; les couches (3) et (4) utilisent IPLD afin de gérer les fichiers (à l'aide d'un système semblable à SFS) et les objets de données (à l'aide d'un système semblable à Git) à travers IPLD qui utilise les Merkle DAG 2.4.5 ; les couches (4), (5) et (6) utilisent Libp2p pour l'échange des blocs de données à l'aide de Bitswap, le routage à l'aide de Kademlia DHT et de Coral DSHT et la gestion du réseau en supportant une multitude de protocoles différents.

Le protocole utilise une table de hachage Kademlia DHT [59] pour stocker les données dans une structure et identifier quel nœud du réseau héberge des données recherchées par des pairs [29, 79]. Afin de rechercher le contenu, le système utilise le système d'attribution des noms IPNS proposant un espace de noms auto-certifiant, qui utilise l'adressage de contenu pour répondre aux inconvénients de l'adressage de localisation. Un hachage cryptographique unique (hachage de contenu) est calculé sur base du contenu des données sans se soucier de leur taille. Chaque fichier est stocké par blocs et adressé par son hachage cryptographique en fonction de son contenu. Les données sont fractionnées en fragments lorsqu'elles sont prêtes à être ajoutées sur le réseau et chaque fragment est identifié par son propre hachage, puis distribué aux différents nœuds du réseau dont le hachage est le plus proche de celui de l'utilisateur (*peer ID*) [6, 57].
En l'occurrence, l'espace de noms fourni par IPNS est combiné à un système d'échange incitatif de blocs et à une table de hachage distribuée (DHT) à différents endroits du réseau. Avec l'utilisation d'une DHT, le réseau IPFS stocke le hachage des fichiers et les pairs peuvent l'utiliser pour récupérer les données par adressage de contenu. [14, 15, 22, 80]. Ainsi, lorsqu'un pair introduit une demande de récupération à l'aide d'un hachage de contenu (CID), l'IPFS utilise la DHT afin de traverser le réseau de nœuds où le hachage cryptographique est présent. C'est la bibliothèque LibP2P qui gère les couches d'échange des blocs de données, de routage et réseau [42]. Après avoir parcouru tous ces nœuds, les fragments sont recombinés en l'objet de données initial

et le pair récupère ce contenu. [14, 15, 6]
Par ailleurs, l'IPFS dispose d'une application cliente permettant de visualiser le trafic en cours sur le réseau de nœuds [6].

Le diagramme 2.7 représente le flux pour l'ajout et la récupérations de fichiers sur l'IPFS.

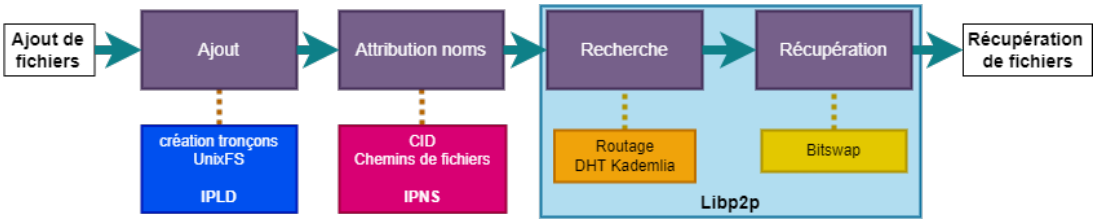


FIGURE 2.7 – Etapes d'ajout de fichiers

2.4.1 Identification des nœuds

Un identifiant (Peer ID) est créé sous la forme d'un hachage cryptographique représentant une clé publique pour chaque nœud du réseau à l'aide d'un casse-tête cryptographique statique fourni par S/Kademlia. Ceci permet d'identifier quel nœud possède quelles données sur le réseau. Lors de la première connexion entre deux nœuds pairs, ils échangent leurs clés publiques et ils vérifient si l'identifiant du pair est égal au résultat du hachage de sa clé publique. La connexion est interrompue en cas d'échec. Les nœuds stockent leurs clés publiques et privées chiffrées par un mot de passe.
Un nœud est destiné à ne pas changer, mais les utilisateurs peuvent recréer une identité de nœud à chaque démarrage. Ils perdent cependant les avantages de l'accumulation des données au sein du nœud en faisant cela. [14, 6]

2.4.2 Réseau

Le réseau IPFS est configurable. En l'occurrence, il gère les connexions entre pairs et il utilise des protocoles réseau sous-jacents afin que de nombreux nœuds communiquent entre eux, potentiellement à travers le web étendu par l'IPFS. De plus, il stocke les adresses sous la forme de chaînes d'octets (format multiaddr) pour le réseau sous-jacent à utiliser. Ce format exprime les adresses, leurs protocoles et la prise en charge de l'encapsulation. [14, 42]

L'adressage des pairs peut être utilisé dans les *réseaux superposés* à l'aide de Coral [34], car il ne s'appuie pas sur l'adressage de localisation IP mais sur l'adressage de contenu, et n'en suppose pas l'accès [57, 14]. Coral est un concept de tables de hachage distribuées sécurisées (DSHT) introduites par le réseau pair-à-pair de distribution de contenu CoralCDN. CoralCDN permet à ses utilisateurs d'exploiter un site web en redistribuant des données. Les DSHTs Coral sont destinées à indexer du contenu sous la forme de paires de clés et de plusieurs valeurs afin de hiérarchiser le contenu. Cela permet de mettre en cache le contenu destiné au stockage distribué pour le web. Les paires de clé/valeurs permettent à Coral de regrouper des noeuds d'un réseau en fonction de sa taille, de localiser et de récupérer des répliques de données proximales sans devoir interroger des noeuds distaux. [34]

2.4.3 Routage

Le système de routage IPFS permet de conserver les informations de localisations des pairs et des objets de données, ainsi que de répondre aux requêtes. Il identifie les adresses réseau des pairs, ainsi que les pairs qui peuvent servir des objets spécifiques. Cela permet donc de retrouver quel nœud possède des données recherchées. [66, 14, 6] L'utilisation de tables de hachage distribuées sécurisées (DSHT) basées sur Coral [34] et S/Kademlia [13] permet d'obtenir :

- une optimisation du nombre de messages de contrôle qu'il envoie à d'autres nœuds à travers DHT Kademlia [59] ;
- une résistance aux attaques en favorisant les nœuds à longue durée de vie à travers DHT Kademlia [59] ;
- un moyen de recherche efficace pour les requêtes sur des nœuds à travers DHT Kademlia [59, 14] ;
- une surcouche *Coral DSHT* [34] à DHT Kademlia, qui résout le stockage des blocs de données directement dans les nœuds pour éviter une utilisation de la bande passante et une perte de stockage inutiles.

Les données $\leq$ 1KB sont stockées sur la DHT (voir figure 2.10). Pour les objets plus volumineux, les références des pairs (identifiants de nœuds) pouvant servir le bloc sont stockées [6, 14] ;
— une surcouche de sécurité *S/Kademlia* [13] à DHT Kademlia pour la protéger d'attaques malveillantes. [6].

2.4.4 Echanges de blocs

[14] introduit la création du protocole d'échange de blocs de données BitSwap [30]. Les blocs de données sont distribués en fonction de l'offre et de la demande sur le réseau, tout en visant à éviter la réplication des données. Un principe de solde et de dettes (en octets) entre pairs permet d'envoyer du contenu en supposant de façon optimiste que des données seront renvoyées en échange. Une méthode probabiliste permet de diminuer le taux d'échange de blocs de données à mesure qu'une dette augmente.

La durée de vie d'une connexion entre pairs se divise en plusieurs étapes [22, 14] :

1. **ouverture de connexion :** envoi de registres jusqu'à acceptation par les pairs ;
2. **envoi des données :** listes d'objets désirés (*want list*) et des blocs de données sont échangés [30] ;
3. **fermeture de connexion :** la connexion entre pairs est désactivée ;
4. **ignorer un pair :** Les pairs peuvent être tentés d'augmenter leur taux d'échange en déclenchant plus de demandes. Une stratégie de nœud permet d'éviter l'envoi de données à un pair pour l'ignorer durant un certain temps (10 secondes par défaut).

2.4.5 Objets de donnnées

L'IPFS construit un graphe acyclique dirigé (*Merkle DAG*) d'objets immuables identifiés par des adresses de contenu (CID). Avec l'utilisation du protocole BitSwap et de la DHT, cela permet de former un système pair-à-pair massif qui stocke et de distribuer des blocs de données. Les liens entre les objets de données échangés sont des hachages cryptographiques des cibles référencées dans les sources de données. [88, 79, 83]. Les objets de données IPFS sont alors accessibles sur le réseau à l'aide de chemins d'accès qui utilisent les CID pour accéder aux arbres de Merkle.
La figure 2.8 représente la structure entre les blocs de données et les Merkle DAG de l'IPFS.

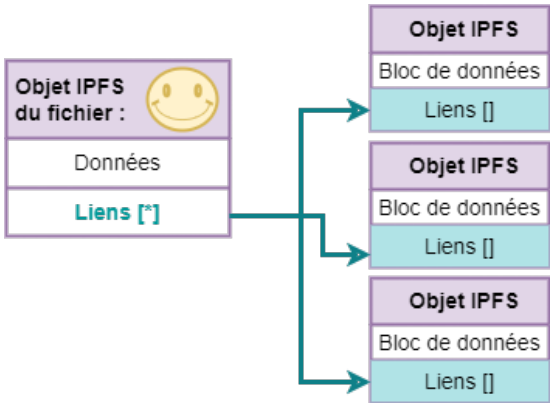


FIGURE 2.8 – Organisation d'objets de données d'un fichier par l'utilisation des arbres de Merkle

Les graphes acycliques dirigés de Merkle fournissent plusieurs propriétés importantes à l'IPFS [83, 8] :
— Adressage de contenu : le contenu est identifié par un hachage cryptographique unique, qui est utilisable sous la forme d'une adresse d'accès au contenu ;
— Unicité des identifiants : les objets qui contiennent exactement le même contenu sont égaux et référencent le même objet stocké sur le réseau ;
— Résistance à la falsification : Tout contenu corrompu ou falsifié est détecté grâce au recalcul du hachage cryptographique.

Un système d'épinglage permet également de conserver des objets sur le système de fichiers et de s'assurer qu'ils ne seront pas supprimés par le garbage collector de l'IPFS. Les objets sont stockés localement sur le système de fichiers d'un nœud. Si le contenu est mis en cache uniquement, l'objet de données est stocké dans la mémoire RAM ; s'il est stocké sur le système de fichiers ou qu'il est épinglé, l'objet est stocké sur disque. [22, 14] Par ailleurs, la DHT permet de publier des objets sur le réseau de façon distribuée en y insérant des clés d'objets. Tout objet étant immuable et versionné, l'ajout de nouveaux objets au réseau ajoute une nouvelle clé dans la DHT.

Enfin, IPFS intègre une gestion d'opérations cryptographiques pour chiffrer, pour vérifier automatiquement les signatures, ainsi que de déchiffrer les objets avec des ensembles de clés spécifiées par l'utilisateur. Un objet peut ainsi être créé en tant qu'une version de l'objet chiffré, ou bien il est possible de vérifier les octets bruts de sa signature. Les opérations cryptographiques présentes dans un DAG de Merkle modifient le hachage de l'objet. Un objet est différent lorsqu'il est chiffré que lorsqu'il n'est pas chiffré.
De plus, les CID d'objets chiffrés sont protégés afin d'empêcher de les parcourir sans clé de déchiffrement. Pour sécuriser les CID vers des objets partagés, un objet parent peut être chiffré sous une autre clé qu'un enfant. [29, 14, 83, 80]

Graphes Acycliques Dirigés de Merkle

La confiance pour les structures de données accessibles dans un environnement de stockage isolé ou local (depuis un disque ou en mémoire) peut être accrue. Cela n'est pas le cas sur le web décentralisé, car les pairs récupèrent les données directement sur le réseau plutôt que d'une autorité centrale [6, 57]. La confiance entre les pairs est presque nulle, car rien ne garantit que les données récupérées sont les données demandées lors de leur récupération. Par exemple, pour une liste chaînée utilisée comme objet de données pour parcourir le web distribué et indiquer la position de chaque objet du réseau, un acteur malveillant pourrait tenter d'insérer des données où il le souhaite sans que l'on le sache. [22, 57, 83, 14]

Les objets de données peuvent être utilisés pour relier des données de tous types entre elles, tout en préservant la capacité des *Content Identifiers* (CID) à vérifier leur intégrité. Elles peuvent contenir une donnée dans son intégralité (tel qu'un fichier complet), ou bien une portion de cette donnée (tel qu'un fichier qui se décompose en plusieurs nœuds). [83, 29]

La déclinaison des arbres de Merkle [64] dans IPFS, les *Graphes Acycliques Dirigés de Merkle* (ou *Merkle DAG*) [83] sont des structure de données arborescentes générées de manière ascendante. Chaque nœud parent possédant du contenu identifie un à plusieurs nœuds enfants à l'aide d'un CID unique dépendant de ses descendants. Les CID sont dérivés d'un algorithme de hachage cryptographique et permettent aux arbres de garantir l'intégrité et la vérifiabilité des données. [29, 83]

Etant donné que le CID de chaque nœud dépend de chacun de ses descendants, si le contenu d'un nœud enfant change, le CID (donc le hachage cryptographique) associé à ce enfant change à son tour. Ce changement d'étiquette se propage à chacun de ses ancêtres, se traduisant par la présence de nouveaux CIDs.

Tout ajout ou toute modification dans un arbre de Merkle se fait de bas en haut car les nœuds parents ne peuvent pas être créés tant que les CID de leurs enfants restent indéterminés. Une modification apportée dans une branche de l'arbre ne modifie pas obligatoirement des CID de nœuds appartenant à des branches soeurs, car le CID d'un nœud ne change qu'en réponse à un changement dans ses propres données ou celles de ses descendants.
Les fonctions cryptographiques utilisées dans la construction des CID de l'arbre de Merkle rendent un chemin auto-référencé impossible à travers le graphe. Il est donc impossible de retrouver une boucle infinie (des cycles) entre les CID référençant les structures des nœuds de l'arbre de Merkle.
Il n'est pas nécessaire de récupérer du contenu structuré tel qu'un arbre dans son contexte intégral lors du partage de donnée. Récupérer un sous-graphe (identifié par le CID de son nœud parent) est suffisant. Il est possible d'intégrer ce sous-graphe en tant que sous-graphe d'un tout autre arbre de Merkle, car le CID d'un graphe (le CID du nœud racine) dépend des descendants du nœud racine, et non de ses ancêtres. De la même manière, intégrer un arbre de Merkle dans plusieurs arbres simultanément permet d'obtenir un tissage interconnecté de graphes.
Enfin, il n'est pas nécessaire que les arbres de Merkle aient strictement un seul nœud racine. Des nœuds parents peuvent soit avoir un nœud racine en commun, soit un nœud a plusieurs parents (voir figure 2.9). On ne peut pas naviguer vers tous les nœuds de cet ensemble de données à partir d'une seule des deux racines. Cela peut donc générer deux arbres de Merkle distincts. [29, 83, 22, 8, 14]

Vérifiabilité et finalités des arbres de Merkle

Une adresse de contenu peut permettre de calculer le CID des données récupérées et de vérifier que les données récupérées sont identiques avec un contenu désiré. La vérifiabilité garantit que les données liées à une adresse de contenu sont invariables (permanence des données dans l'arbre) et qu'elles sont protégées contre des manipulations malveillantes. De cette manière, utiliser les arbres de Merkle permet de calculer si deux répertoires possédant chacun un même fichier avec des versions différentes sont identiques. [83, 14, 64, 8]

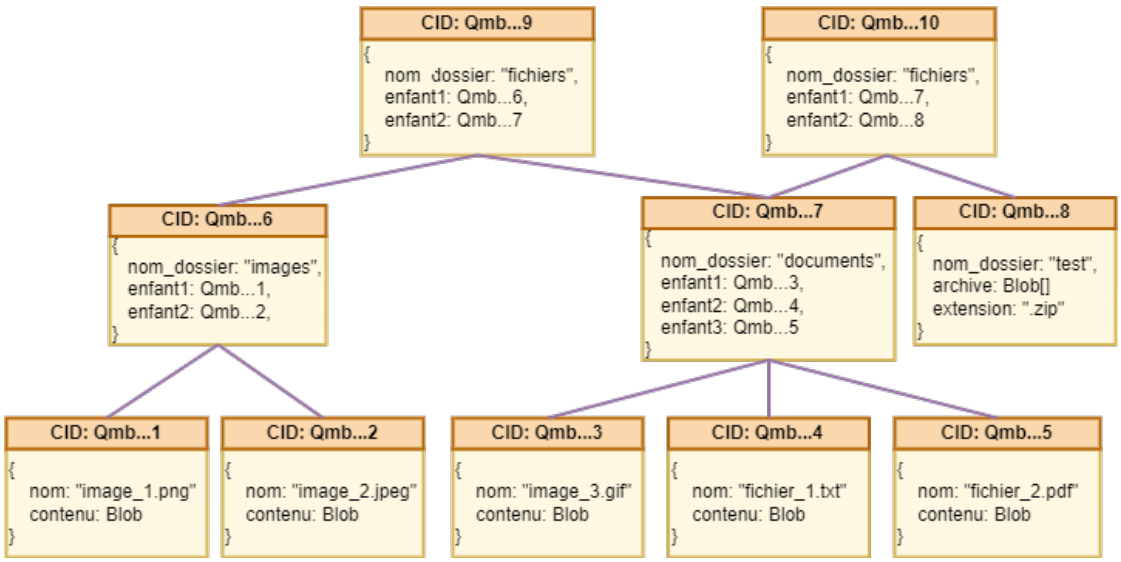


FIGURE 2.9 – Exemple d'un graphe acyclique dirigé de Merkle

Utilisations des arbres de Merkle

Certains systèmes intègrent les arbres de Merkle afin de pouvoir assurer une traçabilité de la gestion des versions des données.

- Dat [72] utilise les arbres de Merkle pour gérer ses registres de modifications ;
- Git utilise les arbres de Merkle pour suivre les différences de versions entre les modifications apportées au code source dans les projets logiciels [32, 14, 72] ;
- IPFS les utilise sous une forme étendue : les Graphes Acycliques Dirigés de Merkle (ou *Merkle DAG*). Ils ne requièrent pas d'équilibrages au niveau des branches et ils ont des données au niveau des nœuds [8, 14, 79] ;
- PPSPP réduit le temps de lecture pour les utilisateurs finaux en utilisant un arbre de Merkle permettant à des sous-ensembles de hachages d'être demandés par un pair [72].

Comparatif de l'utilisation des arbres de Merkle pour le stockage avec le web centralisé	
Web distribué avec les arbres de Merkle	**Web centralisé**
Tout le monde peut distribuer un document.	Un serveur doit être maintenu pour rendre un document disponible.
Des nœuds se trouvant n'importe où peuvent participer à diffuser des données	Un à plusieurs serveurs sont dédiés à partager le même contenu dans le monde.
Chaque nœud de l'arbre de Merkle a son propre CID qui peut être distribué indépendamment des autres.	Les données peuvent être distribuées de façon monolithique en tant qu'archive d'un fichier unique.
Il est aisé de trouver plusieurs fournisseurs de mêmes données.	Il est difficile de trouver plusieurs fournisseurs de mêmes données.
Les nœuds formant le DAG sont petits et peuvent être téléchargés en parallèle à partir de nombreux fournisseurs différents.	Les données sont probablement en gros morceaux qui doivent être téléchargés en série à partir d'un seul fournisseur.
Les ensembles de données plus volumineux englobant l'original lient l'ensemble de données d'origine en tant qu'enfant d'un plus grand DAG.	Il est difficile pour les autres de partager des ensembles de données qui s'appuient sur les données d'origine.

2.4.6 Fichiers

L'IPFS définit un ensemble d'objets utilisés pour définir un système de fichiers versionnés en surcouche du DAG de Merkle. Quatre types d'objets sont utilisés par le système de fichiers :

- *blob* : bloc de données de taille variable qui représente un fichier et contient une unité de données adressable.

— *list* (liste) : représente une collection de blocs ou de listes. Il est utilisé pour représenter un fichier volumineux ou répliqué et composé de plusieurs objets blob listés.
— tree (arborescence) : représente une collection d'objets de type *list* ou de type *tree*. Il forme une arborescence utilisée pour représenter un répertoire.
— *commit* : représente un instantané d'une version d'arborescence dans l'historique de tout objet. Il aide au contrôle des versions d'un objet pour comparer deux versions d'objets du système de fichiers.

Les objets de fichiers peuvent être parcourus dans les DAG de Merkle à l'aide d'une API qui communique avec le système de fichiers. Les objets de données échangés sur le réseau sont des objets binaires similaires au format des objets de fichiers Git. Ils sont importés et exportés au format JSON à l'aide du protocole de sérialisation Protobuf (ou *Protocol Buffers*). [32, 14]

Ajout de fichiers sur l'IPFS

Le diagramme 2.10 représente le flux pour l'ajout d'un fichier sur l'IPFS.

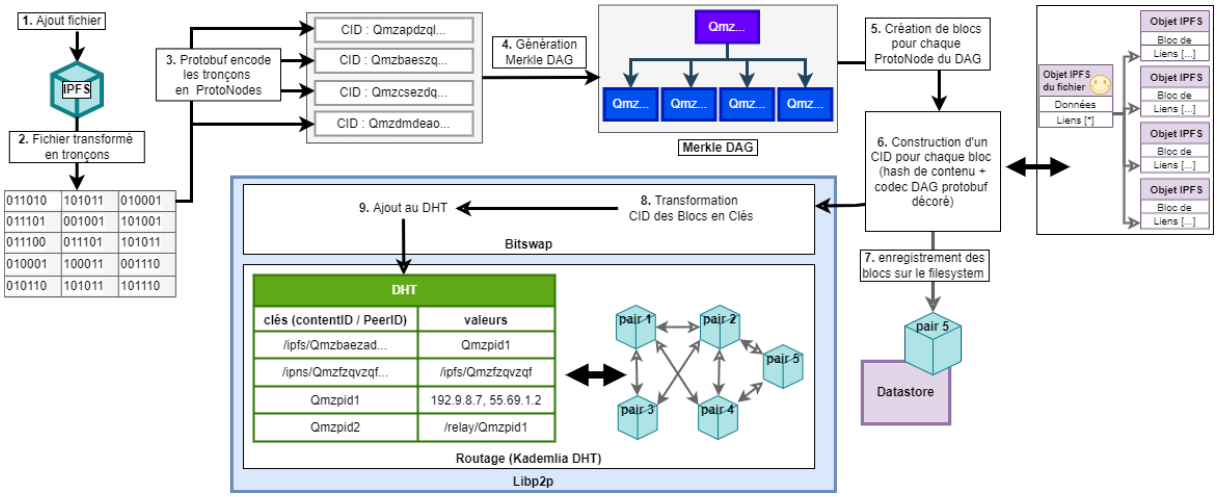


FIGURE 2.10 – Flux d'ajout d'un fichier dans le système de fichiers IPFS. (1) envoi du fichier par un pair sur l'IPFS ; (2) le fichier segmenté en tronçons ; (3) Protobuf encode les tronçons en *Proto Nodes* ; (4) les noeuds obtenus sont insérés dans un arbre de Merkle ; (5) les blocs correspondant à chaque Proto Node du DAG de Merkle sont créés ; (6) Un CID est créé pour chaque bloc de données afin de les relier entre eux ; (7) Les blocs sont enregistrés sur le *filesystem* du nœud désiré ; (8) Les CID sont transformés en clés afin d'être ajoutés au DHT (9).

Récupération de contenu sur l'IPFS

Le diagramme de séquence 2.11 et la figure a figure 2.12 représentent les étapes principales de la récupération de contenu depuis un pair IPFS en lui fournissant un CID.

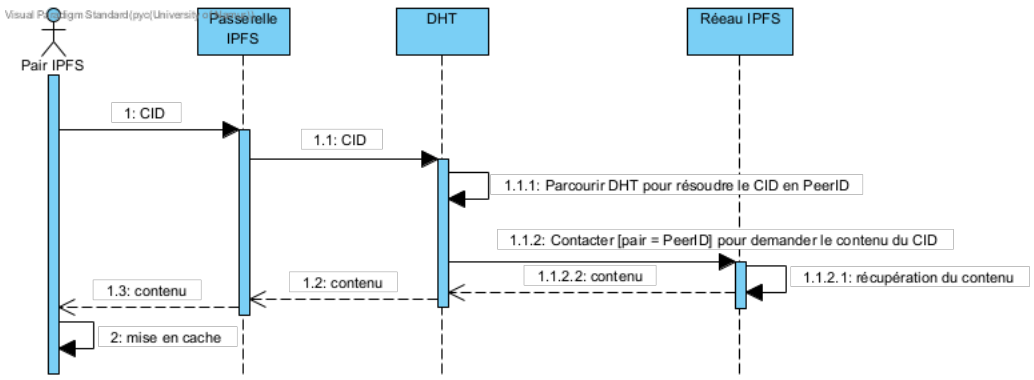


FIGURE 2.11 – Diagramme de séquence pour la récupération de contenu depuis le sytème de fichiers du réseau IPFS par un pair IPFS à l'aide d'un CID.

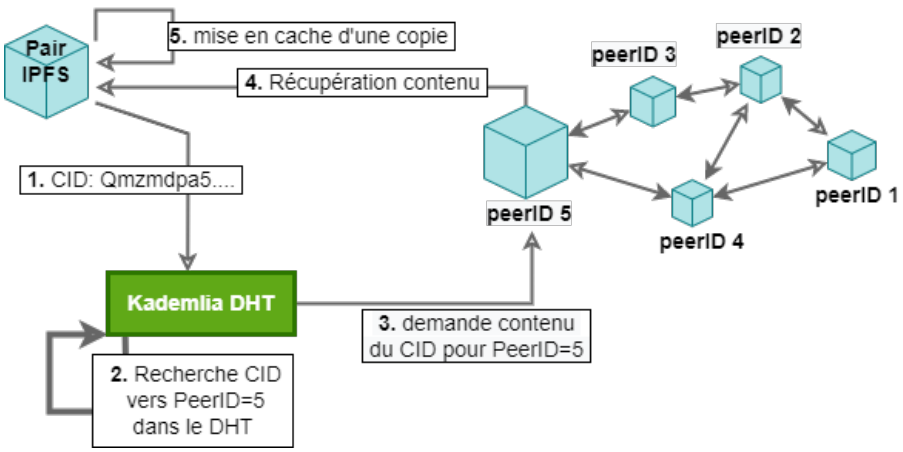


FIGURE 2.12 – Exemple d'un flux simplifié pour la récupération de contenu depuis le sytème de fichiers du réseau IPFS par un pair IPFS à l'aide d'un CID.

2.4.7 Système d'attribution des noms IPNS (*IntePlanetary Naming System*)

Le système d'adressage de contenu d'IPFS est le système auto-certifiant de noms mutables IPNS (*Interplanetary Naming System*). Il crée une adresse de contenu mutable préfixée par $/ipns/ < suffixe >$. L'adresse permet de ne pas devoir partager un nouveau CID à chaque modification d'un objet de données.
Le suffixe est calculé depuis le contenu des objets de données partagées, qui correspond au hachage cryptographique d'une clé publique. Cette clé est liée à un enregistrement contenant des informations sur le hachage auquel il est lié et signé par la clé privée correspondante. De nouveaux enregistrements peuvent être signés et publiés à tout moment. [32, 14, 80]

2.5 Cas d'utilisation d'IPFS

La littérature scientifique comprend de nombreux cas concrets d'application d'IPFS liés au contexte des systèmes de stockage décentralisés. Entre autres, [14] propose plusieurs cas d'utilisation du protocole. Voici quelques cas concrets intéressants étudiés dans la littérature scientifique :

2.5.1 Application de la *blockchain* avec l'IPFS

Bien que l'IPFS ne soit pas dédié aux systèmes *blockchains*, dans de nombreux cas, il se concentre sur la recherche de données afin de remplacer les architectures client-serveur du web par des applications distribuées (DApps) [29]. De nombreux cas d'application proposent de combiner les *blockchains* avec l'IPFS, tels que [5, 68, 46, 107, 68, 33, 6, 22]. Un recensement non-exhaustif de systèmes combinant l'IPFS avec les *blockchains* est fait dans la section 2.5.

Plate-forme de stockage Filecoin

Comme expliqué dans la section 2.2.11, la plate-forme Filecoin intègre le protocole IPFS. Elle propose un système de stockage pair-à-pair pur, basé sur la *blockchain* afin de garantir du contenu sur plusieurs nœuds du réseau en même temps. Elle fournit un système de stockage de fichiers distribué sécurisé en surcouche de l'IPFS.

Blockchains publiques

[33] propose d'utiliser l'IPFS comme système de stockage et de combiner les *blockchains* publiques Ethereum avec les oracles Chainlink [19] afin que les contrats intelligents aient connaissance des données stockées sur l'IPFS. Ces derniers prennent des décisions en fonction des résultats récupérés depuis le réseau.

[70] propose de stocker les codes de création de contrats intelligents Ethereum dans un stockage hors-chaîne IPFS, faisant office de base de données afin d'améliorer les performances de stockage des systèmes de fichiers distribués et diminuer la bande passante du trafic réseau. Ethereum charge les codes de contrats complets en envoyant une valeur de hachage aux pairs IPFS. Les pairs échangent ces valeurs de hachage lors des synchronisations rapides à la place des codes complets.

Blockchains* privées : Framework pour PingER *(Ping End-to-end Reporting)

PingER est un outil de mesure de performances d'internet de bout en bout. Il possède une architecture client-serveur [58] qui se compose de 50 agents de surveillance (*Monitoring Agent*) présents dans 20 pays. Ils sondent plus de 700 sites distants dans plus de 160 pays afin d'en extraire des statistiques (méta-données) qui sont finalement stockées localement sur un agent de surveillance toutes les 30 secondes. Ces agents ne contribuent ni au stockage, ni à l'analyse, ni au traitement, ainsi qu'au reporting des données [5]. Le *Stanford Linear Accelerator Center* (SLAC), créateur et mainteneur du projet, récupère quotidiennement et archive les données dans un système centralisé. Ce système dépend des ressources informatiques du SLAC allouées à la fois au traitement, à l'espace d'archivage, et à la disponibilité [58].

[5] présente un *framework* qui se base sur la *blockchain* privée, afin d'assurer l'accès (disponibilité) et le stockage des données pour *PingER*, en cas d'arrêt de la maintenance du projet par le SLAC. Une table de hachage distribuée (DHT) est utilisée pour permettre de supprimer intégralement l'utilisation d'un référentiel centralisé. Le résultat est un système de stockage hors-chaîne décentralisé, des capacités de recherches efficaces pour PingER et un traitement distribué des données.

Le framework comprend trois couches :

- **Couche de communication :** Le réseau centralisé est transformé en un réseau pair-à-pair d'agents de surveillance. Ils fournissent des installations de stockage et de communication à la DHT. Chacun aide à la collecte des données, est un lieu de stockage et facilite les services Web.
- **Couche de stockage de données :** La *blockchain* est utilisée pour gérer l'identité et le contrôle d'accès sur le réseau en stockant une faible quantité d'informations de méta-données des fichiers de données PingER.
  Les fichiers de données réelles PingER sont stockés hors-chaîne sur des espaces de stockage du réseau pair-à-pair PingER et les références sont stockées dans une table de hachage distribuée (DHT). Chaque fichier est haché avec une fonction cryptographique avant de transmettre et de stocker des fichiers journaliers sur des espaces de stockage sur le réseau. Les hachages de tous les fichiers sont combinés pour créer un arbre de Merkle.

  Caractéristiques :
  - **Transaction :** Chaque agent de surveillance diffuse la transaction en cours sur les réseaux et vérifie la signature de la transaction. Les transactions contiennent le hachage du fichier, une racine de Merkle, l'emplacement du fichier, la signature de l'agent, la taille de fichier, l'horodatage.
  - **Bloc :** Les méta-données des fichiers de chaque agent sont stockées quotidiennement sur la *blockchain* privée et les transactions validées sont regroupées en blocs. Un bloc possède un entête (horodatage, index du bloc dans la *blockchain*, le hachage de la racine de Merkle créé à l'aide des hachages de toutes les transactions d'un bloc et du hachage du bloc précédent) et une liste de transactions (une par agent sur le réseau).
  - **Algorithme de consensus :** Utilise l'algorithme SBFT pour se baser sur un générateur de bloc unique qui a un rôle privilégié dans le système. Le générateur collecte toutes les transactions des agents de surveillance, il commande et regroupe toutes les transactions vérifiées périodiquement (24 heures pour PingER). Le générateur diffuse le nouveau bloc sur tous les agents de surveillance du réseau, qui agissent comme signataires en signant uniquement le bloc signé par le générateur. Le consensus est obtenu lorsqu'un nombre minimum d'autres nœuds du réseau ratifient le nouveau nœud.
- **Couche d'accès aux données :** Le contenu des fichiers est accessible avec une faible latence et à partir de hachages à haut débit. L'IPFS fournit des services web rapides et permanents assurant le fonctionnement de PingER. IPNS et les graphes acycliques dirigés de Merkle peuvent donc être utilisés à travers IPFS pour traiter le contenu en utilisant un espace de noms auto-certifié et des DHT. Tous les fichiers de données sont accessibles depuis leur hachage cryptographique et la disponibilité des données sur le réseau est garantie par la présence de plusieurs emplacements de stockage sur le réseau.

Vie privée

[79] spécifie que le stockage de données privées des utilisateurs de façon immuable entre en conflit avec les règlements européens régissant la protection des données (RGPD) et le droit à l'oubli. Ces données ne peuvent donc pas être enregistrées de façon immuable directement dans un système de *blockchains*. L'IPFS peut être la solution si l'on y intègre directement un protocole de délégation anonyme pour distribuer une demande d'effacement de contenu parmi tous les nœuds du réseau lorsqu'une demande d'effacement de données est faite. Les demandes d'effacement sont alors enregistrées dans un DAG de Merkle afin d'éviter que

des utilisateurs ne revendiquent du contenu pour le supprimer. Les performances de ce nouveau protocole sont également étudiées. La diffusion des requêtes de suppression dans un réseau IPFS croît avec un temps linéaire (pente positive) à mesure que des nœuds s'ajoutent sur le réseau. De plus, pour augmenter les performances, les demandes pourraient avoir une limite de temps de vie.

[40] propose une plate-forme de partage de données basée sur la *blockchain* préservant la vie privée l'IPFS pour garantir l'anonymat des utilisateurs, ainsi que la confidentialité et une haute disponibilité des données. La *blockchain* Ethereum est utilisée, ainsi qu'un système de chiffrement des données. Les résultats des mesures présentent des temps de chiffrement, de déchiffrement, de signature, de vérification de la signature des données linéaires.

Stockage de données volumineuses

[6] propose une solution du stockage immuable de données volumineuses avec un accord entre deux parties. La solution se base sur l'utilisation d'une *blockchain* publique et une DHT. En outre, trois paramètres sont évalués comme étant essentiels dans les systèmes distribués : le temps (la rapidité de communications), le stockage et la performance. Deux cas d'étude sont présentés :

— le premier permet de trouver une solution pour réduire un problème de limitation du stockage ;
— le second permet d'améliorer la rapidité des transactions.

Les résultats expérimentaux (voir 2.6.6) révèlent que l'ensemble du système proposé peut être utile dans des systèmes décentralisés de stockage de données en parallèle et que le système est performant.

Dans la même optique que [108], qui tente de résoudre des problèmes de quantité croissante de données des transactions liées à la technologie *blockchain* en compressant les données d'un réseau de *blockchains* à l'aide de principes utilisés par l'IPFS (voir 2.6.5), [53] propose un modèle de stockage pour les *blockchains* (telles que Bitcoin [67], Ethereum [104], Hyperledger) utilisant IPFS comme moyen de stockage hors-chaine pour les transactions dans un bloc, ainsi que pour l'accès aux transactions d'un bloc particulier. Les mineurs utilisent le système de stockage IPFS pour y enregister une transaction et obtenir une adresse de contenu lié à la transaction dans le bloc de la *blockchain*. En effet, le système de hachage cryptographique de l'IPFS permet de réduire la taille des transactions dans un bloc. De plus, une technique de stockage adressée par le contenu est étudiée afin de sécuriser l'accès à la transaction pour un bloc particulier. Le modèle obtenu doit permettre de garantir un schéma de stockage efficace pour le réseau *blockchain* et de réduire la taille de chaque bloc de données sur la *blockchain*.

2.5.2 Fog/Edge Computing

[25] propose un système de stockage en réseau à grande échelle pour l'informatique Fog/Edge combiné avec l'IPFS. Il compare les performances de latence et de trafic réseau pour l'intégration d'IPFS avec celles de *Cassandra* et *Rados*. Rados et Cassandra évoluent difficilement dans le contexte du *Fog Computing*, tandis que l'IPFS a des latences plus faibles en matières d'accès distants et locaux. D'autre part, il convient d'adapter la gestion des méta-données par la DHT d'IPFS à l'activité locale du contexte du *Fog computing*.

2.5.3 Internet des objets (IoT)

[66] propose un *Framework* pour l'utilisation d'IPFS dans le contexte de l'IoT afin de remplacer un système de centralisation des données d'un modèle d'architecture applicative limité du type client-serveur par un système décentralisé. L'objectif est d'éviter d'être confronté à un point de défaillance unique, de garantir la confidentialité des données et de garantir un accès fiable aux données dans un système interopérable entre les systèmes applicatifs de l'internet des objets. Des capteurs Raspberry Pi embarquent chacun un processus IPFS et exécutent le protocole afin de créer des nœuds IoT. Les capteurs sont connectés au réseau, qui héberge les applications pour créer un essaim de nœuds, formant un cluster avec un nœud maître. Des tests de performances d'échanges sont effectuées entre des essaims de nœuds de tailles variables (de 3 à 20 nœuds), pour des fichiers de données de tailles variables (de 150 octets à 10 Mo). Des données sont envoyées à tous les nœuds du *cluster* simultanément. Le support pour le téléchargement depuis des objets connectés est jugé efficace pour des échanges de fichiers d'enregistrement vidéo, quelle que soit le nombre de nœuds dans le *cluster*. Le temps de téléchargement est néanmoins plus long avec l'augmentation du nombre de nœuds connectés. Ce temps plus élevé est cependant jugé pertinent, car les données sont envoyées en parallèle. L'expérience révèle que, pour cette taille de réseau, le modèle prouve la fiabilité du *framework* avec le système IPFS pour des architectures dédiées à l'IoT. Le *cluster* était stable durant les échanges et les fichiers ont été échangés efficacement avec une latence faible. Il est également estimé que l'étude prouve l'efficacité du modèle de manière appropriée étant donné que les réseaux embarqués échangent de petits fichiers de données.

[35] propose un système pour la technologie de l'Internet des objets et la gestion de données massives pour gérer la tracabilité de bout en bout de la provenance de produits agricoles vers le consommateur. L'objectif fonctionnel est de contrer des risques de sécurité alimentaire. Afin d'éviter la falsfication des données, la technologie *blockchain* est utilisée pour de stocker l'adresse de hachage IPFS des données de provenance. IPFS est utilisé en tant que système de stockage hors-chaîne pour des données de tous types (images, vidéos, données de capteurs). Les résultats expérimentaux estiment que la solution est plus performante pour le stockage de grandes quantités de données que d'autres méthodes existantes dans le même domaine (voir 2.6.7).

[3] propose une architecture de réseau pour assurer la confidentialité des données IoT à partir d'un système de stockage de données hors-chaine dans l'IPFS. Deux plate-formes de *blockchains* existantes sont utilisées pour créer une pile logicielle de contrats intelligents permettant de contrôler et de garantir l'accès aux données uniquement à des utilisateurs autorisés. Il s'agit d'Ethereum, la plate-forme de développement de *blockchain* Ethereum, et de Monax, la plate-forme de développement d'applications de *blockchain* pour les écosystèmes commerciaux. (algorithme de consensus Tendermint). Les performances de leur intégration dans l'IPFS sont étudiées et des considérations de déploiement sont décrites.

2.5.4 Systèmes multi-agents

[94] propose un *framework* de conception de systèmes distribués pair-à-pair ouverts pour des systèmes multi-agents (chaque agent peut entrer ou quitter le système à tout moment), combinant IPFS et la *blockchain*. Dans ce modèle, il est considéré qu'un agent a des informations partielles sur les données du système qu'il utilise. Le *framework* doit aider à gérer des systèmes pair-à-pair où des acteurs interagissent avec un accès aux données, leur découverte et la confiance en ces données. Le système fournit des lignes de conduite pour afin de déterminer la nécessité d'utilisation de la technologie *blockchain*, ou si d'autres technologies sont suffisantes.

2.5.5 Gestionnaires de contenus numériques et multimédias

[107] propose un modèle de stockage immuable décentralisé pour des certificats et des diplômes universitaires afin de garantir leur unicité et leur authenticité. Ils sont authentifiés à l'aide d'un algorithme de clé asymétrique et de la technologie *blockchain*. Des contrats intelligents Ethereum sont utilisés avec IPFS comme système de stockage hors-chaîne. Les documents sont récupérés depuis l'IPFS avec des mécanismes de contrôle d'accès (exemples : les étudiants sont uniquement autorisés à consulter ou télécharger leurs documents ; une personne unique autorisée par établissement peut envoyer un document sur le réseau).

[68] propose un système décentralisé qui s'assurer de l'originalité, de l'authenticité et de l'intégrité de contenus numériques publiés et mis en ligne librement, tels que les livres et des contenus multimédias. Il utilise la *blockchain* avec les contrats intelligents Ethereum pour gérer l'historique du contenu numérique. Le contenu numérique est stocké hors-chaîne avec l'IPFS pour assurer une haute intégrité et une accessibilité globale.

Ces systèmes ne protègent pas du piratage ou de la copie du contenu partagé car le contenu peut être reproduit et modifié hors-ligne une fois téléchargé. Tout autre utilisateur peut alors en revendiquer la propriété et son originalité, car les modifications hors-ligne ne sont pas consignées.

Framework de stockage de contenu : FileShare

[46] présente un *framework* dans lequel du contenu numérique peut être partagé dans un environnement sécurisé, inviolable, qui garantit la provenance d'opérations des lectures, des modifications et du partage de données sur un réseau distribué.

L'application FileShare garantit que le contenu numérique est accessible uniquement dans l'application et ne sera pas disponible depuis le système d'exploitation des utilisateurs finaux. Elle utilise les contrats intelligents de la *blockchain* Ethereum afin de stocker la provenance des données, ainsi que pour contrôler l'accès au contenu numérique. La plate-forme IPFS est utilisée pour héberger l'application définitivement et gratuitement.

L'application FileShare a cinq objectifs :

- **Inviolabilité de l'environnement :** La provenance des données est collectée et enregistrée. Ces données sont ensuite publiées sur le réseau *blockchain* pour assurer leur intégrité. Les données étant obligatoirement chiffrées, elles ne sont pas sujettes à une modification malveillante de la *blockchain* Ethereum ;
- **Unicité des fichiers partagés :** Les fichiers ne sont consultables qu'au travers de l'application FileShare de manière publique ou privée. Ils ne peuvent donc pas être téléchargés sur un système d'exploitation ;

— **Traçabilité de bout en bout :** Les origines des données d'un fichier sont accessibles aux utilisateurs afin d'en connaître les modifications, le propriétaire du fichier, ainsi que les lecteurs ;
— **Origine des données en temps réel :** Des journaux d'audit sont stockés sur la *blockchain*. Ces données peuvent être utilisées pour obtenir des informations analytiques ;
— **Validation de l'origine des données :** La provenance des données est publiée et validée par des nœuds sur le réseau *blockchain* TestNet/Ropsten d'Ethereum. L'interaction avec les nœuds utilise Web3.js et une connexion HTTP.

Le *framework* comprend quatre phases :

— **enregistrement et authentification des utilisateurs :** Les détails d'enregistrement de l'utilisateur sont ajoutés par l'application FileShare à la *blockchain* Ethereum ;
— **création et le stockage des fichiers :** L'utilisateur décide si le fichier doit être partagé avec d'autres utilisateurs à l'aide d'une clé publique de l'utilisateur avec lequel il souhaite partager le fichier, ou public, dont le propriétaire peut partager le fichier avec chaque utilisateur inscrit sur l'application FileShare. FileShare chiffre le fichier avec une clé de chiffrement AES-256. Le fichier crypté est ajouté au réseau IPFS et le réseau IPFS renvoie le hachage du fichier téléchargé. L'application déploie ensuite un contrat intelligent qui stocke les méta-données de chaque fichier partagé sur l'application, avec l'adresse IPFS du fichier chiffré et la clé publique de son propriétaire ;
— **récupération des fichiers et collecte :** Pour accéder aux fichiers, les utilisateurs doivent utiliser l'éditeur d'application de partage de fichiers car le fichier serait déchiffré uniquement dans l'éditeur d'application. L'application utilise le contrat intelligent de fichier pour accéder aux méta-données du fichier, extrait le fichier à partir d'IPFS, déchiffre le fichier et l'ouvre dans l'éditeur intégré ;
— **stockage de l'origine des données :** A chaque opération d'un utilisateur telle que le partage ou la lecture de fichiers, ce dernier a besoin d'une clé de déchiffrement fournie par le contrat intelligent déployé correspondant. Les appels aux fonctions du contrat intelligent sont enregistrés en tant qu'opérations de fichier effectuées dans l'éditeur pour collecter l'origine des données. Une fois qu'une opération est effectuée, l'enregistrement est généré, il est téléchargé sur le réseau *blockchain* et stocké dans la *blockchain* de provenance.

[69] propose un système de contrôle de version des documents utilisant les contrats intelligents de la *blockchain* Ethereum. Le système est considéré comme résilient après avoir été testé avec des outils de vérification de failles de sécurité potentielles pour la *blockchain* (*ChainSecurity*) et pour les contrats intelligents (*Oyente*).

[71] introduit *BlockIPFS* pour retracer et auditer toutes les activités associées à l'accès et la modification d'un fichier donné sur l'IPFS en utilisant la *blockchain Hyperledger Fabric*.

2.5.6 Autres domaines

De nombreux autres domaines utilisent l'IPFS comme système de stockage de données, tels que l'archivage des sites web sur la plate-forme *WayBack* de manière permanente [2] ; les domaines de la gestion des accès aux données [91, 12, 99] ; la gestion de données médicales [93, 52, 105] ; le recherche en cybersécurité [75, 41, 44, 55]. Dans le domaine des réseaux sociaux, [106] propose une *DApp* semblable à Twitter qui utilise la plate-forme de la *blockchain* Ethereum, les contrats intelligents et IPFS comme stockage hors-chaîne.

2.6 Mesures de performances de l'IPFS et de ses sous-systèmes

2.6.1 Comparaison des performances des protocoles IPFS et FTP

Dans [1] , le protocole IPFS est comparé au protocole de transfert de fichiers FTP pour l'envoi et la réception de fichiers de petites et moyennes tailles (1Ko à 64Mo) dans des conditions identiques. Le nombre de nœuds IPFS pour ces tests est faible (1 à 3 nœuds). Dans ces conditions, il ressort que le protocole IPFS a une latence plus élevée que le protocole FTP.

Les performances d'opérations en lecture avec IPFS se dégradent quand les données sont répliquées sur plus d'un nœud du réseau, et que les performances en matière de latence et de charge du CPU s'amenuisent à mesure que la taille des fichiers récupérés augmentent.

2.6.2 Performances de Bitswap

[30] compare le routage de contenu d'une solution décentralisée (DHT Kademlia) avec une solution centralisée (DNS) dans le cadre de Bitswap. Ceci afin de déterminer le rapport entre :

— le coût de récupération et d'identification des scénarios dans lesquels des solutions de routage de contenu échouent ;

— la rapidité à trouver le premier bloc de données.

Interprétation des résultats expérimentaux :

Les résultats expérimentaux montrent que le nombre de doublons stockés par des pairs du réseau détermine la réputation d'un bloc stocké dans le réseau.

Plusieurs éléments sont observés :

— Si des doublons sont répartis uniformément dans le réseau, la probabilité pour la DHT Kademlia et le DNS de localiser des blocs sans intervention de systèmes de routage externes est proche de 1 ;
— Le temps de la DHT Kademlia pour découvrir du contenu (même réputé) augmente avec le nombre de blocs de données du contenu demandé et donc avec la taille du réseau ($O(log(n))$ dans le pire des cas) ;
— Les éléments plus grands nécessitent de parcourir davantage la DHT, ce qui augmente le temps total de récupération ;
— Chaque bloc est adressé indépendamment (sans lien avec le fait qu'un seul nœud stocke possiblement tous les blocs d'un élément).

Le temps nécessaire ($O(log(n))$ dans le pire des cas) pour découvrir un bloc avec la DHT Kademlia dépend :

— de la taille du réseau (nombre de nœuds) ;
— de la réputation du contenu ;
— de la taille des éléments présents dans les tables de routage DHT des pairs ; la limite de temps relative étant $O(log_k n)$.

L'ensemble de la recherche des approches centralisées (exemple : le système DNS) dépend du contact avec une seule entité. Elles ont une probabilité élevée de découverte de contenu et une résolution rapide du contenu. La DHT décentralisée a une probabilité élevée de découverte de contenu et des opérations beaucoup plus lentes pour la recherche et la récupération.

[30] compare également la performance d'un sous-systèmes de routage de contenu DHT Kademlia avec et sans Bitswap, et démontre les améliorations de performance grâce à Bitswap. Les tests sont exécutés depuis 30 nœuds qui instancient l'implémentation Go d'IPFS et de Bitswap à travers un exécuter Docker de la plate-forme Testground dans AWS. Chaque nœud est exécuté dans un conteneur Docker différent sans limite de ressources, et avec des configurations spécifiques.

Interprétation des résultats expérimentaux :

— Les nœuds récupérant du contenu Bitswap sont capables de récupérer un seul bloc 30% plus rapidement en moyenne que ceux utilisant la DHT ;
— Le temps d'accès au premier bloc est en moyenne 30% plus élevé pour la DHT que pour le Bitswap ;
— Les nœuds des dernières vagues utilisant la DHT sont capables de récupérer le contenu avec des performances comparables à celles du Bitswap ;
— Bitswap récupère le bloc plus rapidement que la DHT de base (c'est-à-dire sans l'aide de Bitswap) ;
— La DHT doit contacter un nœud pour effectuer la recherche avant de demander la récupération du bloc. Bitswap est capable d'interroger ses pairs connectés et de demander immédiatement la transmission du bloc ;
— Plus le degré et le temps de vie sont élevés, plus la probabilité de découverte est grande. L'équilibre entre le temps de vie et la surcharge du réseau dépend de la taille du réseau et de la connectivité des nœuds décidée au niveau de l'application (en fonction de ses exigences) ;
— Les résultats de la compression sont déterminés par le nombre d'éléments redondants pouvant être exploités ;
— L'application d'une compression au niveau du flux des données (lors du transport) utilise un *wrapper* pour compresser chaque octet qui entre dans le flux d'un nœud. Cela peut améliorer les performances et utiliser efficacement la bande passante par rapport aux autres stratégies de compression ;
— En fonction de facteurs tels que l'algorithme de compression et le format du fichier, les économies de bande passante peuvent atteindre 50% ;
— Pour les grands ensembles de données, l'utilisation de la stratégie de compression au niveau du protocole permet de réduire la bande passante de 75% et d'économiser au moins 12% de la bande passante pour les ensembles de données de toutes tailles.

2.6.3 Performances de recherche dans les nœuds du *Merkle DAG*

La DHT Kademlia permet d'offrir une charge de calcul pour les requêtes de recherche de données dans les nœuds du réseau de l'ordre de $log_2(n)$ [59, 14]. Néanmoins, la charge de calcul peut devenir conséquente pour récupérer chaque objet de données dans l'arborescence des nœuds (Merkle DAG). Principalement parce qu'il faut rechercher la clé cryptographique dans le DHT, mais également parce qu'il faut se connecter à des pairs et récupérer les blocs. La mise en cache des DAG pour une durée indéfinie et leur aplatissement pour répertorier tous les objets accessibles depuis un arbre permet de réduire significativement la charge de calcul de la recherche [30, 14, 79, 29].

[8] présente une étude comparative des performances des arbres de Merkle. Elle compare un nouveau type d'arbre avec une approche vectorielle semblable à [83] utilisée par l'IPFS. Ce nouvel *arbre de recherche de Merkle* (*Merkle Search Tree*, ou MST) est équilibré et maintient ses identifiants dans l'ordre afin d'augmenter l'efficacité des mises à jour des ensembles de clés. Les mesures sont effectuées depuis un système composé de 1000 et 2000 nœuds synchronisés. Les résultats du MST sont comparés avec l'approche vectorielle.

2.6.4 Mesures des performances et amélioration pour le stockage

[88] propose une stratégie adaptée à l'arborescence de Merkle d'IPFS et qui optimise les performances d'enregistrement des données. Elle désigne le nœud de sauvegarde du fichier parmi les nœuds à proximité à l'aide d'un degré d'intérêt. La redondance des nœuds permet aux données enregistrées d'être distribuées pour respecter les fondements du système IPFS [14] et pour les rendre hautement disponibles. Cela tend d'ailleurs à corriger l'une des faiblesses d'IPFS, qui est le risque de présence de point de défaillance unique liés aux systèmes de stockage partiellement décentralisés.

Les performances de l'architecture distribuée de la solution y sont comparées selon deux approches :

— stratégie de sauvegarde distribuée et centralisée pour un système IPFS comptant 50 nœuds. Le taux de transmission et la bande passante utilisés ne varient pas. Le temps d'accès moyen aux nœuds est calculé en faisant varier leur nombre. Les constats sont les suivants :

1. **sauvegarde distribuée :** le temps d'accès est moindre par rapport à la sauvegarde centralisée lorsque le nombre de nœuds est élevé, et il augmente quand le nombre de nœuds augmente. La sauvegarde est lente car les fichiers ont plusieurs nœuds de sauvegarde sur le réseau.
2. **sauvegarde centralisée :** le temps d'accès augmente linéairement avec le nombre de nœuds d'accès avec une bande passante et un débit de transmission égaux.

— performances de la redondance et de la sauvegarde pour deux types de sauvegarde distribuées :

1. **la sauvegarde de proximité (monofactorielle)**.
2. **la sauvegarde d'intérêt et de proximité (multifactorielle)**.

Le système est testé avec un système IPFS de 20 nœuds et des paramètres de nœuds identiques. La redondance des deux stratégies est principalement déterminée par la distribution de la probabilité de disponibilité des nœuds. Lors d'un envoi vers un nœud, les traitements sont faits en fonction de la sauvegarde de proximité et de la sauvegarde du nœud voisin. Pour la sauvegarde de proximité, la proximité du système de stockage peut potentiellement augmenter le taux de satisfaction du nœud de sauvegarde.

La *sauvegarde par intérêt de proximité* tient compte du nœud, de la volonté de stockage et du taux de recherche. Il est observé que les performances de *sauvegarde par intérêt de proximité* sont toujours supérieures à la *sauvegarde de proximité*.

2.6.5 Performances de compression

[108] mesure la rapidité du réseau IPFS et son efficacité en intégrant des caractéristiques inhérentes au réseau IPFS, ainsi que les fonctionnalités de hachage cryptographique du protocole à un système de *blockchains* tel que *Bitcoin*. Cela permet de réduire le coût de stockage des données avec un ratio de compression du réseau de *blockchains* de 0.0817 par rapport à sa taille initiale. Cela est jugé être un taux de compression prometteur pour une *blockchain* pouvant peser plusieurs centaines de gigaoctets.

2.6.6 Performances avec la *blockchain* et une table de hachage distribuée

Comme expliqué précédemment, [6] propose une solution du stockage immuable de données volumineuses avec un accord entre deux parties. La solution se base sur l'utilisation d'une *blockchain* publique et une DHT. Et deux cas d'étude sont présentés. Pour mesurer les performances de la solution, [6] compare les deux cas d'étude avec un troisième cas d'étude utilisant la *blockchain* sans utiliser IPFS, ni les DHT.

En outre, un système est "très performant lorsqu'il peut stocker n'importe quelle taille de données dans un temps le plus court possible" [6] et "la performance est définie en considérant la capacité de stockage et le temps consommé" [6].

Des envois de données sont répétés 100 fois vers le réseau IPFS pour un volume de données variant entre 443Ko et 55,47Mo. Il est mesuré que :

— Le temps de connexion dépend de la vitesse en temps réel de la connexion ;
— La taille plus importante des données provoque un temps d'ajout des données plus élevé que les temps de réponse ;
— Les temps d'épinglage des données sont quasiment identiques ;
— Les deux cas d'étude avec IPFS ont des performances mesurées comme étant plus élevées que le système sans technologie de stockage IPFS ;
— La consommation en ressources est plus élevée lorsqu'il y a de multiples temps de référencement vers la *blockchain* ;
— Les performances sont meilleures lorsqu'il y a peu de requêtes vers la *blockchain* ;
— Les performances sont meilleures lorsqu'il y a une absence de limitation du stockage.

2.6.7 Performance de transferts avec la *blockchain*

[35] compare deux systèmes de *blockchains* interagissant avec des capteurs connectés :

— Le premier utilise les *blockchains* Ethereum de manière standard, sans IPFS.
— Le second utilise les *blockchains* Ethereum couplées à l'IPFS.

Le rapport de vitesse d'exécution entre les deux systèmes concurrents est mesuré. Pour ce faire, le volume de données de capteurs envoyées par minute sur le réseau et le volume de données de capteurs traitées par minute sur le réseau sont mis en relation. Les résultats expérimentaux dévoilent :

— lorsque le volume de données envoyées est faible (en dessous de 70 envois/traitements par minute), la vitesse de stockage est légèrement plus rapide sans le système combiné à l'IPFS ; l'envoi des données dans IPFS a un coût en temps de répartition des données sur le réseau qui est plus élevé qu'avec le stockage standard utilisant la *blockchain* Ethereum ;
— lorsque la quantité de données envoyées augmente au delà de 70 traitements par minute sur le réseau, un point de rupture est atteint et la quantité de données traitées IPFS augmente drastiquement. Pour 2000 traitements par minute, le système de *blockchains* Ethereum se stabilise à 70 envois par minute sur le réseau, tandis que le système couplé à l'IPFS se stabilise aux environs de 450 envois par minute sur le réseau. Cela est expliqué par une méthode d'empaquetage des données et leur envoi six fois par heure, afin d'obtenir une efficacité six fois plus élevée pour le système étudié par rapport au système de *blockchains* Ethereum standard ;
— que l'augmentation de la capacité des paquets peut potentiellement améliorer l'efficacité du stockage, mais que cela diminue la capacité d'obtenir des données en temps réel. Un compromis doit donc être fait entre l'instantanéité et l'efficacité de l'envoi des données.

2.6.8 Performances pour l'évolutivité du système

[87] évalue les opérations d'entrée et de sortie de données d'un client dans le système de stockage d'IPFS. Des instances réelles sont déployées et distribuées géographiquement avec le *cloud* Amazon EC2. Les résultats des mesures expriment que les schémas d'accès des clients affectent significativement les performances d'entrée et de sortie. Les opérations d'échange et de résolution seraient des facteurs de goulot d'étranglement lorsque les clients lisent des objets sur des nœuds distants.

[71] (voir 2.5.5) évalue l'évolutivité de BlockIPFS. La latence consommée est mesurée par les transactions d'échanges et de lecture de chaque fichier stocké dans le système en faisant varier le nombre de nœuds (3 à 27). Les résultats expérimentaux démontrent qu'il n'y a pas d'augmentation significative du temps des transactions lorsque le nombre de nœuds augmente. Le nombre de nœuds ne permet pas de connaître l'évolutivité pour un déploiement à grande échelle [42].

[100] mesure les performances d'évolutivité d'IPFS avec des facteurs de réplication et des *clusters* de tailles variables. Les résultats expérimentaux démontrent que le temps moyen des échanges de données stockées dans IPFS augmente avec le facteur de réplication et la taille des *clusters*. La vitesse des échanges de données diminue et le temps de réponse entre les pairs du réseau augmente. Il est estimé qu'IPFS est faiblement évolutif à cause d'une limitation de bande passante pour chaque instance des nœuds IPFS.

[22] propose une modification de l'architecture du système de fichiers d'IPFS en le combinant à la *blockchain* afin d'y sauvegarder les informations de chaque nœud de manière simplifiée et optimisée. Il introduit un

modèle en "zigzag" afin d'améliorer le mode de stockage en bloc du protocole. De plus, un concept de nœud avec un rôle de *fournisseur de services* doit permettre de fournir un meilleur débit aux utilisateurs individuels sans qu'ils ne doivent maintenir un nœud individuel. Les utilisateurs individuels interagissent avec d'autres nœuds par l'intermédiaire des fournisseurs de services pour échanger le contenu qui les intéresse, évitant ainsi la communication au niveau local avec d'autres nœuds. Les résultats révèlent plusieurs éléments :

— Le schéma de stockage des données volumineuses est optimisé pour les fournisseurs de services de contenu. L'occupation de la bande passante est moindre. Elle augmente uniquement lors d'une interaction avec les fournisseurs de services de contenu pour échanger des données sur le réseau ;
— Théoriquement, le protocole BitSwap peut être plus rapide et plus efficace par l'introduction d'un nouveau schéma de stockage qui combine trois schémas de réplication, ainsi qu'un modèle de stockage de données par codes d'effacement en "zigzag". Le temps de répartition avec les codes en "zigzag" pour réparer un bloc de données est meilleur que les codes RS et CRS [22] et le temps de latence d'accès dépend du temps de répartition.

2.6.9 Mesure de la résistance d'IPFS contre la génération d'un *botnet*

[75] expérimente l'hébergement de contenus malveillants dans IPFS afin de déterminer la manière dont un *botmaster* pourrait utiliser le réseau pour créer un réseau de robots (*botnet*) dans IPFS. Deux expériences sont menées :

— Comprendre si un hôte qui a un générateur d'adresses aléatoire pour IPFS (*Resource Identifier Generation Algorithm*, ou RIGA) est bloqué par un outil de mesure lorsque de nombreuses requêtes sont faites d'hôtes vers une passerelle (pour la résolution des requêtes) ;
— Mesurer le temps nécessaire pour qu'un *botmaster* distribue du contenu (1000 fichiers de 4Ko) à son réseau de *botnets* et étudier si les passerelles appliquent un seuil quelconque aux demandes des clients.

Les résultats expérimentaux démontrent que l'IPFS peut être utilisé pour mener de telles attaques avec une efficacité élevée. Les causes identifiées sont :

— le manque de mécanisme de suppression efficace ;
— le manque de mécanismes de défense au niveau des passerelles ;
— la vitesse du réseau permettant des attaques en temps réel (3 à 4 secondes, malgré la présence de valeurs aberrantes qui augmentent ce temps moyen). Ce temps doit néanmoins permettre de lutter contre les logiciels malveillants adaptables en temps réel en écartant rapidement le contenu malveillant.

2.7 Conclusion de l'état de l'art

Dans ce chapitre, les solutions que peut apporter l'IPFS en tant que système de stockage pour le web décentralisé ont été présentées à partir de la littérature scientifique. Les fondements de l'IPFS, les systèmes qui se confrontent aux technologies qu'il utilise, ses utilisations et ses performances ont été décrits. Cette technologie offre un système de stockage de fichiers pour la publication de données sur le web et fournit un protocole associé aux technologies distribuées pour le Web 3.0. En effet, dans la littérature scientifique, le système de stockage que propose l'IPFS est étroitement abordé dans des contextes utilisant les systèmes *blockchains* afin d'obtenir des applications complètement décentralisées comme résultat. L'IPFS permet de partager les données des pairs à l'aide de l'adressage de contenu en utilisant des liens uniques immuables, ainsi qu'à l'aide de ses protocoles sous-jacents. L'IPFS peut être utilisé seul ; dans des systèmes partiellement décentralisées ; il peut être combiné à des solutions telles que des systèmes de stockage complémentaires ; il peut être combiné aux *blockchains* pour une orientation d'application pour le Web 3.0. Enfin, la passerelle IPFS propose un web potentiellement permanent réparti sur des noeuds d'hébergement entre pairs et offrant un accès au contenu semblable au *World-Wide-Web* à l'aide de son protocole d'échange et d'adressage de contenu.

Chapitre 3

Développement de la recherche

3.1 Avant propos sur la contribution apportée

Dans cette section, la réalisation d'une application expérimentale doit aider à répondre à la question de recherche « Pourquoi et comment mettre en œuvre une solution de stockage décentralisé ; le cas d'IPFS ? ».

La méthodologie décrite dans ce chapitre doit permettre de répondre à la question de recherche en identifiant pourquoi (dans quel cas) et comment il est pertinent d'utiliser l'IPFS comme moyen de stockage. Cela est expliqué au travers de la description d'un processus de création d'une application web qui utilise le système de fichiers IPFS.
Les contextes d'utilisation de l'IPFS sont identifiés à partir de la littérature scientifique sur le sujet, ainsi qu'en fonction de l'identification de besoins et de contraintes rencontrés lors de la réalisation de l'application logicielle.
Les moyens d'hébergement et de mise à disposition publique du contenu de l'application utilisant l'IPFS sont étudiés.
De plus, des mesures des performances du réseau IPFS pour l'envoi et la récupération de données sont réalisées afin de mesurer le délai de réponse et le débit (quantité de données transférées) pour des requêtes parallèles à l'aide de l'outil de tests de charge JMeter [1]. Un nombre de nœuds de stockage de fichiers réduit a été utilisé afin de recréer un contexte proposant une disponibilité de contenu réduite pour des fichiers de tailles variables.

L'ensemble des résultats obtenus sont présentés et remis en perspectives dans le chapitre de discussion. Une réponse est apportée à la question de recherche dans la conclusion à l'aide d'une proposition d'arbre de décision permettant aux informaticiens intéressés par l'utilisation de l'IPFS de comprendre dans quel cas il peut être pertinent de choisir d'utiliser l'IPFS comme système de stockage décentralisé.

3.2 Problématisation et identification de la question de recherche

Comme cela est présenté dans l'état de l'art, la littérature scientifique présente des cas d'utilisation du système de stockage de fichiers IPFS, elle explique son contexte global, son fonctionnement, ses performances et les systèmes concurrents. Mais il n'est pas clairement expliqué pourquoi et comment utiliser IPFS. Il convient donc de se poser la question de recherche suivante : « Pourquoi et comment mettre en œuvre une solution de stockage décentralisé ; le cas d'IPFS ? ».

La question peut être divisée en deux parties :
D'une part, la première partie de la question de recherche est « Comment mettre en œuvre une solution de stockage décentralisé avec l'IPFS ? ». Cette question nécessite de comprendre le fonctionnement de la mise en place d'une solution par un développement informatique d'un *Proof of Concept*.
D'autre part, la seconde partie de la question de recherche est « Pourquoi mettre en œuvre une solution de stockage décentralisé avec l'IPFS ? ». Cette question tente d'identifier le cheminement des contraintes rencontrées qui peuvent amener un informaticien (développeur, analyste technique, architecte logiciel, chercheur) à réaliser une solution dans laquelle l'utilisation de l'IPFS peut être entreprise de façon pertinente.

La question de recherche implique plusieurs sous-questions, telles que « Pourquoi faire du stockage décentralisé ? » ; « Quelles solutions de stockage décentralisé existent ? » ; « Quels défis ces solutions soulèvent-elles ?

1. Apache Software Foundation, "Apache JMeter", `https://jmeter.apache.org/` (2022) (Date d'accès 16/06/2022)

» ; « Quels sont les principes de fonctionnement d'IPFS ? » ; « Quelles sont les performances d'IPFS ? ».

L'annexe B reprend les étapes d'identification de la problématique et de la question de recherche, ainsi que de la méthodologie de suivie pour le ciblage des articles en vue de la réalisation de l'état de l'art.

3.3 Méthodologie de recherche

Afin de répondre à la question de recherche, plusieurs expérimentations de la technologie IPFS ont été menées en parallèle dans un contexte concret. Une application expérimentale a été créée avec un scénario spécifique d'utilisation de l'IPFS comme solution de stockage pour les données. Ceci afin d'identifier pourquoi, quand et comment le stockage sur l'IPFS peut être utilisé pertinemment à travers un cheminement de réflexion complet. Une telle mise en œuvre a permis d'identifier les contraintes techniques rencontrées les unes après les autres lors de l'association d'une application et de l'IPFS, ainsi que les options envisageables pour les résoudre.

Les résolutions de problèmes ont été menées dans un contexte de développement identique, avec une base de choix technologiques semblables et à partir de développements d'applications logicielles en partant de zéro (*from scratch*). La technologie IPFS et son contexte d'utilisation sont étudiés dans la littérature scientifique (littérature blanche), mais la documentation pour l'utilisation pratique et la résolution des contraintes qui y sont associées sont principalement abordées dans la littérature grise, telle que des *white paper*. De plus, la documentation disponible à propos de l'IPFS et des systèmes associés pour la résolution de la recherche se trouve sur les sites web respectifs des systèmes présentés ainsi que dans de nombreuses sources non officielles qui traitent ces sujets et qui ne sont pas abordés dans la littérature scientifique (dépôts de code open source, blogs, vidéos, etc.).

3.3.1 Déroulement de la réalisation d'une application *Proof of Concept*

Etant donné que l'IPFS est destiné à proposer une solution pour aider à remplacer le web centralisé avec de l'adressage de contenu [14], il m'a semblé pertinent de répondre à la question de recherche par l'étude *Proof of Concept* (PoC) de la réalisation d'une application que l'on retrouve sur le web centralisé/partiellement décentralisé et qui requiert impérativement une sécurité des infrastructures et des données quel que soit le contexte dans lequel elle est déployée.
Dès le début de la recherche, l'utilisation d'une architecture du type client-serveur a été choisie arbitrairement pour la réalisation d'une application, car on le rencontre couramment dans les entreprises travaillant avec le web complètement centralisé/partiellement décentralisé. Ce type d'architecture permet, entre autres, d'ajouter une forme d'abstraction de la logique métier au niveau du serveur applicatif ; d'ajouter des couches de sécurité des accès au niveau du fonctionnement du serveur applicatif ; d'ajouter des ressources de configurations en fonction des besoins du serveur applicatif dans des fichiers sur les serveurs d'hébergement ayant des accès restreints ; de répliquer un contexte fonctionnel d'architectures applicatives susceptibles de ne pas utiliser l'IPFS. Au fur et à mesure du développement, des limites, des contraintes et des incohérences dans l'architecture ont été identifiées et d'autres formes de solutions architecturales ont été étudiées. La confrontation directe aux contraintes liées à l'IPFS a permis d'identifier des éléments techniques liés à l'adoption ou non de certaines solutions plutôt que d'autres pour le stockage de données. Cela a permis de créer un arbre de décisions pour l'utilisation de l'IPFS.

3.4 Installation de l'IPFS

Des dépôts sont disponibles depuis le site officiel d'IPFS [2] pour installer son système *core*. Le système est compatible avec les *operating systems* Windows, Mac, Linux. Il est possible d'installer le système pour l'utiliser uniquement en ligne de commandes (CLI), et/ou avec une application cliente *IPFS Desktop*. Le CLI permet d'utiliser les fonctionnalités d'IPFS uniquement en lignes de commandes. Le client IPFS Desktop est une application de bureau : elle propose une interface graphique permettant d'utiliser les fonctionnalités du protocole, telles que d'épingler des fichiers afin de les rendre disponibles définitivement sur un nœud (ou tous les nœuds d'un cluster) ; d'héberger et de partager du contenu avec le réseau ; de visualiser d'autres nœuds du réseau ; de parcourir les Merkle DAG des nœuds en cherchant ou en inspectant du contenu mis à disposition du réseau ; etc.

Le système IPFS peut être installé sur une machine hôte et permet à cette dernière de devenir un nœud du réseau (public, ou bien privé s'il n'est pas connecté au réseau). D'autres outils sont disponibles depuis le site officiel, tels qu'une extension pour la prise en charge des adresses IPFS par un navigateur qui ne le prend

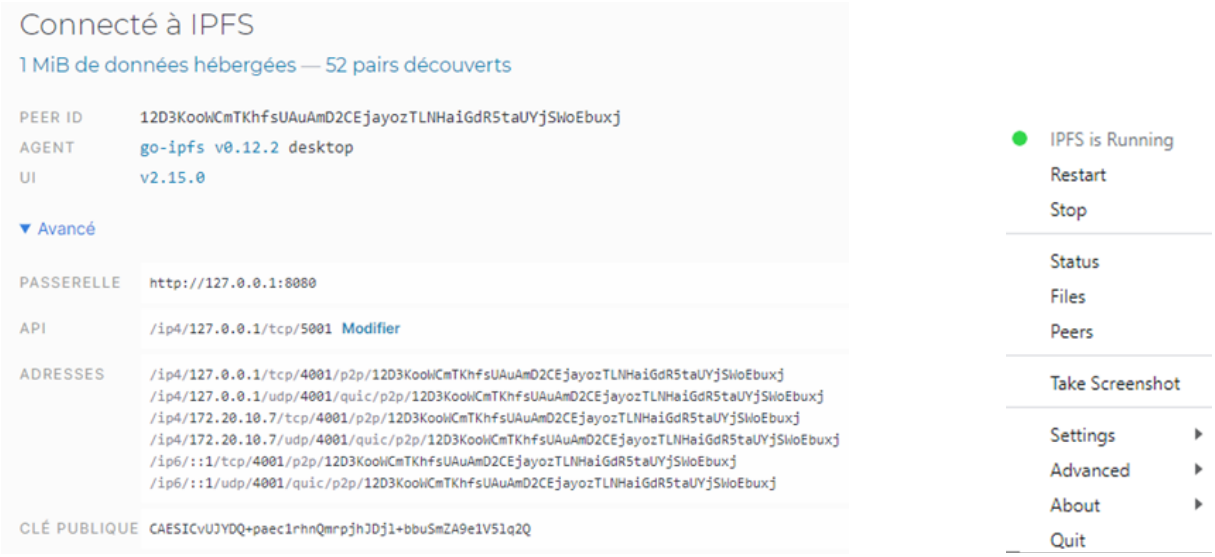
2. Protocol Labs, "IPFS powers the Distributed Web", `https://ipfs.io/#install` (2022) (Date d'accès 16/05/2022)

pas en charge (par exemple, le navigateur Brave intègre un support des adresses IPFS) ; un orchestrateur de nœuds afin d'héberger des clusters IPFS ; un accès aux *repositories* Github pour l'accès aux bibliothèques de code dédiées à l'utilisation et au développement du réseau IPFS ; des exemples [3] et idées de cas d'utilisations du système IPFS [4].

3.4.1 Mise en route d'une instance IPFS

D'un point de vue pratique, [14] propose une bibliothèque IPFS permettant d'utiliser le protocole pour le développement applicatif, un gestionnaire graphique d'IPFS (IPFS Desktop), ainsi qu'un outil en ligne de commandes (CLI) afin de pouvoir manipuler des objets de données et partager des données en interagissant avec les nœuds du réseau. L'installation d'IPFS desktop et du CLI ont permis de démarrer un nœud IPFS sur une machine locale, graphiquement ou en lignes de commandes, de la même manière qu'un serveur applicatif.

Plusieurs informations sont associées au nœud, telles qu'une adresse de l'API IPFS ; une adresse de passerelle vers le réseau ; la clé publique du nœud et plusieurs formes d'adressage du nœud local associées à un *PeerID*.



(a) Informations d'une instance IPFS (b) Menu IPFS

FIGURE 3.1 – Instance IPFS en cours d'utilisation

L'installation du système a créé un dossier local sur la machine hôte (path « /.ipfs/ »), comme représenté sur la figure 3.2, dans lequel les fichiers sont placés lorsqu'ils sont mis à disposition du réseau s'ils ne sont pas dans un autre répertoire de la machine hôte. En effet, des fichiers peuvent également être manipulés depuis d'autres répertoires de la machine hôte à l'aide du code d'une application.

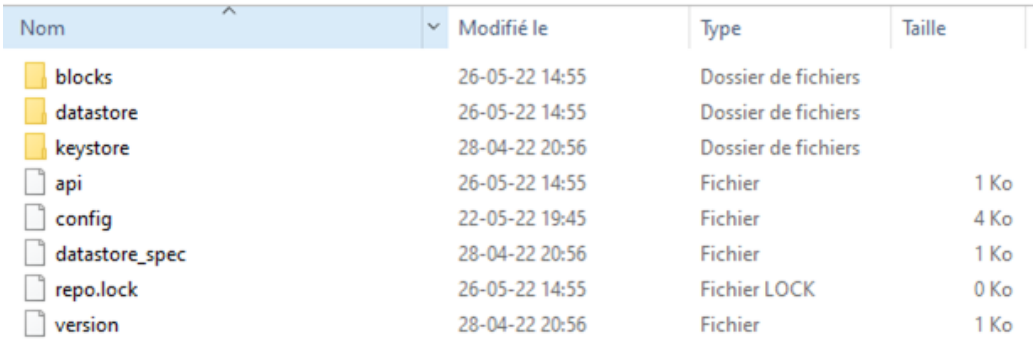


FIGURE 3.2 – Contenu de la structure des fichiers du répertoire de l'instance IPFS

Avec l'installation standard de l'IPFS, le système IPFS peut être utilisé pour partager, héberger, accéder à

3. "Awesome IPFS", `https://awesome.ipfs.io/` (2022) (Date d'accès 19/05/2022)
4. Protocol Labs, "Usage ideas and examples", `https://docs.ipfs.io/concepts/usage-ideas-examples/` (2022) (Date d'accès 18/05/2022)

du contenu depuis le CLI ou l'interface graphique. Néanmoins, pour une application utilisant les fonctionnalités d'IPFS, il convient d'utiliser l'une des bibliothèques (API) mises à disposition sur le *repository* Github IPFS [5].

3.4.2 IPFS sur téléphone mobile

L'application IPFS Lite permet d'utiliser les fonctionnalités d'échange et d'hébergement de contenu au sein d'un nœud IPFS sur *smartphone*. Elle propose également un navigateur permettant de consulter le contenu publié. [6]

3.5 Réalisation d'une application client-serveur avec l'IPFS

L'application développée pour ce *Proof of Concept* est une application de transactions monétaires entre deux comptes bancaires de deux utilisateurs différents d'un même système. Les données enregistrées sont destinées à être sauvegardées sur IPFS afin qu'il se substitue à un système de stockage central (base de données ou système de fichiers) pour le stockage des données structurées (objets) et non structurées (fichiers de données). Dans ce contexte, il a été choisi de développer une application offrant un système simple de transactions de paiements à l'aide d'un scanner de QR codes. Les QR codes sont des fichiers png, et les données des transactions sont enregistrés sous la forme d'objets de données.

La réalisation se limite principalement à l'étude de l'utilisation de l'IPFS pour le stockage des données, sans complètement intégrer les aspects de sécurisation des applications.

3.5.1 Objectifs du *Proof of Concept*

Cette expérience doit permettre de mettre en évidence :
— la manière dont IPFS peut être utilisé pour partager du contenu temporaire et persistant pour une transaction telle qu'un paiement d'un compte bancaire vers un autre entre deux utilisateurs ;
— les moyens d'implémentation d'une application client-serveur avec la technologie IPFS ;
— les difficultés rencontrées (contraintes techniques ; choix architecturaux ; choix d'implémentation ; choix technologiques) pour créer un système partiellement/complètement décentralisé avec l'utilisation d'IPFS comme moyen de stockage de contenu, tout en garantissant la sécurité des données des utilisateurs.
— Les moyens disponibles pour que des données privées des utilisateurs enregistrées sur l'IPFS restent sécurisées étant donné que tout utilisateur du réseau possédant un lien d'adressage de contenu (CID) peut accéder et télécharger des données stockées sur IPFS tant qu'un nœud du réseau les rend disponibles.

3.5.2 Choix technologiques préalables

Initialement, il a été délibérément choisi de créer une application avec une architecture client-serveur comprenant plusieurs choix technologiques que l'on retrouve couramment dans le monde professionnel pour la réalisation *from scratch* d'une application web :
— **Partie serveur :** Le langage du code utilisé côté serveur (back-end) est le langage Java, pour la création d'une application avec le *framework* Spring Boot. Les objets de données (objets structurés) utilisés dans l'application respectent des structures interprétables par l'ORM Hibernate. Ces objets pourraient être menés à être enregistrés sur l'IPFS, ou bien en base de données. La bibliothèque Java-ipfs-http-client [7] proposant une API pour la configuration et l'utilisation d'IPFS est incluse dans le projet. C'est donc la partie *back-end* Java de l'application serveur qui contactera l'IPFS.
— **Partie cliente :** La partie cliente est une application qui combine le langage *TypeScript* avec un environnement *Node.js* dans le *framework* Angular. La partie *front-end* JavaScript a pour objet de contacter des API du *back-end* Java afin de laisser les responsabilités de logique métier à la partie serveur de l'application. La partie cliente s'occupe uniquement du service d'interface entre l'utilisateur et le serveur applicatif.
— **Base de données :** L'objectif de ce cas d'étude est d'utiliser l'IPFS comme un moyen de stockage pour remplacer l'utilisation d'un système de stockage centralisé. Néanmoins, cela est fait par étapes. Pour l'étude de l'utilisation de l'IPFS, il convient de comprendre quand et comment une base de données peut être complètement remplacée par l'utilisation d'IPFS. Cependant, l'utilisation d'une base de données lors du processus de développement doit permettre de ne pas être bloqué par d'éventuelles contraintes

5. Protocol Labs, "IPFS A peer-to-peer hypermedia protocol", `https://github.com/ipfs` (2022) (Date d'accès 16/05/2022)
6. Remmer Wilts, "IPFS Lite App", `https://play.google.com/store/apps/details?id=threads.server&hl=en&gl=US` (08/07/2022) (Date d'accès 12/07/2022)
7. IPFS Community, "A Java implementation of the HTTP IPFS API", `https://github.com/ipfs-shipyard/java-ipfs-http-client#usage` (17/08/2020) (Date d'accès 21/05/2022)

techniques. Une base de données PostgreSQL est installée sur la machine hôte de l'application (serveur) afin d'y enregistrer des données lorsque cela ne l'est pas dans l'IPFS. Cette configuration permet également de comprendre la manière dont peuvent coexister une base de données avec le système de fichiers IPFS dans une même application.

3.5.3 Mise en contexte : scénario de cas d'utilisation de l'application

Paul et Bob sont connectés à l'application depuis leurs téléphones respectifs. Chacun possède un compte utilisateur avec des données bancaires (comptes en banque, numéros de comptes, montants disponibles). Paul créée une demande de transaction de paiement afin de pouvoir être payé par Bob pour un montant de 100€ vers l'un de ses comptes bancaires. La création de la transaction doit générer un QR Code permettant à un Bob de le scanner depuis l'application du téléphone et d'accepter la transaction de paiement depuis l'un de ses comptes vers le compte spécifié de Paul. Si elle est acceptée, la transaction est effectuée et le montant est versé depuis un compte bancaire de Bob vers un compte bancaire de Paul. Bob et Paul peuvent ensuite respectivement consulter les transactions qu'ils ont effectuées précédemment. Le diagramme de séquence 3.3 représente le déroulement d'une transaction de paiement.

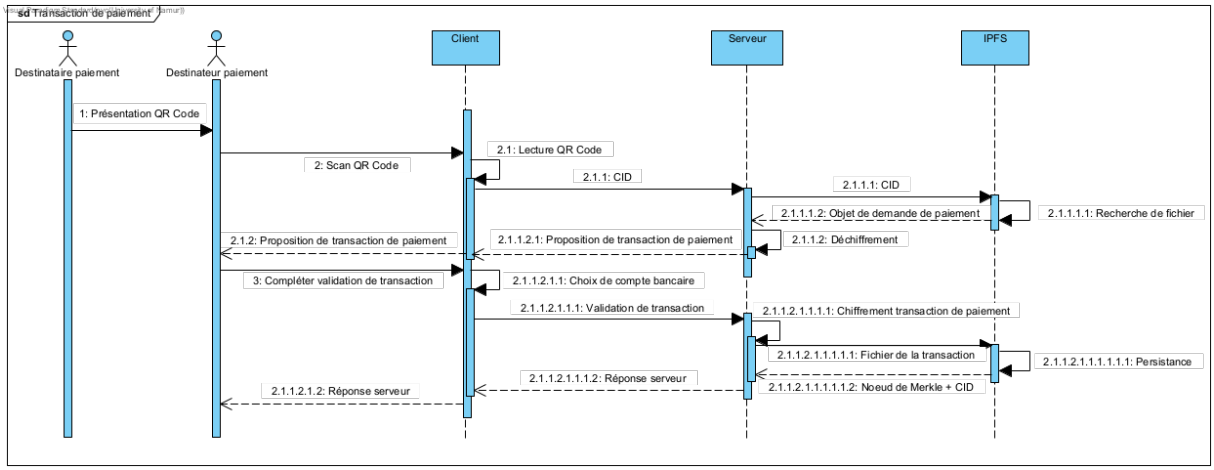


FIGURE 3.3 – Diagramme de séquence d'une transaction de paiement.

La figure 3.30 représente le fonctionnement de l'application *Proof of Concept* réalisée dans le cadre de la recherche.

Pour des raisons de limitation du développement à l'étude du stockage décentralisé et pour des raisons de facilités techniques, il est initialement décidé que le moyen de connexion à l'application n'est pas relié à IPFS. Les utilisateurs sont connectés à l'application à l'aide d'un nom d'utilisateur et d'un mot de passe qui sont en base de données. Le sujet de la décentralisation des données pour une application complètement décentralisée est réabordé plus loin (voir 3.6.4), lors d'une étude d'un moyen de décentralisation totale de l'application (application distribuée).

3.5.4 Communications entre le client et le serveur

Dans une application partiellement décentralisée, la partie cliente a pour objet d'être uniquement l'interface visuelle depuis laquelle les utilisateurs interagissent avec le serveur. Elle communique avec l'application serveur (back-end) à travers des services web (des *API*). Dans ce contexte applicatif, le client *Angular* utilise le protocole http pour communiquer avec le serveur applicatif Java à l'aide de contrôleurs REST *Spring*. Le système de fichiers du nœud IPFS stocke les fichiers HTML, CSS et JavaScript qui communiquent avec l'application serveur. Il stocke également les fichiers liés aux demandes et aux transactions de paiements. L'application serveur Java sert la logique métier ; elle communique avec l'application client, l'IPFS, la base de données et le serveur physique qui héberge le serveur applicatif. Le serveur applicatif exécuté sur chaque nœud conserve les fichiers de configurations de l'application et héberge des fichiers de l'application Java. Il est possible d'interagir avec l'IPFS depuis l'application Java à l'aide d'une API mise à disposition par la bibliothèque Java IPFS http Client [8] disponible sur le dépôt Github du projet IPFS. Grâce aux interfaces mises

8. IPFS Community, "A Java implementation of the HTTP IPFS API", `https://github.com/ipfs-shipyard/java-ipfs-http-client#usage` (17/08/2020) (Date d'accès 21/05/2022)

à disposition par cette bibliothèque, l'application Java peut se connecter à un nœud IPFS via sa passerelle, ainsi qu'échanger (envoyer et récupérer) des objets de données avec l'IPFS. L'API met à disposition des méthodes de récupération d'objets de données à partir d'adresses de contenu (CID), ainsi que des méthodes d'envoi d'objets sur le réseau. Les objets de données IPFS sont manipulés en tant que type *NamedStreamable*. Il sont récupérés sous des formats byte[], qui doivent ensuite être désérialisés vers des objets de l'application Java.

3.6 Description de la méthodologie de réalisation de l'application client-serveur

Pour démarrer le contexte IPFS, l'application Java doit posséder une classe de configuration IPFSConfig qui instancie un objet IPFS à l'aide d'un objet de type *multi-addresse* (type *MultiAdress*). Cela permet d'associer un à plusieurs nœuds IPFS à l'application Java à partir de la passerelle réseau. Dans ce contexte, un nœud local est donc associé à cette application (multi-addresse : « /ip4/127.0.0.1/tcp/5001 »). Cette valeur associée à la propriété « API » est lisible depuis l'interface graphique d'IPFS Desktop (voir 3.1). Le démarrage de l'application, un service Spring (singleton) instancie la classe de configurations afin de pouvoir faire communiquer l'application avec le réseau IPFS à l'aide des méthodes de l'API java-ipfs-http-client.

3.6.1 Echange de données avec le réseau

Toutes les classes ayant pour objectif d'échanger du contenu disponible avec le réseau doivent interagir avec deux méthodes définies dans une classe de type service *IPFSService* de l'application :

- *uploadToNode* (deux implémentations possibles) avec en paramètres :
  - **input :** un fichier (MultipartFile) , ou un chemin de fichier (String). Le fichier récupéré par l'implémentation de la méthode est ensuite encapsulé dans un objet de type *NamedStreamable.FileWrapper* et est ajouté au réseau (envoyé) à l'aide de la méthode ipfs.add(MultipartFile) de l'API *Java IPFS http client*.
  - **output :** un nœud du DAG de Merkle (type *MerkleNode*) associé à l'envoi du fichier est récupéré en réponse de la méthode ipfs.add. Le hachage cryptographique (type *Multihash*) associé à ce nœud est extrait (merkleNode.hash) et renvoyé sous un format interprétable en tant que CID (type String en Base58).
- *downloadFromNode(hash : MultiHash)* :
  - **input :** un hachage cryptographique (CID) au format *String* en Base58. Ce dernier est transformé en objet *MultiHash* et envoyé à l'aide de la méthode IPFSConfig.IPFS.cat(MultiHash) de l'API *Java IPFS http client*.
  - **output :** Le contenu associé au hachage cryptographique est retrouvé sur le réseau à l'aide de la Kademlia DHT sous la forme d'un tableau de bytes (*byte[]*). Cette structure *byte[]* doit être désérialisée pour pouvoir être récupérée en un objet Java standard (type *Object*). L'utilisation complémentaire d'un *casting* pour convertir l'objet de type *Object* vers le type initial de l'objet avant l'envoi vers l'IPFS permet de le récupérer sous une forme interprétable par l'application Java.

Lorsqu'un utilisateur désire recevoir un paiement d'un tiers, il encode une demande de transaction de paiement (ou demande de paiement, un objet structuré de type *RequestPayment*). Il remplit un formulaire avec plusieurs informations, telles que le numéro de compte bancaire cible du paiement, un intitulé, une description, un montant. Ces informations sont encapsulées dans un objet spécifique avec des métadonnées de l'utilisateur et de la requête. Elles sont ensuite envoyées sur l'IPFS en tant que fichier structuré et un *Content IDentifier* est généré par le protocole. La figure 3.4 représente cette fonctionnalité. La figure 3.21 représente la séquence complète.

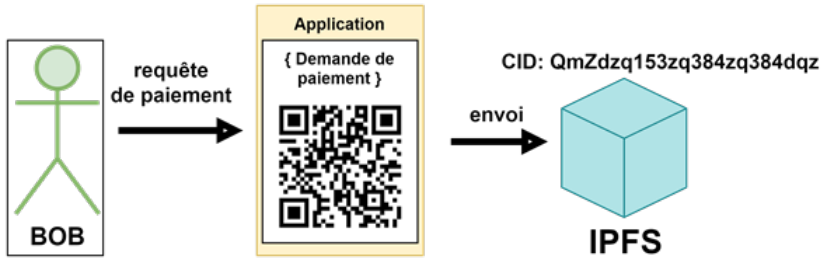


FIGURE 3.4 – Envoi d'une demande de paiement sur IPFS

La méthode *uploadToNode* récupère l'adresse de contenu qui référencie l'objet *RequestPayment*. Ce CID

est transformé en QR Code afin que la transaction puisse être récupérée sur le réseau IPFS à tout moment par un utilisateur en le scannant. L'utilisateur connecté qui scanne le QR code peut lire les informations de la transaction et accepter le paiement s'il le souhaite.

Une fois le QR code généré, il est à son tour envoyé dans IPFS afin d'être stocké et un CID qui le référencie est récupéré en réponse de *uploadToNode* (voir 3.6.1). L'utilisateur qui crée la transaction peut visualiser le QR code depuis l'application cliente grâce à ce lien. Il suffit ensuite de scanner le QR code pour récupérer l'objet RequestTransaction sur le réseau IPFS. Il est téléchargé à l'aide du CID associé, qui est passé en paramètres à la méthode *downloadFromNode* (voir 3.6.1).

3.6.2 Plusieurs défis apparaissent à cette étape du développement

1. **Objets lisibles sur le réseau :** Si l'on se connecte sur le réseau IPFS, on constate que les informations envoyées sur le réseau sont clairement lisibles par tout utilisateur possédant l'adresse de contenu (CID). Le contenu est accessible à l'aide de la passerelle IPFS, ou encore à l'aide de l'inspecteur de contenu si l'on héberge un à plusieurs nœuds (voir 3.5). Ces informations sont consultables sous un format de type texte ; un objet JSON ; un autre format téléchargeable d'un type de fichier structuré (objet de données) ; un objet non structuré. Il est donc nécessaire de chiffrer tous les objets envoyés sur le

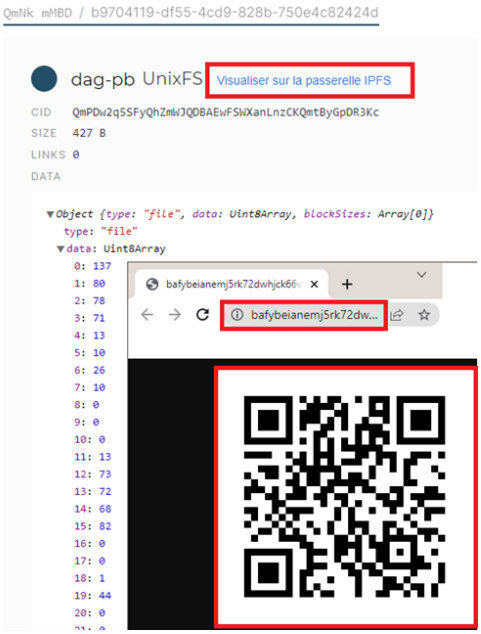


FIGURE 3.5 – Visualisation d'un fichier PNG QR Code envoyé directement sur le réseau IPFS

réseau depuis la méthode *uploadToNode* (voir 3.6.1) et de les déchiffrer lors de leur récupération avec la méthode *downloadFromNode* (voir 3.6.1) afin qu'ils ne soient pas lisibles par d'autres personnes que par les utilisateurs finaux. La figure 3.6 représente la recherche d'un même fichier hébergé sur l'IPFS par deux utilisateurs. Le fichier est récupéré en clair sur le réseau car il n'a pas été chiffré préalablement.

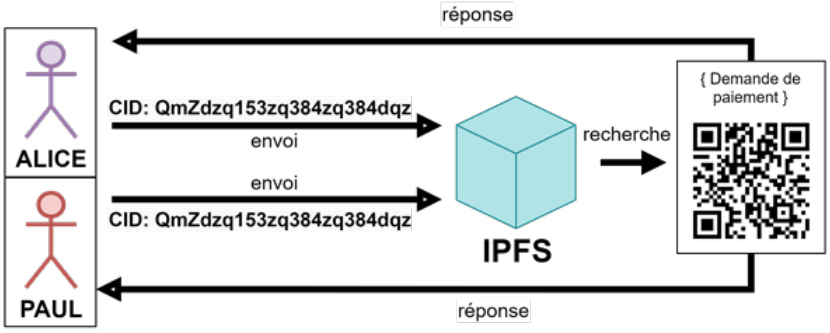


FIGURE 3.6 – Récupération d'un fichier QR code non chiffré sur le réseau IPFS par deux utilisateur à partir d'un CID

2. Les demandes de paiement (fichiers structurés) et les QR codes (fichiers non structurés) générés sont enregistrés sur l'IPFS. A chaque envoi de données sur le réseau, le CID associé à l'objet envoyé est récupéré en réponse de la méthode *uploadToNode*. Ces adresses de contenu ne sont pas indexées, ni liées entre elles. Si elles ne sont pas directement utilisées, il est compliqué de les retrouver et même de comprendre à quoi elles correspondent exactement si on les retrouve par la suite en inspectant l'objet relié à l'adresse. Il pourrait par exemple s'agir d'une version obsolète d'un fichier de données d'une application quelconque. Il convient donc de trouver un moyen de les indexer ou de les relier entre elles afin que l'application courante ne perde pas la trace des données les plus à jour.
   Plusieurs solutions sont envisagées :

   (a) **Utiliser une base de données centralisée :** La base de données relationnelle PostgreSQL installée préalablement pour l'application peut être suffisante pour indexer les CID des objets les plus récents à l'aide d'une table spécifique. Cela permet garder en mémoire les données les plus récentes associées à chaque utilisateur, ainsi qu'à garder un historique. Cependant, cela n'aurait pas de sens d'utiliser

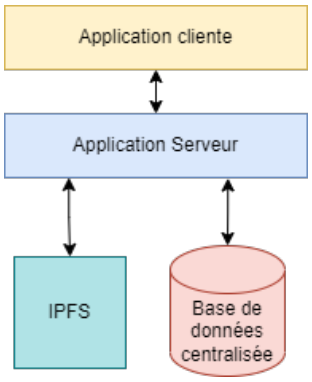


FIGURE 3.7 – Architecture combinant l'IPFS et une base de données centralisée

   l'IPFS partiellement pour enregistrer des données de paiement/transactions ainsi que les QR codes, mais d'utiliser une base de données pour les indexer. Il semble pertinent de faire un choix entre ces deux types de solutions.

   (b) Utiliser une base de données distribuée qui stocke les données sur IPFS, telles que OrbitDB [9] ou AvionDB [10]. Ces bases de données sont destinées aux applications complètement décentralisées

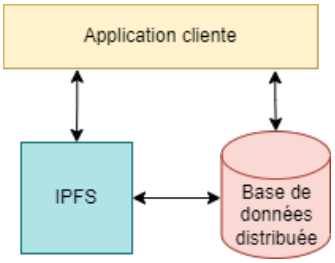


FIGURE 3.8 – Architecture combinant l'IPFS et une base de données distribuée

   utilisant l'IPFS. Elles sont utilisables uniquement à l'aide d'une application JavaScript, sans l'intervention d'un serveur applicatif intermédiaire. Il n'y a donc pas d'API disponible et applicable pour une architecture client-serveur. [38] Cependant, les CID IPFS sont récupérés sous la forme de

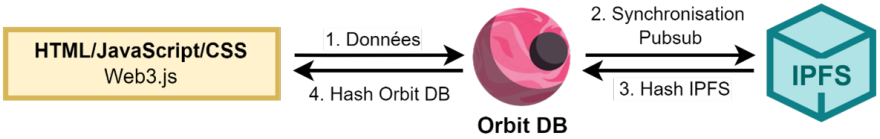


FIGURE 3.9 – Architecture utilisant une base de données distribuée Orbit DB

9. OrbitDB Community, "OrbitDB", `https://orbitdb.org/` (2021) (Date d'accès 08/06/2022)
10. IPFS Community, "AvionDB : A Distributed, MongoDB-like Database", `https://github.com/dappkit/aviondb` (13/10/2020) (Date d'accès 04/06/2022)

hachages OrbitDB, qui doivent également être indexés à leur tour. Pour une application distribuée, il convient d'indexer les CID à l'aide d'une *blockchain*.

(c) **Utiliser le système de fichiers d'IPFS pour y ajouter un répertoire par utilisateur :** Pour chaque utilisateur, ce répertoire possède des sous-répertoires spécifiques dédiés aux types d'objets de fichiers persistés dans l'application (Requêtes de paiement, QR codes, Transactions effectuées, etc.). Le contenu de chaque dossier est listé par répertoire avec des métadonnées (identifiant technique, date de création, date de dernière mise à jour, ...) permettant de les indexer dans l'application et qu'ils restent consultables tant qu'ils sont disponibles sur le réseau.

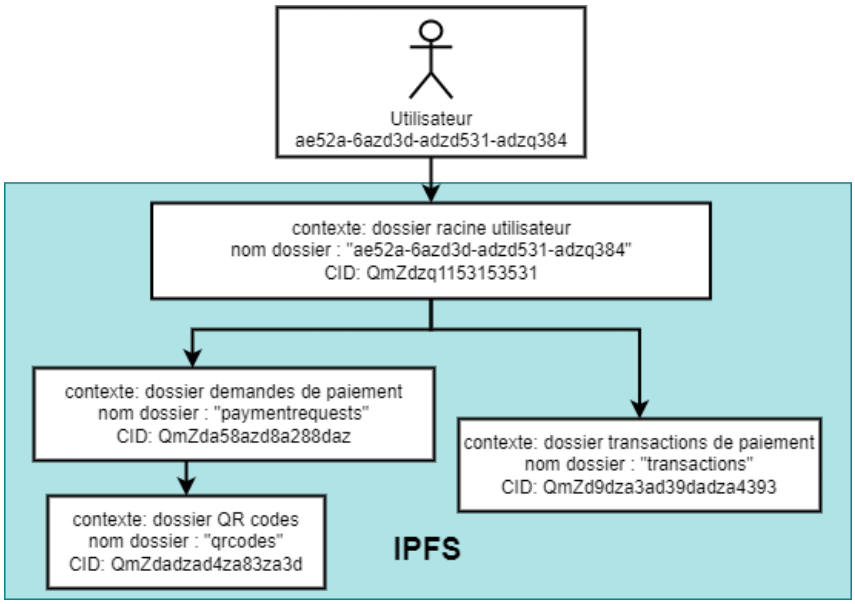


FIGURE 3.10 – Structure des dossiers d'un utilisateur sur l'IPFS. L'entité utilisateur référencie chaque répertoire catégorisant ses fichiers de transactions de paiement, de demandes de paiement, des QR codes. Les répertoires se retrouvent dans un dossier parent spécifique à l'utilisateur.

Cela implique plusieurs contraintes :

i. Il est nécessaire que les données restent illisibles par d'autres personnes que l'utilisateur propriétaire du dossier.

ii. Il faut conserver les références (CID) des dossiers dans l'objet utilisateur (cela peut être en base de données, dans un fichier sur l'IPFS, ou dans une *blockchain*).

iii. Lors de l'envoi d'un nouvel objet de données sur le réseau, un nœud est créé dans le Merkle DAG (objet de type *MerkleNode*). Ce dernier est référencé par zéro, un ou plusieurs nœuds de Merkle. Les données doivent être liées entre elles pour organiser la hiérarchie des fichiers. Récupérer les liens de cette hiérarchie permet d'indexer les objets toujours présents sur le réseau. A chaque modification du dossier de l'utilisateur, un nouvel objet immuable est créé et référencé (CID) dans l'IPFS. Ce lien doit être relié comme nouvelle référence de dossier de l'utilisateur pour remplacer l'ancienne. La figure 3.11 représente la manière dont la modification du contenu d'un dossier d'utilisateur est répercutée sur l'IPFS. La figure 3.12 représente le diagramme de séquence pour la création d'un utilisateur et l'ajout de la structure de dossiers de l'utilisateur sur l'IPFS avant la persistance des *Content IDentifiers* associés dans la base de données de l'application. Cependant, une solution hybride pour le stockage de données privées conciliant l'IPFS et une base de données n'est pas pertinente. Ce point est rediscuté plus loin (voir 2d). Il convient dans ce cas d'utiliser les *blockchains* pour le référencement des CIDs (voir 2(c)iv) et de remplacer l'utilisation de bases de données par une solution de stockage hors-chaîne (voir figure 3.17) afin que les informations de l'utilisateur soient complètement décentralisées (voir figure 3.23).

iv. Dans le cas des applications distribuées, l'utilisation d'une *blockchain* permet de conserver un index immuable des transactions référencées par les CID, tout en garantissant qu'une référence vers un CID n'a pas été modifiée (par l'utilisation d'un système de preuve). L'utilisation de contrats intelligents Ethereum permet d'automatiser le processus en échange de récompenses et de remplacer les méthodes transactionnelles des applications serveur par une architecture du

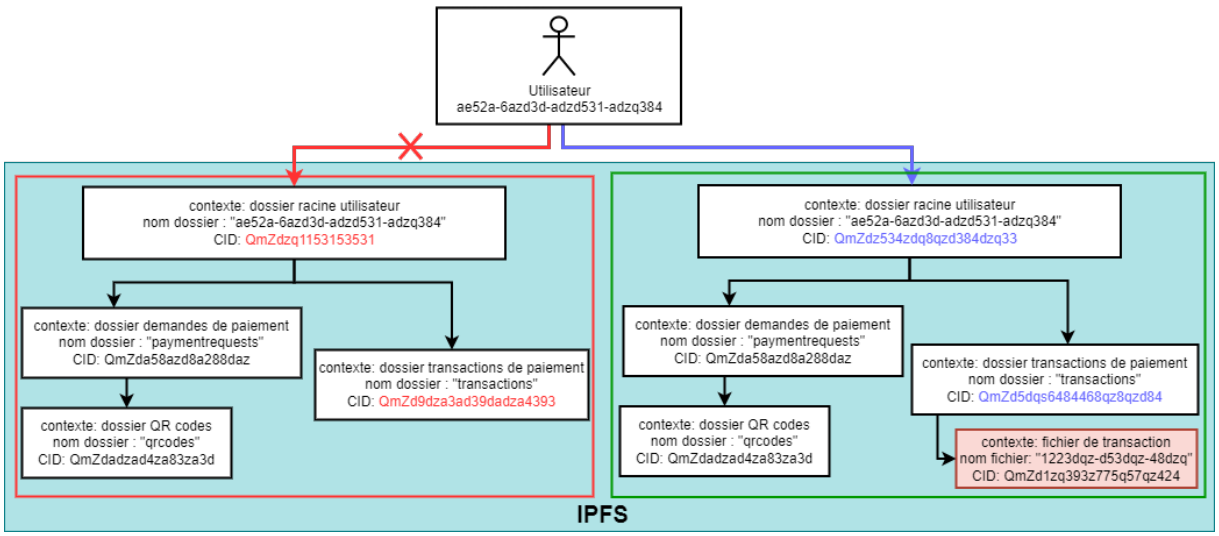


FIGURE 3.11 – Modification d'une structure de dossiers d'utilisateur sur l'IPFS. La structure des dossiers de l'utilisateur initialement créé sur l'IPFS est encadrée de rouge. L'application supprime les CID associés à ces structures pour l'utilisateur courant et crée de nouveaux liens pour la structure (encadrée en vert) qui a été mise à jour avec l'ajout d'un nouveau fichier dans le dossier des fichiers de transactions.

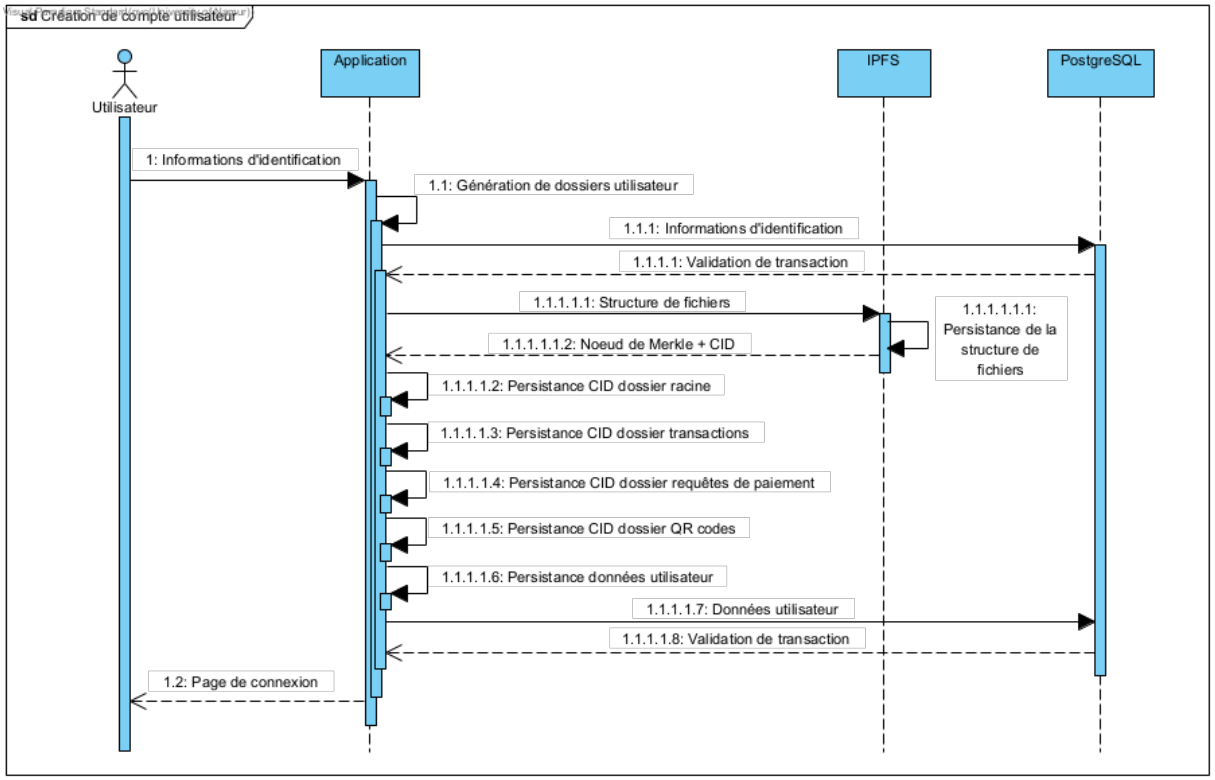
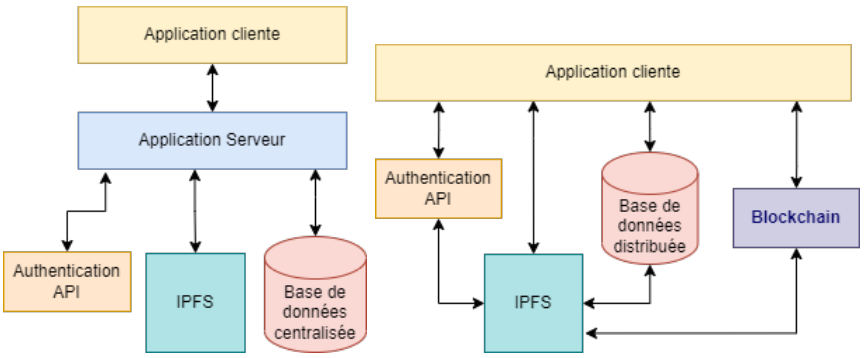


FIGURE 3.12 – Séquence pour la création d'un utilisateur et l'ajout de la structure de dossiers de l'utilisateur sur l'IPFS avant la persistance des *Content IDentifiers* associés dans l'application. L'utilisateur est créé dans l'application et ses informations sont enregistrées en base de données. Sa structure de dossiers est sauvegardée sur l'IPFS et les CID sont récupérés par l'application. Les CIDs sont ensuite sauvegardés dans la base de données.

type *serveless*[11] [10, 41, 6].

(d) **Persistance de données de transactions :** Il convient de déterminer la manière dont une transaction de paiement confirmée par un utilisateur va être persistée sur le réseau. Outre la garantie de sécurité des données des utilisateurs afin que personne ne puisse les lire et les utiliser, si l'utilisateur Bob possède 1000€ et qu'il envoie 100€ à Paul qui possède 0€, il faut que le montant initial de chaque utilisateur et que leur montant final soit enregistré et puisse être mis à jour.

11. Preethi Kasireddy, "The architecture of Web 3.0 application", `https://www.preethikasireddy.com/post/the-architecture-of-a-web-3-0-application` (22/09/2021) (Date d'accès 12/06/2022)



(a) Architecture de l'application partiellement décentralisée

(b) Architecture complètement décentralisée

FIGURE 3.13 – Comparaison des interactions entre l'architecture de l'application en cours de développement et une architecture complètement décentralisée dans laquelle les interactions se font directement entre l'application cliente et l'IPFS à l'aide des bases de données distribuée

Les informations privées utilisées pour les utilisateurs dans l'application peuvent être traitées de plusieurs manières :

i. Dans le cas de l'utilisation d'une base de données relationnelle pour une application partiellement décentralisée, il convient d'enregistrer les informations de l'utilisateur dans une entité de type *User* (table "user"), les informations de ses comptes bancaire dans une entité *Account* (table "account"). Les données persistées sont reliées à ces structures, comme représenté dans la figure 3.14. Cependant, de la même manière que pour les demandes de paiement 2, utiliser l'IPFS

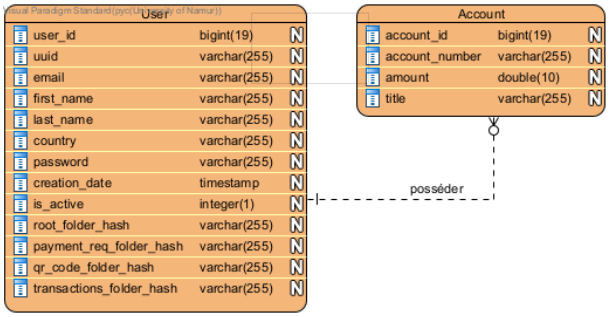


FIGURE 3.14 – Entité utilisateur (*User*) reliée à l'entité compte bancaire (*Account*) dans l'application

partiellement pour enregistrer des données de transactions et des QR codes est une solution qui multiplie les moyens de persistance de données et augmente la quantité de systèmes intriqués. Il convient de choisir entre l'utilisation d'une base de données et l'IPFS pour les transactions., car le premier sert à centraliser l'information et l'autre sert à la décentraliser. Enregistrer les données dans un répertoire « transactions » dans le réseau IPFS comme le montre la figure 3.10 et sauvegarder la référence du dossier (CID) dans l'entité utilisateur est la continuité de la solution choisie précédemment. Les données restent privées car elles sont accessibles depuis des nœuds de confiance.

ii. Utiliser une base de donnée distribuée reliée à l'IPFS (OrbitDB, AvionDB, etc.) comme discuté précédemment (voir 2b) pour les solutions complètement décentralisées. Il convient cependant de s'assurer que les données sont sécurisées et qu'elles sont donc chiffrées lorsqu'elle se trouvent sur le réseau IPFS. De plus, les CID doivent être enregistrées sur la *blockchain*.

iii. Utiliser des fichiers chiffrés spécifiques aux utilisateurs en interaction pour augmenter la sécurité des données privées. Cela peut être fait par l'utilisation d'un outil de chiffrement asymétrique des fichiers tel que GPG, comme représenté dans la figure 3.15. Le cas de figure de cet exemple fonctionne si Bob a accès à la clé publique de Paul. Il est également nécessaire que l'on connaisse les destinataires des fichiers avant leur chiffrement [79]. Ce système implique que les données qui doivent être privées soient chiffrées à l'aide de la clé publique de chaque utilisateur devant accéder à ces données. Par exemple, ce cas de figure est applicable si Bob et Paul doivent accéder à un même fichier structuré de tel qu'une transaction de paiement effectuée entre ces deux utilisateurs. Les commandes de GPG permet de chiffrer un fichier pour le rendre accessible à plusieurs utilisateurs (plusieurs clés publiques) .

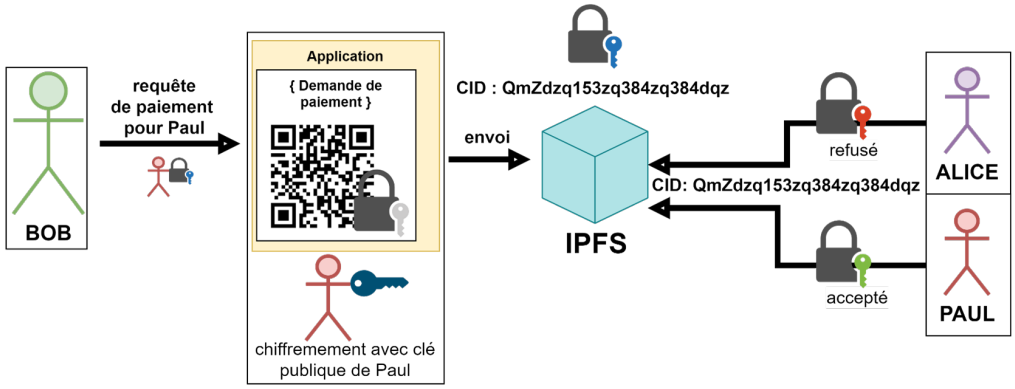


FIGURE 3.15 – Solution de chiffrement asymétrique avec IPFS. L'utilisateur Bob crée une demande de paiement à destination de Paul dans l'application. L'objet est créé et un QR code est généré avec la clé publique de Paul. Le fichier est envoyé par l'application sur IPFS et un CID est généré. Il faut la clé privée de Paul pour déchiffrer le fichier en y accédant sur IPFS à l'aide du CID.

Exemple de commande GPG pour le chiffrement d'un fichier pour plusieurs clés publiques :

```
gpg --encrypt \
  --recipient paul@example.com \
  --recipient bob@example.com \
payment-request.json
```

Néanmoins, cette application est conçue pour permettre un paiement par n'importe quel utilisateur connecté qui désire effectuer un paiement en scannant le QR code à travers l'application. Dans ce cas, l'objet de données récupéré lorsque le QR code est scanné doit pouvoir être déchiffré par tout utilisateur, sans discernement. Il est donc nécessaire que l'application possède une clé de chiffrement qui lui est propre afin de chiffrer les données de l'application de manière globale lorsque ses fonctions d'échange de données de l'IPFS sont utilisées. Cette dernière configuration est applicable pour une application serveur hébergée sur des serveurs privés (nœuds de confiance), mais pas pour un contexte d'application distribuée sans nœuds de confiance.

iv. Utiliser un magasin de clés (*keystore*) [12] [13] chiffré, directement chargé sur le système IPFS. Il peut être mis à jour par l'application à l'aide d'une clé privée et il conserve les informations des utilisateurs comme une base de données, mais sous le format d'un fichier qui peut être modifié par l'application. Les informations des utilisateurs s'y trouvent et sont donc modifiables. Un couplage du système avec une *blockchain* peut être utilisée pour tracer la version du fichier le plus à jour.

v. Utiliser des outils externes dédiés au Web 3.0 afin d'enregistrer de façon immuable des transactions sur la *blockchain*, tout en validant les transactions à l'aide de contrats intelligents.
En l'occurrence, une application distribuée (DApp) combinant les technologies ci-dessous permet de distribuer complètement une application à l'aide de la *blockchain* Ethereum. Elle garantit que les transactions sont réalisées de façon sécurisée, traçable et transparente, directement entre l'utilisateur et la DApp :
— web3.js [14] permettant la communication d'une application avec les écosystèmes Ethereum par l'intermédiaire de contrats intelligents ;
— Truffle [15] pour créer des contrats intelligents Ethereum ;
— Ganache [16] pour développer, déployer et tester les contrats intelligents sur une *blockchain* telle qu'Ethereum ;

12. Protocol Labs, "Use an existing private key", `https://docs.ipfs.io/how-to/use-existing-private-key/` (2022) (Date d'accès 12/06/2022)
13. IPFS Community, "go-ipfs-keystore", `https://github.com/ipfs/go-ipfs-keystore` (13/06/2022) (Date d'accès 14/06/2022)
14. Chain Safe Community,"web3.js - Ethereum JavaScript API", `https://web3js.readthedocs.io/en/v1.7.4/getting-started.html` (2022) (Date d'accès 15/06/2022)
15. Truffle Community, "Truffle, Smart Contracts Made Sweeter",`https://trufflesuite.com/docs/truffle/` (15/06/2022) (Date d'accès 16/06/2022)
16. Truffle Community, "Ganache, One click blockchain", `https://trufflesuite.com/docs/ganache/` (2021) (Date d'accès 16/06/2022)

— un portefeuille centralisé tel que MetaMask (voir 2(d)v) pour signer des contrats intelligents (pour la *blockchain* Ethereum) [10, 85].

L'utilisation d'un portefeuille connecté à l'application distribuée à l'aide d'une API permet d'externaliser les informations d'utilisateurs destinées au web distribué dans un outil multiplateforme relié par la bibliothèque web3. Un utilisateur connecté à cet outil à l'aide de sa clé privée, a la possibilité de partager sa clé publique avec les applications qu'il désire afin de garantir son identité, tout en restant anonyme. MetaMask [17] [18] est une application de portefeuille de cryptoactifs interagissant avec la *blockchain* Ethereum. Elle est accessible sous la forme d'une application mobile ou d'une extension de navigateur. Les utilisateurs peuvent y stocker et gérer des clés de leurs comptes. Ils peuvent diffuser des transactions, envoyer et recevoir des cryptoactifs ou des jetons basés sur Ethereum. Les développeurs d'applications décentralisées réalisent une connexion entre MetaMask et leurs applications décentralisées afin que les utilisateurs interagissent avec elles pour s'authentifier [85] [19]. Cela est possible avec l'utilisation de contrats intelligents et de bibliothèques JavaScript telles que Web3.js ou Ether.js.

L'utilisation des contrats intelligents permet d'appliquer la logique métier des transactions de la DApp sans nécessiter l'intervention d'un serveur applicatif intermédiaire (application *serverless*). Ils permettent donc aux utilisateurs d'interagir directement avec l'application utilisatrice, tout en communiquant avec la *blockchain*. [10]

La figure 3.16 présente l'architecture d'une application distribuée avec l'IPFS utilisant des technologies Web 3.0.

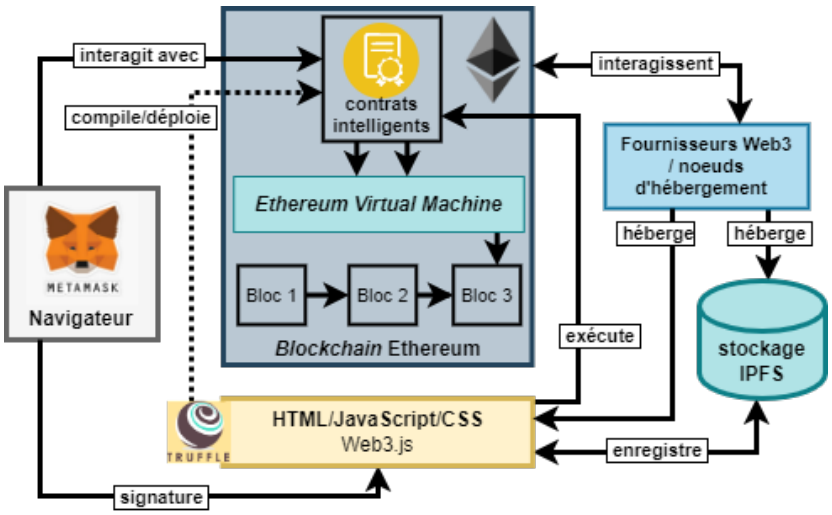


FIGURE 3.16 – Architecture d'une application *serveless* distribuée utilisant les contrats intelligents

vi. Utiliser un système proposant une API qui utilise des standards des identités décentralisées (DID) W3C [20] pour le Web 3.0, tel que Ceramic (Ceramic 3ID [21]), un protocole public sans autorisation et open source qui est couplé à l'IPFS. Le Ceramic Network fournit un réseau global décentralisé et sécurisé qui peut être couplé à l'IPFS, Filecoin ou Amazon S3. 3ID est un SDK d'authentification des utilisateurs se basant sur le protocole d'identité décentralisée IDX [22], qui permet à un DID d'être accessible à partir d'une table d'utilisateurs multiplateforme décentralisée [23]. 3ID peut être associé à un portefeuille de cryptoactifs pour le web distribué tel que MetaMask (voir 2(d)v) [10]. Il permet l'utilisation de données privées en interaction

17. MetaMask Community, "A crypto wallet & gateway to blockchain apps", https://MetaMask.io/ (2022) (Date d'accès 16/06/2022)
18. MetaMask Community, "MetaMask Documentation, Why MetaMask", https://docs.MetaMask.io/guide/#why-MetaMask (14/01/2022) (Date d'accès 16/06/2022)
19. Richard MacManus, "Web3 Architecture and How It Compares to Traditional Web Apps", https://thenewstack.io/web3-architecture-and-how-it-compares-to-traditional-web-apps/ (04/10/2021) (Date d'accès 16/06/2022)
20. W3C Community, "W3C Proposed Recommendation, Decentralized Identifiers (DIDs) v1.0", https://www.w3.org/TR/did-core/ (03/08/2021) (Date d'accès 27/07/2022)
21. Michel Sena, "What is 3ID Connect ?", https://blog.ceramic.network/what-is-3id-connect/ (25/11/2020) (Date d'accès 27/07/2022)
22. IDX Community, "IDX Developers, Overview", https://developers.idx.xyz/learn/overview/ (19/04/2022) (Date d'accès 25/06/2022)
23. Ceramic Developers, "Ceramic Developers, Application Protocol Standards, Account-Based Stream Index", https://developers.ceramic.network/docs/advanced/standards/application-protocols/identity-index/ (2022) (Date d'accès 25/06/2022)

avec le Web 3.0, sans nécessiter de relier des données signées à la *blockchain*[24] [25]. En effet, ajouter une signature au-dessus des données et conserver l'enregistrement de signature sur la *blockchain* telle qu'Ethereum est coûteux[26]. Le stockage hors-chaîne combinant l'utilisation du Ceramic Network, IPFS et la *blockchain* Ethereum peut aider à décentraliser complètement une application avec des coûts réduits. [79]

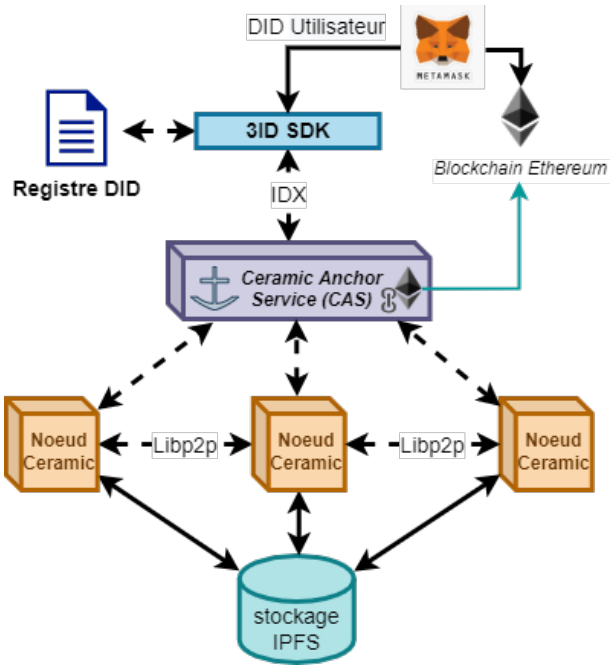


FIGURE 3.17 – Architecture utilisant Ceramic Network. Les données sont enregistrées hors-chaîne sur le réseau IPFS depuis des nœuds Ceramic communiquant à l'aide de Libp2p. Le service d'ancrage Ceramic, relié à la *blockchain* Ethereum, permet l'utilisation de la technologie IDX au travers du SDK 3ID pour intégrer un registre DID. Un utilisateur d'un portefeuille de cryptoactif, tel que MetaMask (voir 2(d)v) connecté à la *blockchain* Ethereum, peut signer des contrats intelligents afin de valider une transaction d'une application distribuée hébergée par des nœuds Ceramic (publiée sur IPFS).

(e) **Suppression des données du réseau :** IPFS possède un système de *garbage collection* qui collecte et supprime des fichiers de manière régulière lorsqu'ils ne sont pas « épinglés » (système de *pinning*). Il est nécessaire de savoir quels objets doivent persister pour une durée précise, voire définitivement dans l'application et lesquels doivent pouvoir être supprimés. Il ne faudrait pas que des informations soient inutilement conservées, et donc involontairement consultables, ou dans le cas contraire, qu'elles soient supprimées alors qu'elles sont toujours utiles pour l'application. La figure 3.18 présente le menu d'épinglage de l'IPFS. Les figures 3.19a et 3.19 présentent les formulaires d'intégration des services d'épinglage pour un nœud IPFS.

Il est possible de configurer les propriétés du système de *garbage collection* d'IPFS depuis un nœud dans un fichier de configurations (également accessible depuis les paramètres de l'interface graphique)[28] [29].

Les propriétés de configuration de l'IPFS concernées sont :

— Option d'activation/désactivation du *garbage collector* ;
— La fréquence à laquelle le *garbage collector* s'exécute (GCPeriod) : 1 heure par défaut ;
— *StorageGCWatermark* : Détermine un pourcentage de stockage maximum qui déclenche l'activation du *garbage collector* ;
— *StorageMax* : détermine la capacité de stockage maximale dans un nœud (10Go par défaut).

Il convient de s'assurer des types de données qui sont utiles à garder (historique des transactions) et celles qui ne le sont pas (QR codes et demandes de transactions avant validation de paiement). Il est envisageable d'utiliser des *blockchains* afin de persister les données utiles de manière immuable

24. Ceramic Developers, "Ceramic Developers, Advanced Tech Overview", https://developers.ceramic.network/learn/advanced/overview/ (2022) (Date d'accès 26/06/2022)
25. Morgan Phuc, "Ceramic Network : Les flux de données décentralisées du Web3", https://journalducoin.com/analyses/ceramic-network-flux-donnees-decentralisees-web3/ (31/01/2022) (Date d'accès 26/06/2022)
26. Joel Thorstensson, "How to store signed and encrypted data on IPFS", https://blog.ceramic.network/how-to-store-signed-and-encrypted-data-on-ipfs/ (19/11/2020) (Date d'accès 08/06/2022)
28. Protocol Labs, "IPFS Pinning Service API (1.0.0)", https://ipfs.github.io/pinning-services-api-spec/ (2022) (Date d'accès 04/06/2022)
29. Protocol Labs, "The Kubo config file", https://github.com/ipfs/go-ipfs/blob/master/docs/config.md (2022) (Date d'accès 04/06/2022)

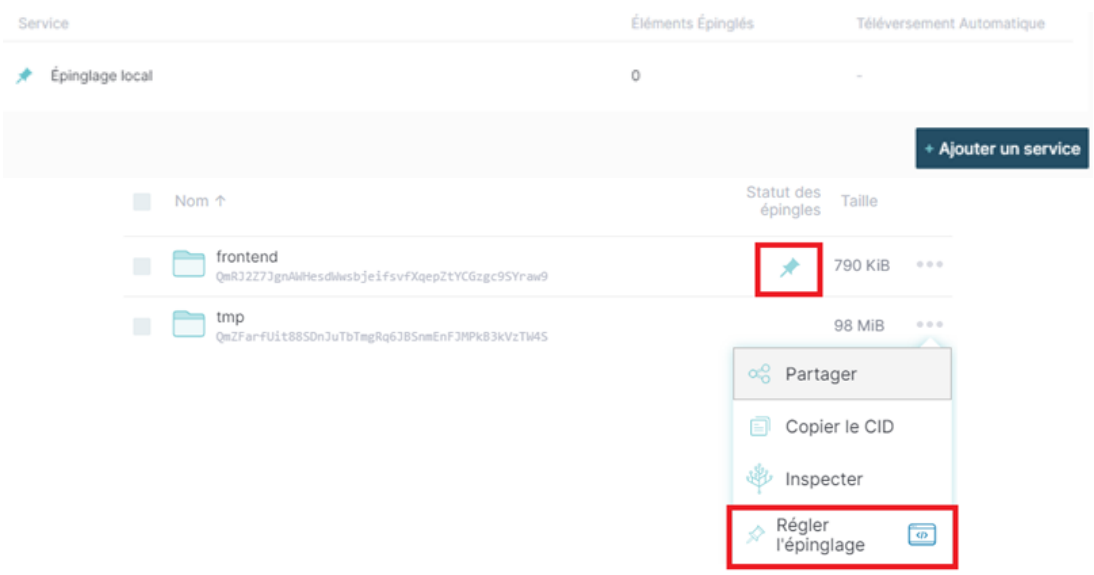


FIGURE 3.18 – Système d'épinglage accessible depuis le menu de l'interface graphique d'IPFS. Le nombre d'éléments épinglés sont répertoriés et répartis sur différents services d'épinglage

(a) Configuration d'un service d'épinglage IPFS (b) Configuration d'un service d'épinglage Pinata

FIGURE 3.19 – Un à plusieurs services d'épinglage peuvent être ajoutés depuis un nœud IPFS. Cela peut être configuré pour un nœud IPFS hébergé, ou bien avec un fournisseur de service d'épinglage tel que Pinata. [27]

et de pouvoir les retracer par la suite.

En complément de la proposition de changer la référence des dossiers lors de leurs mises à jour (voir 3.11), comme représenté sur la figure 3.20, il convient de signaler au système d'épinglage IPFS qu'il faut épingler (*pin*) le dossier possédant la nouvelle référence, ainsi que de supprimer (*unpin*) l'épinglage de l'ancienne structure de dossiers. Cela permet de toujours conserver le dossier utilisateur le plus récent. Ce principe peut être répercuté pour tout type d'objet dans l'application.

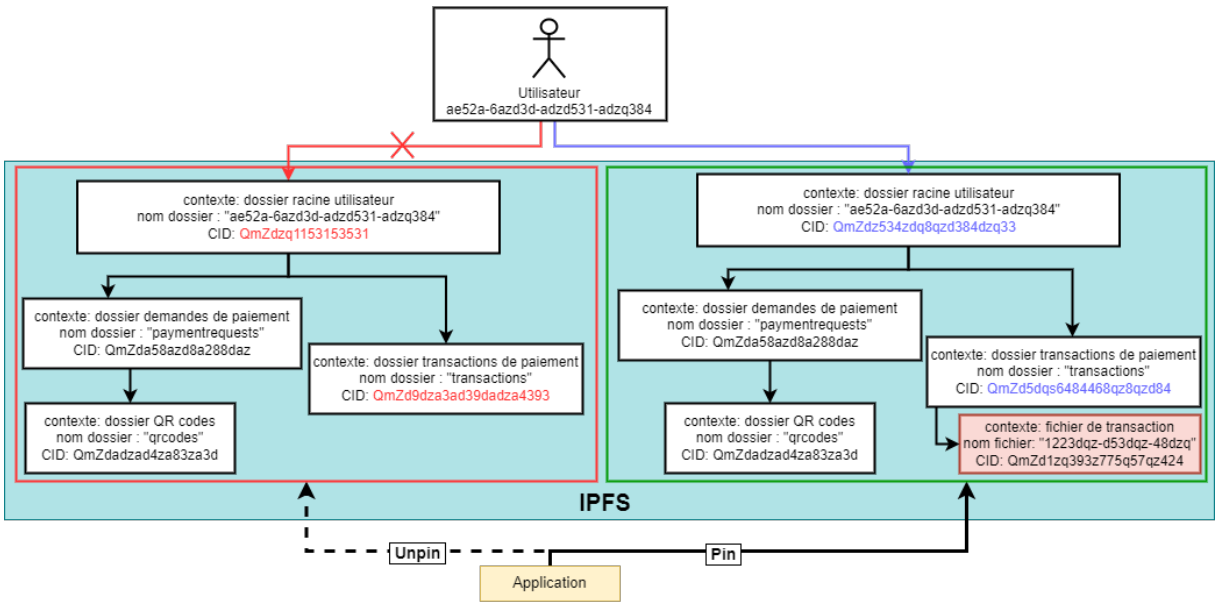


FIGURE 3.20 – Remplacement de l'épinglage du CID d'une structure par une nouvelle structure afin que le *Garbage Collector* puisse supprimer l'ancienne et ne puisse pas supprimer la nouvelle.

3.6.3 Description des choix de résolution des contraintes identifiées

Chiffrement des objets (voir 1)

Pour l'architecture client-serveur de l'application partiellement décentralisée, il est considéré que le code Java est destiné à être déployé sur des serveurs officiant en tant que nœuds de confiance avec des configurations d'accès privées. Une clé privée est donc enregistrée dans les configurations de l'application serveur afin de permettre de chiffrer et de déchiffrer des objets de données avant de les échanger avec l'IPFS. Ces configurations ne sont pas destinées à être distribuées sur des nœuds non fiables. Des méthodes statiques sont donc créées dans l'application Java pour le chiffrement et le déchiffrement des données échangées (chiffrement symétrique AES-128). Les objets de données sont alors chiffrés avant chaque envoi et déchiffrés après chaque récupération du réseau. De cette manière, en accédant à une adresse de contenu, les fichiers de données peuvent toujours être téléchargés mais ils sont illisibles sans utiliser la clé privée utilisée au travers de la fonction de déchiffrement. D'autres types de fonctions de chiffrement peuvent être utilisés car l'IPFS ne dépend pas du type de chiffrement des objets enregistrés.
Enfin, pour une application serveur, des outils tels que Jasypt peuvent être ajoutés en complément pour chiffrer les configurations.

Mise à jour des données des dossiers de l'utilisateur (voir 2)

Un utilisateur de l'application doit pouvoir accéder à des données qui lui appartiennent sur le réseau lors d'une action d'échange de données avec l'IPFS. Il s'avère que jusqu'à cet instant du développement, chaque fichier de données chiffré qui est ajouté au réseau avec l'utilisation de la méthode *IPFSConfig.IPFS.add(data)* de l'API Java http Client par un utilisateur crée un nouveau DAG de Merkle. Ce dernier n'est pas relié à un arbre plus grand qui permettrait de lister et de tracer (voir 3.11 tous les fichiers d'un dossier personnel global, qui comprend les fichiers de transactions d'un utilisateur.

Un utilisateur de l'application possède un identifiant unique (UUID). Cet identifiant est récupéré depuis l'entité *User* grâce au jeton d'authentification de l'utilisateur. Il est alors utilisé pour créer un dossier local sur la machine courante (dossier racine) identifié par cette référence unique dans l'application lors de la création du compte dans l'application. Ce dossier racine est subdivisé en sous-répertoires (voir 3.10) envoyés sur le réseau afin qu'un utilisateur connecté puisse mettre à jour des données reliées à un DAG de Merkle sur le réseau :

- ***paymentrequest*** : contient les demandes de paiement introduites par l'utilisateur ;
- ***paymentrequest/qrcodes*** : contient les QR codes associés à la demande de paiement ;
- ***transactions*** : contient les traces des transactions de paiement réalisées entre deux utilisateurs ;

Lors de l'ajout de cette arborescence de dossiers au réseau (méthode *uploadToNode*) (voir 3.6.1, l'IPFS retourne les CIDs des objets ajoutés. Par facilité, ces hachages sont associés à l'objet utilisateur persisté hors de l'IPFS (voir 3.12). De cette manière, les hachages peuvent-être utilisés pour visualiser le contenu de chaque dossier (méthode *downloadFromNode*) (voir3.6.1).

Plusieurs services web sont également créés afin de pouvoir lister les fichiers de chaque dossier à l'aide de la méthode *IPFSConfig.ipfs.ls(Multishash)*. Ainsi, le développement d'une page dans l'application cliente permet de lister tous les documents de l'utilisateur dans une version mise à jour dès qu'on accède à cette page. Avec l'utilisation de ce service, si l'on sélectionne un élément de la liste, le contenu de l'objet associé à sa référence est visualisé dans l'application.

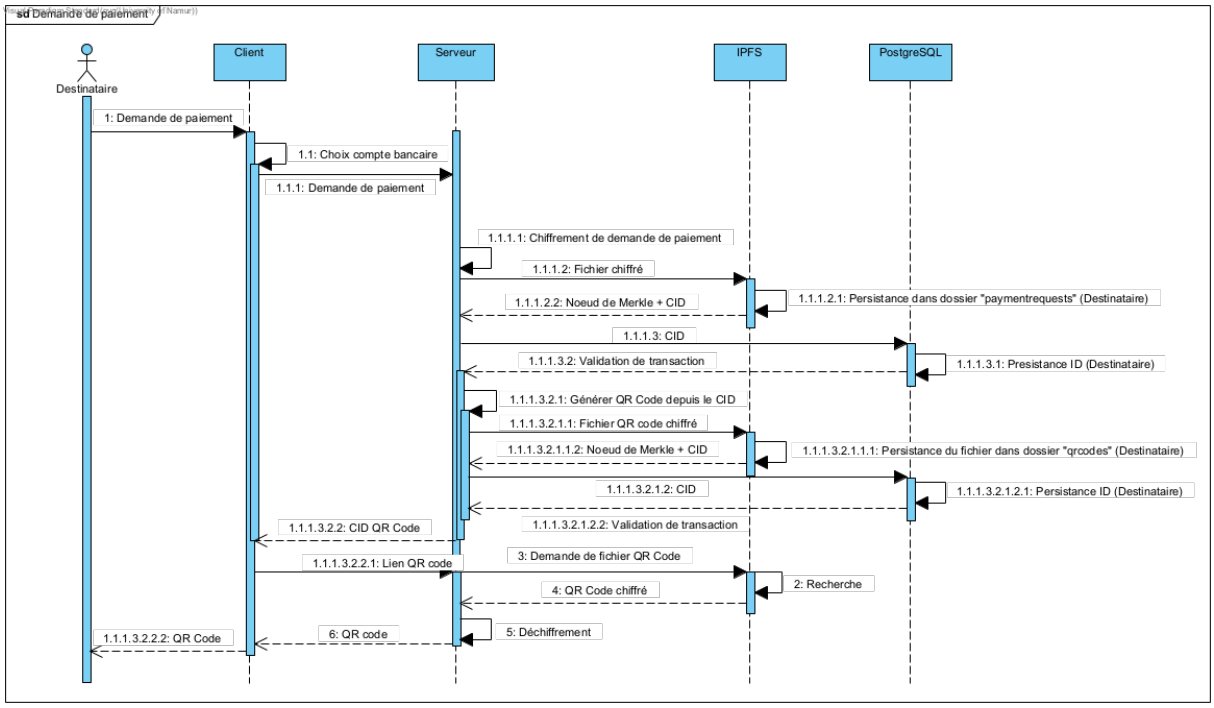


FIGURE 3.21 – Diagramme de séquence d'une demande de paiement.

A cette étape du développement, plusieurs problèmes critiques liés à l'API Java sont rencontrés et décrits dans la partie discussion 4.3.

De plus, étant donné qu'il est uniquement possible d'ajouter des données immuables sur le réseau IPFS, les données qui y sont enregistrées ne peuvent pas y être modifiées. Seules les références des objets ajoutés peuvent donc être modifiées dans l'application utilisatrice lors de la mise à jour d'un objet.
Ainsi, comme représenté à la figure 3.11, pour un dossier (ROOT) contenant plusieurs sous-dossiers (A, B) et fichiers (F1, F2), il convient alors de recréer de nouveaux objets à chaque modification du dossier global d'un fichier échangé avec le réseau. Il faut ensuite mettre à jour les liens référençant l'objet modifié avec son dossier par IPNS. La méthode *IPFSConfig.ipfs.object.patch(...)* (dépréciée, voir 4.3) peut créer une nouvelle référence (CID) depuis un DAG d'un objet vers un nouveau DAG avec une copie de l'objet précédent. De cette manière, pour un dossier ROOT d'un utilisateur a un certain CID... ROOT1 et deux sous-dossiers vides A et B, en ajoutant un fichier F1 dans un sous dossier, il faut d'abord ajouter le fichier seul sur le réseau (méthode *IPFSConfig.IPFS.add(F1)*, et ensuite utiliser *IPFSConfig.IPFS.patch(hash_dossier_cible_A, "add-link", (byte[]) F1 , "nom_fichier", hash_fichier_ajouté_F1)* pour créer un lien dans un nouveau DAG entre le CID du dossier A et ce fichier F1.
Ainsi, le dossier A (CID... A1) a été copié depuis son dossier d'origine ROOT dans un nouvel objet associé à un nouveau DAG (CID... A2) , et cela a lié le CID du fichier F1 à ce DAG(CID... A2) . Le DAG(CID... A2) n'est cependant pas lié à son dossier parent supposé (ROOT). Il faut alors utiliser la méthode *IPFSConfig.IPFS.object.patch(hash_dossier_parent_ROOT, "add-link", null, "nom_dossier_A", hash_dossier_A)* afin de créer un nouveau DAG (CID... ROOT2) avec le dossier parent ROOT, son dossier enfant ROOT.A et le fichier enfant A.F1. Cette méthodologie d'ajout de lien *patch(... , "add-link", ...)* doit être répétée récursivement pour chaque niveau de dossiers ajoutés dans l'arborescence des fichiers. Allant du plus bas niveau au plus haut niveau de l'arborescence de fichiers. A cette fin, la méthode *uploadToNode* (voir 3.6.1) est adaptée en une nouvelle méthode *updateNode* qui prend en compte ces modifications :

- **input :**
  - Un fichier (type MultipartFile) à envoyer sur le réseau ;
  - le CID du dossier cible ;
  - le nom du fichier composé de l'UUID de la demande de paiement/transaction afin de l'identifier de

manière unique et cohérente.
— **output :** une paire de valeurs (type *Pair<String, String>*) qui comprend :
  — Le CID du fichier ajouté ;
  — Le CID du dossier conteneur du fichier.

Immuabilité des objets

Comme représenté sur la figure 3.11, l'ancien arbre de l'objet CID... ROOT1 est recopié vers une nouvelle version CID... ROOT2. Le répertoire est toujours consultable dans son état précédent sur le réseau alors que sa référence n'est plus reliée à l'entité utilisateur. Cela reste donc d'application si les données de l'utilisateur sont supprimées en cascade de l'application.

Les données ajoutées au réseau IPFS sont susceptibles de ne pas disparaître du réseau. Si un nœud y accède et les distribue à son tour, elles peuvent persister et rester disponibles sur le réseau malgré leur suppression ou leur mise hors service sur un nœud d'hébergement initialement unique.

En conséquence, dans le cas d'un téléchargement par un autre nœud, la conservation des CIDs des répertoires et des fichiers supprimés donne la possibilité de toujours accéder aux fichiers même s'ils ont été supprimés du nœud d'hébergement initial [30].

Un lien est créé entre une demande de paiement et son propriétaire. Cette requête de paiement est encapsulée dans un fichier qui est dans un répertoire propre à l'utilisateur sur le réseau. Cependant, si deux utilisateurs sont liés à un fichier (lors de la transaction de paiement par exemple), la transaction doit être référencée de manière unique pour les deux utilisateurs. Le fichier de transaction sera unique sur l'IPFS (un CID unique), mais il devra être référencé dans les dossiers « transactions » de deux utilisateurs différents qui ont conclu la transaction.

Transaction de paiement

Dans l'application, une transaction de paiement peut être effectuée d'un utilisateur Bob vers un utilisateur Paul lorsque Bob scanne le QR code généré par Paul après l'encodage d'une demande de paiement vers son compte bancaire identifié par un UUID unique. Le QR code lu par l'application correspond à un CID, qui permet de récupérer les données de la demande depuis le réseau IPFS (en utilisant la méthode *downloadFromNode* pour déchiffrer l'objet associé). Comme représenté sur la figure 3.22, les données contiennent notamment le numéro de compte bancaire cible du paiement vers Paul, ainsi que le montant à payer par l'utilisateur qui scanne le QR code. Tout utilisateur qui scanne ce QR code avec l'application peut effectuer un paiement.

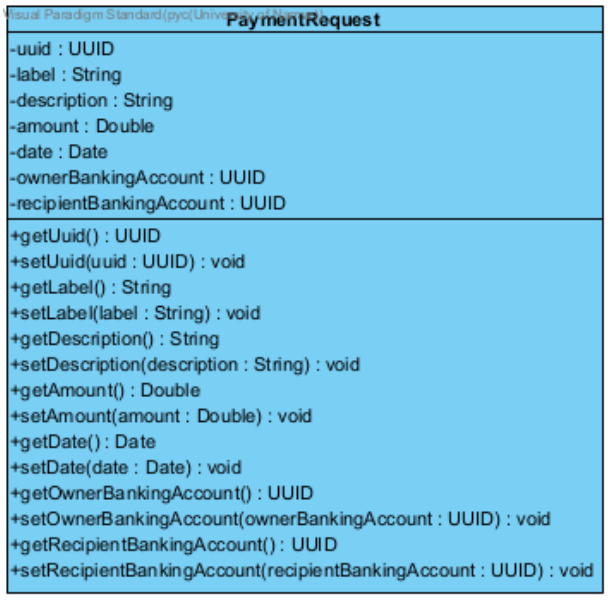


FIGURE 3.22 – Diagramme de classe de l'entité liée à une demande de paiement.

30. Hacker News, "One of the biggest challenges with IPFS in my mind is the lack of a story around how to delete content", https://news.ycombinator.com/item?id=19649893 (04/2019) (Date d'accès 14/04/2022)

La figure 3.3 représente le déroulement de la transaction de paiement. L'utilisateur Bob sélectionne le compte bancaire qu'il souhaite utiliser pour effectuer le paiement. Si le montant sur ce compte est suffisant, la transaction sera acceptée par le système et le montant sera transféré de Bob vers Paul. Dans le cas contraire, la transaction sera abandonnée avant l'envoi de fichier de preuve de transaction de paiement sur le réseau IPFS.
Les données des comptes bancaires (entité *BankingAccount*) sont enregistrées en base de données comme c'est le cas pour les données des entités *User*. Cela permet de sécuriser les données d'un système partiellement décentralisé, où seul l'historique des fichiers est enregistré sur l'IPFS.

Cependant, une base de données contient des valeurs modifiables. Il est donc possible d'éditer le montant d'un compte bancaire directement en base de données sans que cela ne soit obligatoirement tracé à l'aide d'un système de journalisation. Et malgré la présence d'un système de journalisation, les journaux peuvent être supprimés par une personne malveillante qui parvient à y accéder. L'utilisation d'un système travaillant avec des données immuables tel qu'IPFS comme stockage hors-chaîne en combinaison d'une *blockchain* permet de tracer toutes ces modifications de bout en bout et d'en garder une trace immuable.

Garbage Collection

En conséquence de la proposition d'épinglage et de suppression de l'épinglage de données lors du changement de référence d'objets de données sur le réseau IPFS (voir figure 3.20), il est nécessaire d'adapter la méthode updateNode afin d'y ajouter l'utilisation des méthodes *IPFSConfig.ipfs.pin.add / rm / update* proposées par l'API Java IPFS.

3.6.4 Distribution des données privées de l'application

Jusqu'à cette étape, dans l'application, un utilisateur se connecte à l'aide de son adresse email et de son mot de passe avec une base de données. Une entité utilisateur (*User*) (voir 3.14) permet d'enregistrer les données privées d'un individu afin qu'il puisse accéder à un ou plusieurs comptes bancaires. L'entité *User* conserve également les références vers les CID des dossiers associés à la structure des fichiers de transactions de l'utilisateur sur le réseau IPFS. Ces informations sont persistées en base de données afin de pouvoir sécuriser ses informations de manière centralisée, ainsi que de manière à pouvoir tracer simplement les CID des dossiers de l'utilisateur. De plus, sur l'IPFS, un dossier racine pour les fichiers de l'utilisateur est créé lors de l'enregistrement de l'utilisateur. Il est différencié par un identifiant unique propre à ce dernier (UUID attribué de manière aléatoire) (voir figure 3.10).

Cependant, l'utilisation de la combinaison d'une base de données centralisée et de l'IPFS comme systèmes de stockage pour des données privées n'est pas pertinent. Les données (privées ou non) ont un intérêt d'être décentralisées si elles ne sont pas centralisées, un choix s'impose donc entre les deux solutions.
Pour distribuer les données de l'application, il convient alors de supprimer la dépendance à la base de données centralisée. L'entité utilisateur *User* ne devrait plus y être persistée, mais doit rester manipulable à tout moment depuis l'application interagissant avec l'IPFS pour que l'application continue de fonctionner.
Une solution est d'enregistrer toutes les données de l'utilisateur qui ont été mises en base de données dans un fichier structuré, tel qu'un fichier JSON ; un CSV ; un XML ; un objet sérialisé ; etc. chiffré à l'aide d'une clé privée lors de la création du compte de l'utilisateur. L'objet utilisateur est enregistré sur l'IPFS comme cela a été fait pour les autres fichiers dans l'application. Il reste illisible sans la clé de déchiffrement combinée à la fonction de chiffrement choisie, voire sans le cassage de l'algorithme de déchiffrement.

Plusieurs contraintes sont donc identifiées pour que cela soit applicable à l'application :

1. Il est important d'assurer la présence d'une sécurité renforcée pour que la clé de déchiffrement des fichiers de données ne soit récupérée ou trouvée, sans dépendre de l'*obfuscation* que procure l'application serveur.
2. Etant donné que l'entité utilisateur est régulièrement modifiée afin d'adapter les hachages cryptographiques des dossiers de l'utilisateur, il faut que le fichier puisse être récupéré et modifié à son tour. Toute modification du fichier structuré entraîne la création d'un nouvel objet (immuable) de données sur le réseau, et donc une nouvelle référence vers un nœud du Merkle DAG.
   De cette manière, pour que les données des utilisateurs soient distribuées sur l'IPFS et qu'ils puissent continuer à se connecter à l'application, il est une nouvelle fois nécessaire d'indexer le CID de l'objet *User* pour permettre à l'application de retrouver et d'utiliser les données de l'utilisateur courant les plus à jour sur le réseau.
3. L'utilisateur doit être retrouvé par l'application sur l'IPFS pour qu'il puisse s'authentifier en tant qu'un utilisateur valide. Par exemple, un cas d'une utilisation d'une connexion combinant un moyen d'authentification tel qu'une adresse email et un mot de passe, mais dont les données d'authentification

seraient stockées sur l'IPFS, nécessiterait la présence d'un registre dans l'application qui permet de retrouver l'association entre l'email de l'utilisateur courant et son adresse de contenu la plus à jour (CID). En effet, l'application doit valider ou invalider une tentative de connexion d'un utilisateur qui entre des informations de connexion valides ou invalides. Dans ce contexte, le fichier disponible sur le réseau IPFS de l'utilisateur devrait alors être lu à travers l'API IPFS, puis déchiffré et le mot de passe devrait être comparé.

Une solution envisagée pour permettre d'interagir avec l'IPFS sans interagir avec une base de données centralisée est d'utiliser d'un système de chiffrement asymétrique tel que fournie par GPG (voir 2(d)iii) afin de chiffrer le fichier de l'utilisateur directement sur IPFS. Une autre possibilité serait l'utilisation d'une clé privée unique/un certificat d'authentification (fournie par un objet physique tel qu'une clé usb d'authentification ; un certificat de sécurité ; un eID ; une empreinte digitale ; etc.). La clé publique de l'utilisateur doit être indexée dans un fichier retrouvé par l'application pour qu'il puisse s'identifier à l'aide de sa clé privée en tant que la personne qu'il prétend être. Il faut également que la référence du fichier qui comprend ses informations les plus à jour soit répertoriée à l'aide d'une *blockchain*. [31]
De la même manière, utiliser un fournisseur de connexion externe pour une authentification partiellement décentralisée est possible. En effet, des API fournies par des fournisseurs de services d'authentification telles que Google, Facebook, eID, Itsme, Keycloak, etc. offrent des solutions de connexion externes respectant les standards OAuth2.0 ou OpenID Connect (OIDC). L'utilisation d'un service de connexion standardisé permet à l'utilisateur de s'authentifier sans dépendre d'une base de données reliée à l'application par l'utilisation d'un *JSON Web Token* (JWT) [32] [33] [50]. Les informations de connexion de l'utilisateur sont enregistrées dans le Jeton d'authentification. La combinaison de la clé privée avec la clé publique de l'utilisateur peut être utilisée pour que l'application récupère les données chiffrées de l'utilisateur sur l'IPFS.
Dans ce cas de figure, l'application utilise l'identifiant utilisé pour retrouver un compte en faisant un calcul à la volée en toute transparence (côté client) sur base de la valeur de la clé privée et dont le résultat, un hachage cryptographique, est un identifiant unique de l'utilisateur. L'identifiant alors indexé dans un fichier – une Table de correspondances (type *HashMap*) – propre à l'application et associé à un CID en tant que paire *<hash, user-cid>*. La figure 3.23 représente ce système. La figure 3.24 représente le diagramme de séquence de ce système destiné à l'application courante. [79]

Cependant, plusieurs problèmes sont identifiés :

— soit ce fichier est répertorié sur l'IPFS, et donc la référence vers un nouveau CID change à chaque modification. Le CID le plus à jour du fichier utilisateur doit être répertorié afin d'être retrouvé par l'application ou un utilisateur ;
— soit le fichier doit être centralisé par un fournisseur externe, ce qui revient à utiliser un système externe comme c'est déjà le cas par l'utilisation d'un système d'authentification OAuth2.0 qui a la possibilité d'intégrer des moyens de connexion à travers des API telles que Google, Facebook, etc. ;
— soit le fichier doit être stocké par tous les nœuds et mis à jour à chaque modification par un utilisateur.

Ce système n'est donc pas auto-suffisant pour fonctionner de manière distribuée sur des nœuds indépendants du réseau. Il convient d'utiliser une *blockchain* afin de pouvoir le mettre en place.

Si les données des utilisateurs sont enregistrées et chiffrées de manière robuste dans un fichier sur l'IPFS, il est nécessaire de pouvoir tracer au moins une référence vers un fichier du réseau depuis l'application. Cela peut être un fichier qui indexe tous les CID à jour des utilisateurs, ou bien plusieurs fichiers. L'utilisation d'une *blockchain* doit permettre d'assurer cette tâche en référençant les CID dans une suite de blocs de données vérifiables. Ces blocs se référencient à l'aide d'une propriété *hash* pour le hachage du bloc courant, et *previousHash* pour le hachage du bloc qui le précède. Dans le cas de figure présenté par la figure 3.23, chaque nouveau bloc de la *blockchain* référencie le CID du fichier/dossier le plus récent. [6] En l'occurrence, comme représenté sur la figure 3.25 le dernier bloc référencie le CID le plus récent de la « Table de correspondances ». [79]

Intégrer un système de connexion décentralisé de type IDX avec 3ID Connect fournit un système d'authentification DID utilisé par Ceramic Network (voir 2(d)vi) à partir d'une connexion à un portefeuille de cryptoactifs. Le portefeuille combiné à IDX permet de fonctionner en tant que *keystore* privé interagissant avec une *blockchain*. 3ID stocke le lien du compte *blockchain* du DID de l'utilisateur dans IDX, qui fournit un moyen de stocker les enregistrements d'identité pour un DID sur le Ceramic Network. Les enregistrements peuvent représenter des liens vers des comptes *blockchain*, ainsi que les données de l'entité utilisateur.

Cet écosystème permet de décentraliser complètement l'application, avec un système de connexion utilisant

31. Coral Health, "Learn to securely share files on the blockchain with IPFS !", `https://mycoralhealth.medium.com/learn-to-securely-share-files-on-the-blockchain-with-ipfs-219ee47df54c` (20/10/2018) (Date d'accès 27/04/2022)
32. Auth0 Inc., "OAuth 2.0 Simplified", `https://auth0.com/docs/` (2022) (Date d'accès 27/04/2022)
33. Gery Ducatel, "Signature and Encryption Options for OAuth 2.0 and OIDC", `https://developer.forgerock.com/docs/platform/how-tos/signature-and-encryption-options-oauth-20-and-oidc-part-2` (2022) (Date d'accès 27/04/2022)

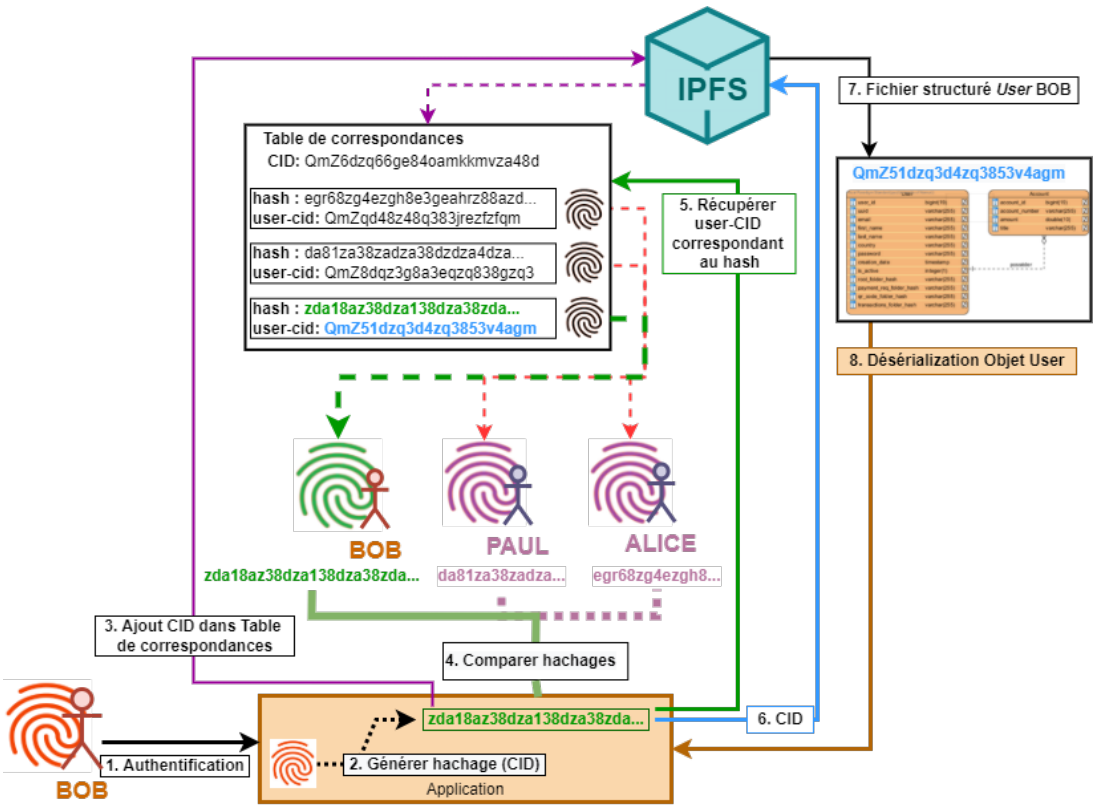


FIGURE 3.23 – Récupération des informations de l'utilisateur sur le réseau IPFS à l'aide d'une table de correspondances : (1) L'utilisateur utilise son certificat de connexion ; (2) Le hachage cryptographique est généré ; (3) S'il n'est pas indexé, le CID généré pour le fichier utilisateur à jour est ajouté à la table des correspondances ; (4) S'il existe, il est retrouvé dans la table des correspondances afin de retrouver le hachage à jour du fichier de l'utilisateur associé au CID ; (5) Le CID de l'utilisateur courant est récupéré parmi la liste des résultats des utilisateurs indexés dans la table de correspondances ; (6) Le CID est envoyé à IPFS afin de retrouver le fichier structuré de l'utilisateur courant ; (7) Le fichier est récupéré pour l'utilisateur ; (8) La structure est désérialisée en objet de type *User* et utilisée dans la session de l'utilisateur courante.

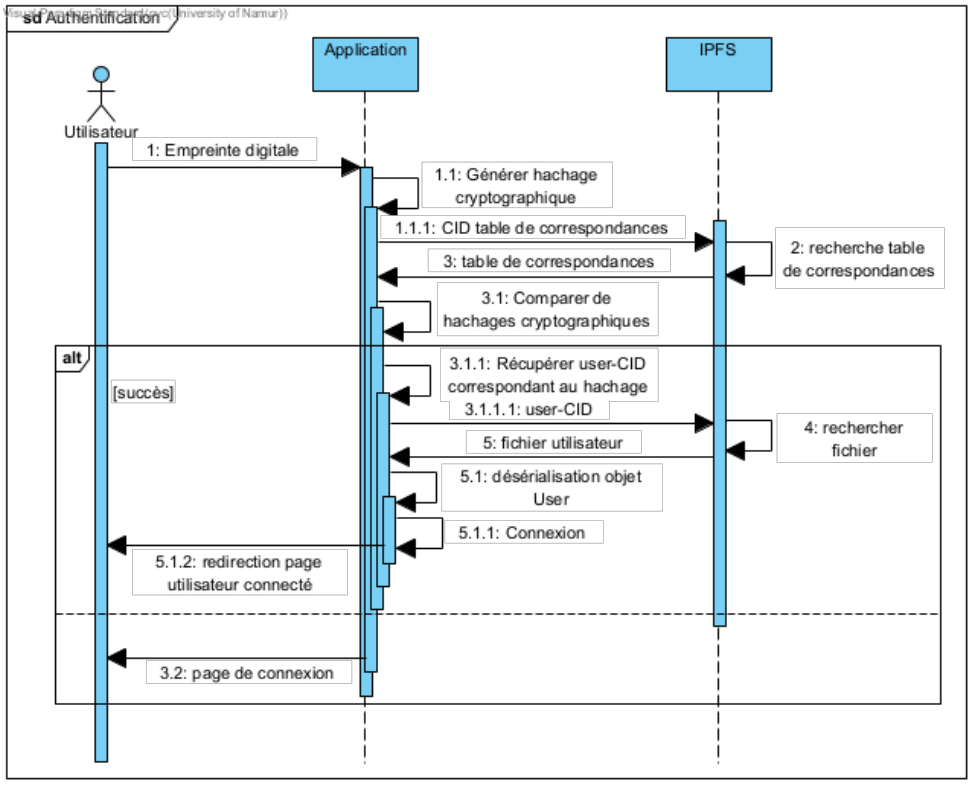


FIGURE 3.24 – Diagramme de séquence d'un système de registre pour IPFS destiné à l'application courante

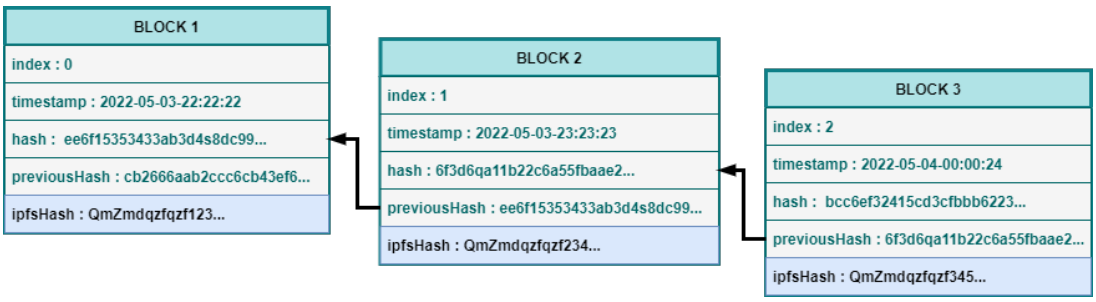


FIGURE 3.25 – Le dernier bloc référencie le CID le plus récent de la « Table de correspondances »

un DID. Il permet la décentralisation des données de l'utilisateur en les conservant dans le portefeuille externe à l'application et au réseau IPFS, ainsi que de garder les informations de connexion hors de tout système de stockage relié au réseau.
Les données conservées dans le portefeuille interagissent avec une *blockchain* et sont vérifiées lors d'interactions avec l'application afin de s'assurer que les données utilisées pour la connexion ou les transactions de paiement n'ont pas été altérées sans avoir été validées par un système de contrat intelligent.

3.6.5 Déploiement de l'application

Pour publier l'application sur l'IPFS, plusieurs solutions sont identifiées :

— **Application cliente sur l'IPFS uniquement et application serveur sur un système de stockage hors IPFS :** Pour une application cliente avec l'utilisation d'un *framework* applicatif JavaScript tel qu'Angular, il convient de suivre plusieurs étapes pour le déploiement sur un nœud du réseau [34] [35] :

1. Le système de routage doit utiliser une configuration '$\{useHash : true\}$' afin d'ajouter un # avant les routes de navigation sur la passerelle IPFS. Cela permet de ne pas obtenir d'erreur 404 lors d'une navigation entre les pages web sur le réseau IPFS.
   De plus, les configurations de l'application cliente doivent pointer vers l'adresse du serveur applicatif. L'utilisation d'une adresse pointant vers un proxy peut être utilisée. Il est également nécessaire que les configurations de sécurité de l'application serveur soient adaptées afin d'accepter des requêtes de la part des nœuds IPFS et de pouvoir communiquer avec le réseau P2P. Accepter l'adresse de l'application déployée sur passerelle IPFS permet de ne pas obtenir d'erreurs de type *Cross-origin resource sharing (CORS Policy)*.
2. Des commandes de *build* de l'application cliente telles que *'npm run build –prod –aot'* permettent de générer un package de fichiers HTML, CSS et JavaScript dans un dossier *'dist/<nom_app_frontend>'*.
3. L'ajout du dossier de l'application cliente générée à l'aide du daemon IPFS (*'ipfs add -r <chemin vers dist/nom_app_frontend>'*, ou bien copie manuelle dans IPFS Desktop permet de mettre l'application en ligne sur le réseau IPFS (partager).
4. Les fichiers de l'application cliente peuvent être épinglés et également être distribués (partagés) sur le réseau.

Lors de l'ajout de fichiers sur le réseau depuis une machine hôte locale (localhost), les fichiers peuvent être retrouvés avec succès à l'aide d'une recherche sur base du CID obtenu en réponse à l'ajout des fichiers depuis le client IPFS faisant office de passerelle http pour l'hébergement. Les fichiers sauvegardés sur la machine hôte de l'instance ipfs en développement se trouvent dans un chemin local $/home/.ipfs/$ sous la forme de blocs de données. Une recherche directe sur la passerelle ipfs à l'aide d'un lien tel que $https://ipfs.io/ipfs/ < CID >$ ne permet pas de récupérer les fichiers. Il convient de partager les fichiers avant de pouvoir y accéder à distance.

— **Application cliente et application serveur déployées sur l'IPFS à l'aide d'une passerelle d'hébergement telle que Textile buckets [24] / Infura [36], gratuit et basé sur Ethereum / Fleek, basé sur Ethereum / etc. pour assurer la distribution des données :** L'outil Java IPFS Maven Deploy requiert la création d'une connexion à l'une des plateformes Textile, ou Fleek préalablement

34. Travis Reeder, "How to Deploy an Unstoppable Angular App to IPFS", https://medium.com/gochain/how-to-deploy-an-unstoppable-angular-app-to-ipfs-c2dabb52e517 (26/03/2019) (Date d'accès 30/05/2022))
35. https ://nolife.cyou/,"How to deploy Angular app to IPFS/IPNS", https://nolife.cyou/how-to-deploy-angular-app-to-ipfs-ipns/ (Date d'accès 28/05/2022)
36. Infura Inc., "Infura, Start building on Ethereum and IPFS", https://docs.infura.io/infura/ (2022) (Date d'accès 17/06/2022)

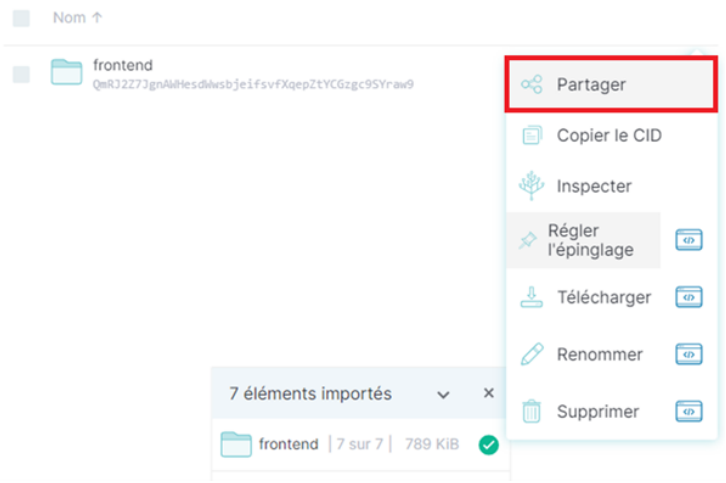


FIGURE 3.26 – Publier des fichiers depuis l'interface graphique

afin d'y déployer du contenu.
Les étapes sont les suivantes :

1. *Build* de l'application Angular ;
2. *Build* de l'application Java avec le package Maven com.github.ericglau.ipfs-deploy-maven-plugin et déploiement sur l'IPFS à l'aide d'une passerelle Textile Buckets ou Fleek [37] :
   (a) **Textile :** intégration à l'aide du Textile Hub CLI [38] ;
   (b) **Plate-forme Fleek :** Intégration à travers le SDK AWS pour Java ; [39]
3. **Filecoin :** intégré à travers l'utilisation de Textile pour l'archivage vers Filecoin [40]

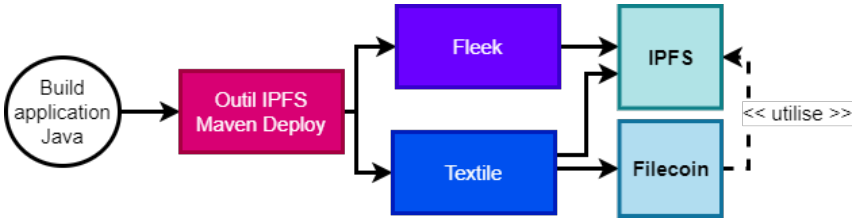


FIGURE 3.27 – Publication d'une application Java sur le réseau IPFS à l'aide de la bibliothèque IPFS-deploy pour Java [41]

Publication du contenu sur des *Clusters* IPFS :

Il est possible de configurer un système pour héberger des nœuds IPFS organisés en *clusters*, comme représenté sur la figure 3.28, qui s'ajoutent aux nœuds décentralisés de l'IPFS. [42]

Le service de *cluster* IPFS [43] permet de distribuer du contenu à l'aide d'une automatisation de l'épinglage et de la réplication du contenu sur un ensemble de nœuds d'un réseau IPFS. Il s'agit d'une extension de l'IPFS sous forme d'un package supplémentaire au système *core* du réseau, qui est utilisable à l'aide d'un système en ligne de commandes (CLI). Tous les systèmes pairs qui l'utilisent pour former le *cluster* ont besoin d'avoir le système IPFS installé et qu'il s'exécute en parallèle du service de *clustering* IPFS.

Le système de clusters peut être configuré au sein du nœud préalablement installé à l'aide du package complémentaire au système *core*. Il convient d'y spécifier les pairs qui le composent et de déployer le service de

37. Protocol Labs, "IPFS Gateway Documentation", https://docs.ipfs.io/concepts/ipfs-gateway/#gateway-types (2022) (Date d'accès 21/06/2022)
38. "Textile Documentation", https://docs.textile.io/threads/
39. Fleek Community, "Fleek Documentation", https://docs.fleek.co/#ipfs-gateway (2020) (Date d'accès 21/05/2022)
40. Protocol Labs, "What is Filecoin", https://docs.filecoin.io/about-filecoin/what-is-filecoin/ (2022) (Date d'accès 23/05/2022)
42. Protocol Labs, "Test Cluster Quickstart", https://ipfscluster.io/documentation/quickstart/ (2022) (Date d'accès 22/05/2022)
43. Ross Bulat, "Using IPFS Cluster Service for Global IPFS Data Persistence", https://rossbulat.medium.com/using-ipfs-cluster-service-for-global-ipfs-data-persistence-69a260a0711c (03/01/2019) (Date d'accès 01/07/2022)

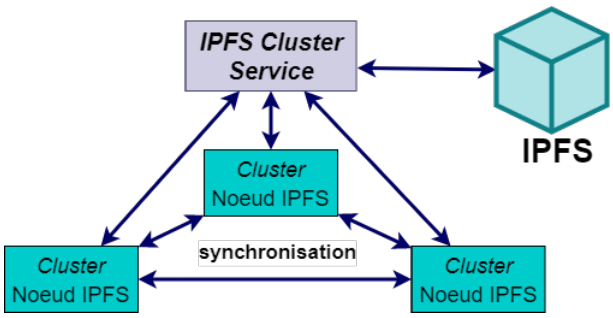


FIGURE 3.28 – Représentation d'un réseau composé de trois Clusters IPFS synchronisés destinés à hérberger du contenu. Ils interagissent avec le réseau IPFS par l'intermédiaire du service de *clustering* IPFS intégré par le *deamon* IPFS

clustering afin qu'il se synchronise avec les nœuds pairs et avec le réseau. Ce système peut être utilisé sur des serveurs qui exécutent des logiciels de déploiement automatique et de répartition de charge. Kubernetes, Ansibe, Jenkins peuvent être configurés de cette manière pour travailler avec le service de *clusters*.

Le système de *clusters* peut également être configuré de manière à introduire manuellement des nœuds IPFS dans un cluster initial afin de déployer des pairs supplémentaires après le déploiement d'un premier nœud.

3.6.6 Mesures des performances de l'IPFS

Mise en place de *clusters* IPFS

Des tests de mesures de performances du réseau IPFS ont été réalisés avec l'outil de tests de charge JMeter [44] [45]. Des requêtes parallèles d'envoi et de réception de fichiers de différentes tailles ont été mesurées sur un réseau composé de plusieurs *clusters* IPFS.

Grâce à cela, les tests de performances ont été réalisés sur les nœuds configurés pour interagir ensemble à travers des conteneurs Docker fonctionnant sur plusieurs ports HTTP, redirigés vers le port HTTP de l'instance globale transitant vers la passerelle IPFS [46]. Une configuration Docker définie dans un fichier docker-compose (YAML) a permis d'obtenir autant de *clusters* que nécessaires à la réalisation des tests. Dans cette configuration, les nœuds clusterisés sont synchronisés et communiquent entre eux automatiquement. La charge du réseau est répartie sur les nœuds pour l'hébergement de données de tests.

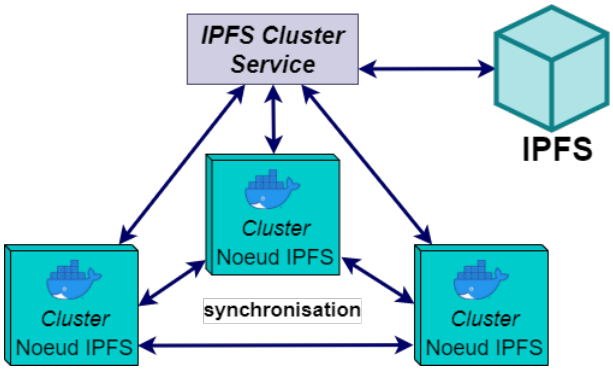


FIGURE 3.29 – Représentation d'un réseau composé de trois Clusters IPFS synchronisés conteneurisés à l'aide de Docker et destinés à hérberger du contenu. Ils interagissent avec le réseau IPFS par l'intermédiaire du service de *clustering* IPFS intégré par le *deamon* IPFS

Les tests ont été réalisés avec une connexion réseau fournissant une bande passante ≈1Gb/s. Les configurations de la machine du conteneur Docker utilisent un processeur Intel Core i7-1165G7 2.8GHz comprenant 4 cœurs et 8 processeurs ; 4 Go d'allocation mémoire RAM ; un espace disque (SSD) disponible de 80 Go.
Les requêtes ont été réalisées dans des conditions identiques et à un même moment. Un nombre de trois *clusters* a été configuré afin de représenter une configuration minimale pour la distribution de données. Etant

44. NaveenKumar Namachivayam, "Performance Testing IPFS Protocol", `https://qainsights.com/performance-testing-ipfs-protocol/` (23/11/2019) (Date d'accès 05/07/2022)
45. Baeldung, "Intro to Performance Testing using JMeter", `https://www.baeldung.com/jmeter` (04/07/2022) (Date d'accès 05/07/2022)
46. Rahasak Labs, "IPFS Cluster with Docker", `https://medium.com/rahasak/ipfs-cluster-with-docker-db2ec20a6cc1` (16/10/2020) (Date d'accès 07/07/2022)

donné que Filecoin distribue des données sur un minimum de trois nœuds afin d'assurer le contenu, cette configuration est représentative d'un contexte minimum réel afin d'étudier le temps de réponse et le débit des données échangées pour un nombre de de 2000 requêtes envoyées par salves de 500 requêtes en parallèle.

3.7 Résultats de la recherche

3.7.1 Résultats du développement

La figure 3.30 représente le fonctionnement de l'application *Proof of Concept* réalisée dans le cadre de la recherche.

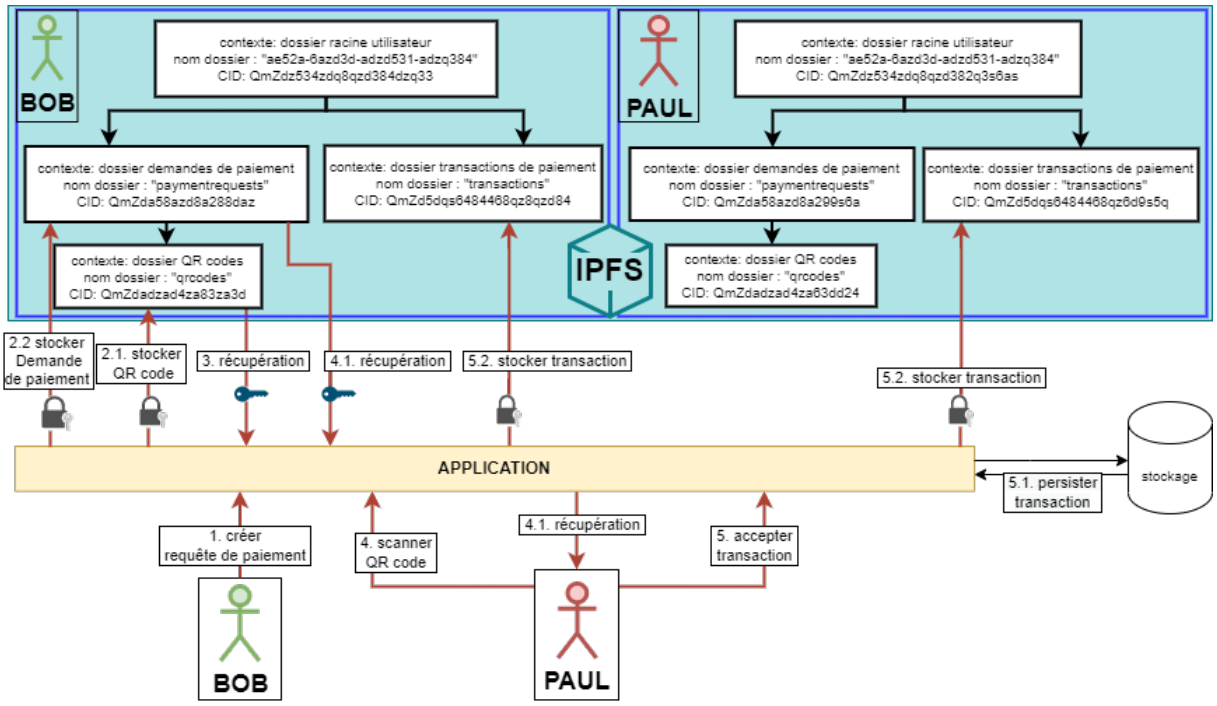


FIGURE 3.30 – Fonctionnement de l'application *Proof of Contept* de paiement bancaire réalisée dans le cadre de la recherche.

L'application réalisée lors de la recherche possède une architecture client-serveur qui utilise IPFS en tant que système de stockage décentralisé, dont le contenu, accessible à l'aide d'une passerelle IPFS, est immuable pour un CID donné. Il a été identifié que ce type d'application peut être hébergée sur des nœuds de confiance qui garantissent l'intégrité des données des configurations permettant de servir l'application. L'application client-serveur développée dans le cadre de la recherche a permis d'aboutir à l'identification de cas d'application de l'IPFS, ainsi qu'à des solutions permettant de mener à bien un projet informatique pour le cas étudié, qu'est la mise en place d'un système de paiement bancaire sécurisé entre deux utilisateurs à l'aide du stockage de données privées. L'hébergement de manière décentralisée pour le déploiement de l'application avec l'IPFS peut être fait à l'aide d'un fournisseur de contenu externe comme le supportent les plateformes Fleek, ou Textile, ou bien à l'aide d'infrastructures hébergées par des nœuds pairs de l'IPFS. Dans le cas d'une application nécessitant une infrastructure offrant un serveur applicatif, ils doivent pouvoir être considérés comme des nœuds de confiance qui hébergent des configurations de l'application *back-end*.

Pour la réalisation d'applications interagissant avec IPFS, des bibliothèques proposées sur le dépôt Github de la communauté IPFS permettent d'étendre le protocole afin de pouvoir l'utiliser à l'aide d'APIs dédiées, dans des langages de programmation variés. Les langages Go et JavaScript sont les langages principalement utilisés par la communauté IPFS, mais d'autres langages, tel que le langage Java, sont également supportés. Des modules disponibles depuis le dépôt Github IPFS proposent également des outils pour l'utilisation de la technologie dans des cadres d'application divers.

Par ailleurs, le système d'épinglage de l'IPFS permet de signaler au *garbage collector* que des données ne doivent pas disparaître, afin de les rendre disponibles de manière permanente depuis un ou plusieurs nœuds actifs.
Dans le cadre de l'hébergement des données d'une application pour sa publication sur le web, le code destiné à être déployé côté client peut être épinglé et ensuite *partagé*, afin de fournir à tous le réseau un accès direct et permanent à une version donnée de l'application. Cette dernière devient alors accessible à l'aide d'un CID utilisé à l'aide d'une passerelle donnant l'accès à l'IPFS.

D'autre part, il a également été identifié que la mise en œuvre d'applications complètement décentralisées est possible à l'aide d'une combinaison d'IPFS et des technologies du Web 3.0, pour la distribution des don-

nées et l'anonymisation des transactions. Le langage JavaScript est approprié à ce type d'infrastructures car il est directement interprété par les navigateurs web, ne nécessitant alors aucun hébergement d'un serveur applicatif, ni l'intervention d'un logiciel de base de données pour l'hébergement des données qui sont enregistrées sur le *filesystem* des nœuds IPFS. Ces applications peuvent être totalement dynamiques afin d'interagir avec les actions des utilisateurs ; elles peuvent enregistrer des données qui forment des hiérarchies de fichiers à l'aide des DAG de Merkle ; elles peuvent utiliser un système de chiffrement pour échanger des fichiers entre utilisateurs sans nécessiter de sauvegarder des données reliées ; dans le cas d'un besoin de traçabilité, elles peuvent être utilisés à l'aides des *blockchains* ; elles peuvent utiliser les identités décentralisées (DID) pour interagir avec des systèmes de données privées d'utilisateurs ; elles peuvent combiner les *blockchains* et des contrats intelligents afin de créer des systèmes transactionnels ; des bases de données distribuées en surcouche d'IPFS telles que OrbitDB ou AvionDB peuvent aider à manipuler les données directement depuis le code de l'application cliente.
En l'occurrence, IPFS peut également héberger du contenu de manière totalement statique (versionné) et publique afin d'agir en tant que réseau de diffusion de contenu (CDN).

De plus lorsqu'un nœud pair consulte du contenu à l'aide d'une recherche à partir d'un CID, le contenu récupéré est mis en cache au sein du nœud afin de rendre le contenu hautement disponible au sein du réseau. Le système est alors renforcé et favorise la résistance à la censure. Le système de stockage IPFS peut donc héberger du contenu sur de nombreux nœuds, qui peuvent être configurés en tant qu'un réseau de *clusters* interagissant ensemble. Ces *clusters IPFS* peuvent être configurés pour être hébergés et servis par des fournisseurs de services d'hébergement web.

3.7.2 Résultats des mesures de performances

Le débit (*throughput*) et le temps de réponse moyen pour des échanges de données avec le réseau IPFS lors de l'envoi de fichiers de différentes tailles (*upload*) et lors de leur réception (*download*) donnent des résultats avec des tendances semblables. Les observations ont été faites sur un total de 2000 résultats de requêtes par fichier (plusieurs fichiers de tailles différentes), appliquées par salves de 500 requêtes en parallèle.
Les résultats permettent d'observer que le débit (Ko/s) diminue fortement pour l'envoi et la réception de données, tandis que le temps de réponse moyen augmente avec une tendance linéaire à mesure que la taille des fichiers de données échangés augmente. De plus, dans le cas de l'envoi de données, le nombre de Ko/s reçus diminue, et le nombre de Ko/s envoyés augmente ; dans le cas de la réception de données, le nombre de Ko/s reçus augmente, et le nombre de Ko/s envoyés diminue.
Les résultats des mesures sont légèrement meilleurs pour l'envoi de données par rapport à la réceptions de données.

Téléchargement directement depuis le réseau								
Taille du fichier téléchargé (Ko)	Nombre de nœuds d'hébergement	Nombre de requêtes totales envoyées	Temps de réponse moyen (ms)	Temps de réponse minimal (ms)	Temps de réponse maximal (ms)	Débit (requêtes/sec)	Ko/sec reçus	Ko/sec envoyés
2	3	2000	1772	7	4038	35,3	84,91	5,76
48	3	2000	1216	7	3719	39,9	1930,46	6,5
96	3	2000	1295	9	3102	24,1	2327,36	3,94
384	3	2000	3945	98	7140	12,1	4663,38	1,98
960	3	2000	8048,5	253,5	149731,5	5,05	4822,155	0,825
7000	3	2000	9685	3901	47234	2	17261,9	0,37
15000	3	2000	18876,25	12276,75	51599,5	0,166	21448,35	0,2275
74000	3	2000	455611	411836,5	517172,5	0,1155	734,69	0

Envoi directement sur le réseau								
Taille du fichier envoyé (Ko)	Nombre de nœuds d'hébergement	Nombre de requêtes totales envoyées	Temps de réponse moyen (ms)	Temps de réponse minimal (ms)	Temps de réponse maximal (ms)	Débit (requêtes/sec)	Ko/sec reçus	Ko/sec envoyés
2	3	2000	4013	57	5447	36,8	17,22	88,07
48	3	2000	3717	52	6220	32,1	15,01	1552,79
96	3	2000	4609	39	7434	30,9	14,44	2975,72
384	3	2000	4972	105	8497	30,4	14,21	11678,6
960	3	2000	9067	137	19085	17,2	8,05	16519,4
7000	3	2000	41372	5744	82057	3,5	1,63	26528,45
15000	3	2000	112472	16307	158840	2	0,95	30988,63
74000	3	2000	449132	57874	1061375	0,83	0,09	15197,01

Pour l'envoi (*upload*) et la réception (*download*) de données avec un nombre de trois nœuds, la réduction du débit et l'augmentation des temps de réponse moyens des requêtes se traduisent en une augmentation des latences des nœuds IPFS à mesure que la taille des fichiers échangés augmente.

Plus la taille des fichiers augmente, plus les résultats dans les tableaux présentent des valeurs mesurées de moins en moins précises. Cela est dû à la présence d'un taux d'erreurs non négligeable qui est apparu lors de mesures, ayant provoqué un stockage en mémoire RAM (*Out of Memory*) trop important lors de la récupération des fichiers, ce qui a diminué les performances de l'outil de mesures JMeter. En deçà de fichiers de 7 Mo, les mesures ont été plus précises, car le taux d'erreurs est marginal. Des erreurs HTTP avec un *Response Code* de type 500 (Internal Server Error) sont également apparues lors des requêtes de toutes tailles en raison d'une erreur inconnue lors de la connexion répétée de l'API Java à la passerelle IPFS. L'augmentation du nombre de requêtes dans les échantillons on permis de diminuer le taux d'erreurs, car seuls les résultats en succès (*Status Code 200*) ont été pris en compte.

Etant donné que ces erreurs causent une diminution de la qualité des résultats généraux, en réduisant des temps de réponse moyens, ainsi que la quantité mesurée de données échangées, il est à considérer que la moyenne globale des résultats pour 2000 requêtes donne une tendance représentative de la réalité ; l'exécution des plans de tests ont été répétés à plusieurs reprises afin de vérifier cette tendance.
Deux graphiques permettent de compléter les résultats. La figure 3.31 représente le rapport du débit par rapport à la tailles des données, pour l'envoi et pour la réception de fichiers ; la figure 3.32 représente le rapport entre le temps de réponse moyen et l'augmentation des fichiers échangés pour l'envoi et la réception de données.

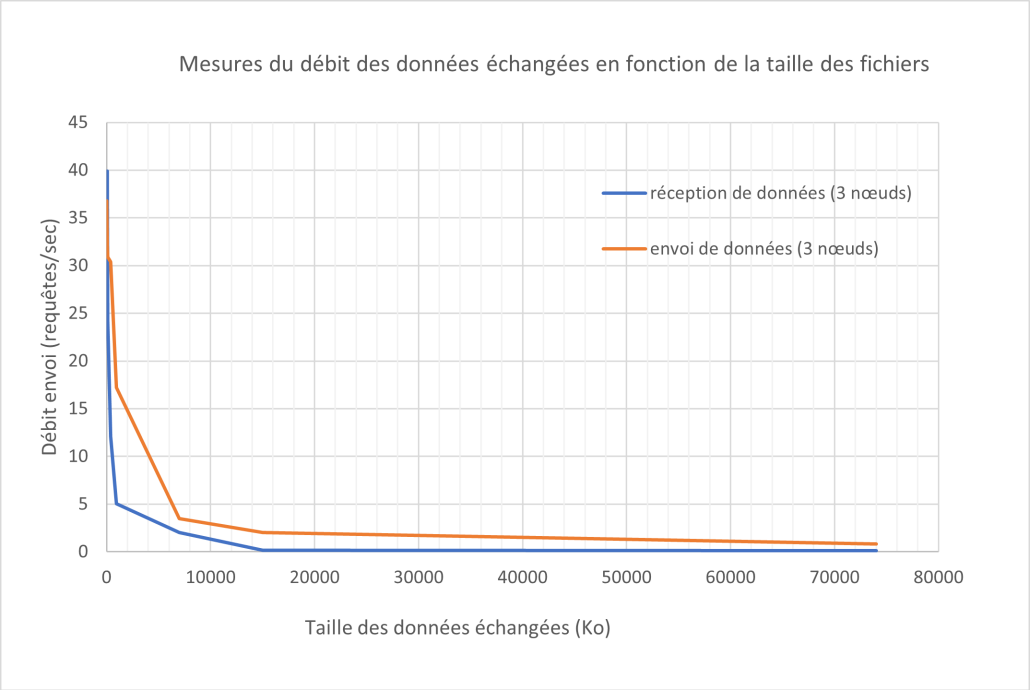


FIGURE 3.31 – Graphique présentant les mesures de performances du débit en fonction de la tailles des fichiers. Les mesures sont présentées pour l'envoi et la réception de fichiers sur trois nœuds IPFS.

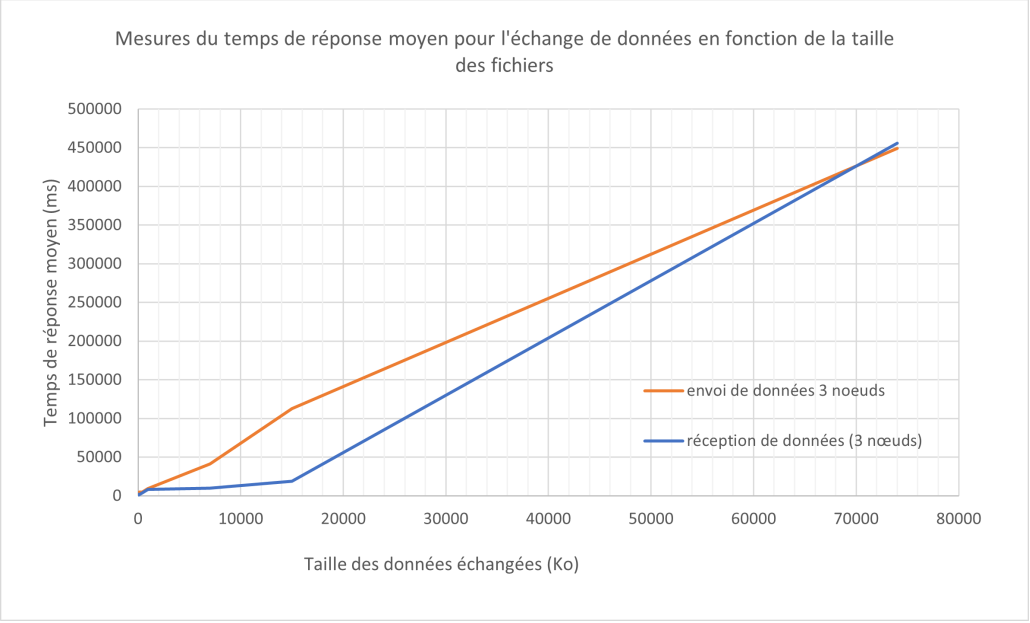


FIGURE 3.32 – Graphique présentant les mesures du temps de réponse moyen (ms) en fonction de la taille des fichiers. Les mesures sont présentées pour l'envoi et la réception de fichiers sur trois nœuds IPFS.

Chapitre 4

Discussion

4.1 Interprétation des résultats de la recherche

Dans la littérature scientifique, l'InterPlanetary File System est très fortement associé aux sujets relatifs au stockage distribué, au web distribué, aux protocoles d'échanges pair-à-pair et aux systèmes reposant sur les *blockchains*. Ces domaines évoluent régulièrement grâce aux résultats de nombreuses avancées dans la recherche scientifique. Il y transparaît clairement que les technologies des *blockchains* et de l'IPFS sont complémentaires pour obtenir des architectures d'applications *serverless* qui visent à garantir un stockage distribué des données dans les applications sur le web. Elles comblent les faiblesses du web centralisé, tout en assurant leur traçabilité, leur intégrité et leur sécurité à l'aide de méthodes de chiffrement.
Il apparaît que la plupart des solutions existantes utilisant IPFS sont combinées aux technologies des *blockchains* pour des cas d'utilisation relatifs à une distribution complète des applications. IPFS est également utilisé dans des systèmes de diffusion de données tel que le *streaming*, ou pour le transfert de fichiers chiffrés sans intermédiaire en transmettant un CID à un destinataire.

L'IPFS peut également participer à réduire les risques de censure en distribuant le contenu sur de nombreux nœuds pairs afin de fournir du contenu à l'aide d'un CID unique, mais dont les nœuds peuvent être répartis partout dans le monde. Prenons par exemple un cas de figure où de nombreux nœuds répartis partout dans le monde hébergent des données censurées. Les adresses IP et les domaines des différents passerelles qui servent le contenu doivent être bloquées individuellement, contrairement au web centralisé où il conviendrait de bloquer les adresses IP ou les domaines qui doivent être censurés. Par ailleurs, le navigateur Brave intègre des nœuds IPFS afin de permettre aux utilisateurs de devenir un nœud IPFS [1]. Cela augmente l'accessibilité à la technologie IPFS et réduit les risques de censure par la multiplication des nœuds qui mettent du contenu en cache.

En l'occurrence, l'utilisation de l'IPFS pour la réalisation d'une application web, voire pour l'échange de données directement entre des nœuds n'est pas applicable de la même manière qu'avec les protocoles d'échange standards utilisés sur le web centralisé. En effet, la réalisation d'applications client-serveur avec IPFS est possible pour obtenir un web partiellement décentralisé, mais il convient que les données publiées sur le réseau restent consultables à l'aide d'un lien immuable pour une durée indéterminée ; ces dernières doivent impérativement être sécurisées afin qu'un individu accédant à une adresse de contenu ne puisse pas accéder à des données privées. Cela requiert de s'adapter aux contraintes du protocole pour se prémunir d'effets de bords imprévus ou de la présence de faiblesses de sécurité lors du développement d'une application utilisant le protocole IPFS.

Lors de la réalisation du *Proof of Concept* du travail de recherche, le constat a été fait qu'il est possible de réaliser une décentralisation partielle d'une application client-serveur qui utilise l'IPFS comme système de stockage, mais cela cause alors la présence de points de défaillances uniques, et il convient de multiplier les nœuds de confiance pour éviter les risques de défaillances liées à des fautes et des erreurs.
Par ailleurs, il aurait été relativement aisé de réaliser une application complètement distribuée pour le partage de fichiers qui utilise un système de chiffrement asymétrique à l'aide d'une clé publique et d'une clé privée. La technologie de l'IPFS s'y prête bien pour l'échange spontané de données à l'aide de liens uniques sans devoir tracer des données, tout en assurant une version immuable de contenu pour un CID donné. La littérature scientifique et les ressources sur le web regorgent d'ailleurs de propositions de solutions qui s'y réfèrent. Ce-

1. Phoebe Poon, "Can IPFS — the Distributed Web fight against content censorship ?", `https://medium.com/likecoin/can-ipfs-the-distributed-web-fight-against-content-censorship-300e55cbf88c` (26/03/2021) (Date d'accès 18/07/2022)

pendant, l'objectif de ce travail restait tout de même d'identifier également les limites d'utilisation de l'IPFS. Le développement réalisé permet d'aboutir au constat que l'utilisation de l'IPFS en tant que système de stockage hors-chaîne pour une architecture d'application *serverless* est plus pertinente car il est compliqué de distribuer complètement les données d'une application sur le réseau de manière sécurisée et traçable sans utiliser la technologie des *blockchains*, mais aussi parce que le stockage *on-chain* est coûteux. L'utilisation des *blockchains* en complément des contrats intelligents au sein d'une application permet de s'affranchir d'un référentiel agissant comme autorité de confiance, tout en garantissant la liaison de la *logique métier* d'une application avec un système fonctionnel, fiable et dont les utilisateurs ne sont pas identifiés et enregistrés de manière centralisée.
En d'autres termes, dans un contexte de l'application d'une logique métier, l'utilisation des contrats intelligents des technologies *blockchains* est identifiée comme une solution essentielle pour encoder la logique d'une transaction entre deux pairs - sous la forme d'un contrat qui ne peut être rompu - lors des communications entre la partie client et le système de stockage des données décentralisé d'une application distribuée.

De plus, la réalisation du *Proof of Concept* mène à penser qu'il n'est pas possible de publier une application serveur (*back-end*) pouvant être démarrée de manière complètement distribuée sur IPFS en la répartissant sur des nœuds pairs qui ne sont pas des tiers de confiance dont on a le contrôle pour l'hébergement. Cela signifie qu'un serveur *back-end* destiné à gérer la logique métier d'une application web qui manipule des données privées (pour des utilisateurs non-anonymes qui requièrent une garantie de sécurité) ne peut pas convenir pour une distribution complète sur le réseau IPFS. Le besoin de sécurité des données étant un point clé pour un système de transactions de paiement tel que dans l'application client-serveur développée dans le cadre de la recherche (*PoC*), il ne semble pas évident de distribuer complètement une application avec l'architecture client-serveur sur le réseau IPFS.
Pour une architecture client-serveur telle que celle *PoC*, il est donc intuitif qu'il convient d'utiliser des serveurs dédiés pour l'hébergement de la partie serveur, car des configurations au niveau du serveur *back-end* nécessitent l'utilisation d'informations privées, telles que des chemins de fichiers, des liens hypertextes, des identifiants et des clés privées.

D'autre part, comme cela a pu être expliqué à travers l'état de l'art et le travail de recherche, l'utilisation du protocole IPFS est utile pour obtenir des architectures complètement décentralisées (distribuées). Dans ce contexte distribué avec l'IPFS, les applications de type serveur comme sur le web partiellement décentralisé/-centralisé n'ont plus lieu d'exister car les nœuds du réseau doivent servir d'intermédiaires pour le stockage des données et non pas pour fournir une infrastructure serveur pour le traitement non transparent des données, ni pour garantir la sécurité du système comme c'est le cas sur le web partiellement décentralisé. Etant donné que les données sur le réseau doivent être chiffrées avant d'être stockées, elles ne sont pas interprétables par les nœuds fournisseurs de stockage.

L'objectif des réseaux distribués proposant des fonctionnalités de systèmes de fichiers est de partager l'information sans avoir de point de défaillance unique (un serveur applicatif hôte de la logique métier). Pour que la partie serveur puisse être distribuée, il faudrait donc que le traitement effectué côté serveur (*back-end*) soit exécuté depuis chaque nœud sans compromettre la sécurité de l'accès aux données privées. Cela requiert une séparation complète entre les données privées des utilisateurs et les traitements logiques. Ainsi, une application de type serveur ayant pour objet d'avoir une logique métier cachée qui traite et analyse des données n'est pas possible sans avoir des nœuds reliés à un système de *blockchain* privée, qui chiffrent les données avant un envoi et qui les déchiffrent avant de les lire. Cette particularité provoque alors des risques d'avoir des points de défaillances uniques.

Dans le cas où il serait possible de distribuer une telle architecture client-serveur avec IPFS, il faudrait alors ajouter des couches de sécurité supplémentaires afin d'encapsuler les données de l'application serveur avec des systèmes de communication utilisant un système central qui permet de stocker ces données privées. Fleek, Infura, Textile sont décrits comme étant des solutions d'hébergement décentralisées pour le Web 3.0 qui doivent permettre d'héberger des applications serveur, mais leurs fondamentaux restent des systèmes opaques, qui hébergent parfois des données sur des serveurs centraux. Le contenu des exécutables à installer sur un *operating system* pour interagir avec eux à l'aide d'un CLI ne sont pas toujours vérifiables et signés. Il n'est pas possible de comprendre les rouages exacts de ces plateformes. En cas de malveillance des hébergeurs, il peut être facile pour ces systèmes d'intercepter les données échangées entre les clients et les serveurs afin de les utiliser sans que les propriétaires des applications ne s'en aperçoivent. Faire confiance à de tels systèmes d'hébergement peut donc être risqué.

Dans le contexte d'une application nécessitant le stockage et l'utilisation de données privées d'utilisateurs avec l'IPFS, la technologie proposée par Ceramic Network paraît être une technologie complémentaire à IPFS,

tout à fait appropriée à la réalisation d'une application complètement décentralisée. Elle offre effectivement un système d'identité décentralisé pour permettre une gestion des données d'identité à l'aide de la *blockchain* Ethereum.

Par ailleurs, les solutions de stockage de fichiers distribuées telle que Filecoin doivent aider à garantir l'accessibilité à du contenu avec un minimum de trois nœuds sur le réseau IPFS. Cela est particulièrement intéressant pour le stockage de contenu sur le web.
Néanmoins, dans le cas d'application du *Proof of Concept*, Filecoin ne résout pas la contrainte de devoir assurer la présence de serveurs de confiance pour une décentralisation partielle d'instances côté serveur (*back-end*).
Une solution qui permettrait de résoudre ce problème serait néanmoins l'utilisation de ThreeFold, qui permet de louer des serveurs de manière sécurisée pour former un réseau distribué. Les principes d'utilisation d'IPFS et de ThreeFold sont complémentaires et pourraient être couplés afin de répondre aux contraintes de nécessité de tiers de confiance pour des données privées sur un réseau distribué.

Il convient également de noter que le contexte du Web 3.0., le système Storj Network - qui tend à décentraliser les données qu'il héberge tout en assurant leur sécurité à l'aide d'un chiffrement et l'utilisation des *blockchains* - propose un système d'épinglage du contenu à l'aide d'IPFS pour augmenter la permanence des données et assurer l'unicité du contenu à l'aide de la création de CID [2]. Cela permet d'utiliser les fonctionnalités offertes par l'IPFS pour publier du contenu privé de manière sécurisée sur un réseau distribué. Etant donné que l'IPFS peut être rattaché aux technologies *blockchains* pour créer des applications *serverless*, l'utilisation de Storj avec les fondamentaux de l'IPFS peut permettre de sécuriser les données d'une application web sans devoir implémenter des systèmes de chiffrement des données par soi-même.

4.2 Réponses aux questions sous-jacentes de la question de recherche

Les sous-questions liées à la question de recherche ont pu être éclaircies à partir de l'état de l'art, ainsi que de la mise en œuvre d'une solution applicative utilisant le protocole IPFS et son système de fichiers.

4.2.1 Pourquoi faire du stockage décentralisé ?

Cette question a été traitée dans l'état de l'art, dans la section 2.2.4. En résumé, la décentralisation vise à garantir l'indépendance d'un système à la gouvernance d'une autorité centrale ; à permettre aux utilisateurs de garder le contrôle de leurs données ; à préserver la confidentialité des utilisateurs qui créent leur propre identité ; à connaître les utilisations des données fournies par les utilisateurs ; à tracer les données utilisées pour garantir leur authenticité ; à assurer l'intégrité des données ; à obtenir une disponibilité continue des systèmes répartis sur un réseau.
D'après ces éléments, ainsi que la recherche, il convient de retenir comme information que le stockage décentralisé permet de répartir les données enregistrées au sein d'un système sur un réseau afin d'en retirer les avantages de la décentralisation.

4.2.2 Quelles solutions de stockage décentralisé existent ?

Des solutions de stockage décentralisé existantes autres que le système de fichiers IPFS ont été présentées dans la section 2.2.6 ; des solutions de stockage décentralisé reliées aux systèmes *blockchains* ont été présentées dans la section 2.2.4.

4.2.3 Quels défis ces solutions soulèvent-elles ?

Dans l'état de l'art, les systèmes pair-à-pair ont été associés à plusieurs défis (voir 2.2) ; les systèmes basés sur les *blockchains* (voir 2.2.9) ont permis d'identifier d'autres défis ; concernant l'IPFS, plusieurs difficultés de mise en place pour l'adressage de contenu sont apparues dans le contexte des systèmes de communication pair-à-pair (voir 2.3.2).
De plus, lors du développement d'une application à l'aide de l'API "Java IPFS Http Client", certaines contraintes et défis sont apparus lors de la mise en place de l'application client-serveur (voir 3.6.2), ainsi que pour la distribution des données des utilisateurs (voir 3.6.4).

2. Storj Labs Inc., "Decentralized NFT Storage", `https://www.storj.io/ipfs` (2022) (Date d'accès 09/07/2022)

4.2.4 Quels sont les principes de fonctionnement d'IPFS ?

Les principes de fonctionnement d'IPFS ont été décrits dans l'état de l'art, dans la section 2.4. Sa mise en œuvre et son utilisation dans un contexte applicatif ont été présentées lors du développement d'une application dans la section 3.3.1.

4.2.5 Quelles sont les performances d'IPFS ?

Cette question a été traitée, d'une part, par la présentation des données relatives à l'état de l'art dans la section 2.6, et d'autre part, par les résultats des mesures de performances (voir 3.7.2) appliquée lors du processus de recherche (voir 3.6.6). En l'occurrence, le consensus entre les résultats de la recherche et la littérature scientifique reste qu'à mesure que la taille des fichiers échangés augmente, les performances de débit baissent, tandis que le temps de réponse des requêtes de d'envoi et de récupération du contenu augmente. Dans le cadre d'un réseau à petite échelle, ses performances de débit d'échange sont moins bonnes qu'avec d'autres protocoles d'échange. Néanmoins, l'utilisation d'IPFS améliore l'efficacité des systèmes basés sur les *blockchains* à l'aide du stockage hors-chaîne (*off-chain*) des données.

4.3 Limites dans la recherche

Concernant l'utilisation d'IPFS, des langages de programmation, tel que le langage Java, ne sont actuellement pas idéaux pour fonctionner avec IPFS malgré des tentatives ponctuelles de support par quelques contributeurs de la communauté IPFS. En effet, dans ce travail, la technologie Java a été utilisée pour la réalisation de l'application *Proof of Concept* car c'est un langage largement utilisé dans le monde professionnel. Il est tout à fait adéquat pour créer des application web robustes garantissant la sécurité des données pour une architecture client-serveur, mais les bibliothèques utilisées pour l'utilisation d'IPFS, telle que les APIs proposées par *Java IPFS Http Client* et *Java IPFS Deploy* n'ont plus été maintenues depuis plusieurs années. Elles ont des bogues non résolus ; elles ne sont pas suffisamment documentées ; elles ne sont donc pas mises à jour lorsque le système IPFS est modifié ; elles ne sont pas pleinement fonctionnelles. Il a fallu contourner les problèmes rencontrés et les résoudre en lisant le code source afin de réaliser une communication efficace avec le réseau IPFS.

Un constat reste que la réalisation d'une application client-serveur partiellement décentralisée en Java n'est pas la solution la plus adaptée pour l'utilisation de l'IPFS. En effet, il a été difficile de mener à bien l'entièreté de ce travail, car je ne suis pas parvenu à déployer la partie serveur de l'application sur une passerelle de déploiement décentralisée pour vérifier la faisabilité de l'utilisation d'une application Java côté serveur dans un contexte distribué. Il aurait été plus pertinent d'utiliser le langage Go ou JavaScript pour la réalisation du PoC.

Enfin, les tests de performances pour la vérification de la latence sur le réseau IPFS, pour un nombre élevé de requêtes parallèles et des fichiers de tailles variables, n'ont pas été réalisés avec un nombre de nœuds variable, ni dans des conditions de réplication à grande échelle. Les mesures ont été réalisées pour un nombre réduit de nœuds locaux, mais cela n'a pas mené à une distribution des fichiers hébergés sur le réseau global.
Des tentatives d'utilisation d'un nombre de nœuds plus élevé (sept nœuds) ont été réalisées, mais elles ont été infructueuses, car des erreurs dues à la quantité de mémoire disponible sur la machine de test et à la quantité de charge de CPU demandée pour le traitement des échanges de données ont engendré des latences trop élevées pour pouvoir être prises en considération dans les résultats des mesures.
Parmi les résultats pris en compte, une proportion faible de requêtes ont également engendré des erreurs HTTP avec un *response code* 500 à de nombreuses reprises, ne permettant pas de terminer d'exécuter des tests de charge avec 100% de succès. Il est donc difficile de savoir quels résultats des mesures sont réellement représentatifs, ou si c'est uniquement leur tendance globale qui est à prendre en considération. D'après la littérature scientifique actuelle, ainsi que par ce manque de reproduction d'un contexte de reproduction d'un réseau IPFS complet pour le partage de fichiers pour des tests de performances, il est difficile de conclure que l'IPFS est plus ou moins performant qu'un autre protocole et qu'il convient mieux ou non de l'utiliser pour des raisons de performances étant donné que son contexte d'utilisation à grande échelle peut faire varier l'interprétation des résultats.

Chapitre 5

Conclusion

Dans ce mémoire, une étude a été effectuée en profondeur afin d'identifier et de comprendre les méthodes de fonctionnement de l'IPFS, ainsi que son applicabilité pour faire du stockage décentralisé au travers de contextes applicatifs qui interagissent avec le Web.

Pour rappel, comme expliqué dans la problématisation 3.2, la question de recherche est « Pourquoi et comment mettre en œuvre une solution de stockage décentralisé ; le cas d'IPFS ? ». Cette question peut être divisée en deux questions :
La première partie de la question de recherche, « Comment mettre en œuvre une solution de stockage décentralisé avec l'IPFS ? » a été répondue par la description de l'implémentation d'une solution utilisant l'IPFS et en justifiant les choix qui sont faits au cours d'un développement pour orienter l'application vers un résultat final plutôt qu'un autre.
La seconde partie de la question de recherche, « Pourquoi mettre en œuvre une solution de stockage décentralisé avec l'IPFS », a été répondue au fur et à mesure du développement par la justification d'orientation des choix d'implémentation pris au cours du développement, ainsi qu'à l'aide de l'état de l'art et de la discussion des résultats de la recherche (voir section 4.1).
En complément des informations retirées dans la littérature scientifique pour la réalisation de l'état de l'art, le travail de recherche effectué a permis de mettre en avant la pertinence et l'applicabilité de l'utilisation de l'IPFS pour le stockage décentralisé dans un contexte de développement d'application.

Comme discuté lors de l'interprétation des résultats (4.1), l'utilisation de l'IPFS est considérée comme étant davantage pertinente pour les systèmes distribués qui utilisent les technologies du Web 3.0 et qui favorisent les architectures *serverless* afin d'externaliser complètement des infrastructures sur les réseaux pair-à-pair et de ne pas dépendre d'un référentiel centralisé qui agit comme une autorité centrale. Le contenu stocké sur le réseau pair-à-pair IPFS permet à chaque pair de participer pour héberger et diffuser des données, tout en retirant les avantages du stockage décentralisé (4.2.1).

L'IPFS peut également participer à réduire les risques de censure, ainsi qu'à intégrer des systèmes interopérables pour le Web 3.0 où chacun peut être anonyme et créer sa propre identité afin de parcourir le Web. Les applications web qui interviennent sur le réseau IPFS peuvent alors proposer des systèmes totalement décentralisés basés sur les *blockchains*, proposer du contenu versionné à l'aide des CID et des Merkle DAG, ou encore utiliser le protocole IPFS afin de créer des applications partiellement décentralisées telles que des applications client-serveur hébergées par des nœuds de confiance. Ces dernières utilisent alors l'IPFS pour diffuser du contenu identifié par des CID sur le réseau et peuvent traiter la logique métier sans transparence, tout en ayant la possibilité de garder une trace des données côté serveur si elles n'utilisent pas les *blockchains* et qu'elles ne décentralisent pas le stockage complètement.
Par ailleurs, lors du travail de recherche, les applications possédant une architecture client-serveur ont été identifiées comme étant peu pertinentes pour l'utilisation de l'IPFS malgré le fait que cela est réalisable, étant donné que l'infrastructure serveur de ces applications ne peut pas être complètement décentralisée. Même si ces applications ne stockent pas toujours des données, car elles peuvent être effectives pour des contextes d'échange de fichiers chiffrés à l'aide de clés asymétriques sans la nécessité de stocker des données sur le serveur hôte, ces cas d'application ont semblé marginaux par rapport aux ensembles d'application web nécessitant d'utiliser des bases de données, de la logique métier et des infrastructures permettant l'utilisation de serveurs applicatifs. Par ailleurs, il reste tout à fait possible de créer des applications *serverless* qui ont les mêmes fonctionnalités que ces applications client-serveur sans stockage de données côté serveur. D'autres solutions complémentaires à l'IPFS ont été identifiées comme étant plus adaptées pour une utilisation d'applications possédant des traitements orientés serveur, dans lesquelles il est notamment possible d'héberger des applica-

tion pour créer un web décentralisé.

De plus, malgré la présence de performances pouvant être dépassées par d'autres protocoles, il s'avère qu'IPFS a des avantages quant à l'unicité des liens pour éviter la duplication des liens d'accès aux données et pour augmenter la disponibilité d'un contenu versionné. Utiliser IPFS permet notamment de partager du contenu entre des pairs afin de ne pas subir des indisponibilités en répartissant un contenu identifié de manière unique et uniforme sur des pairs du réseau. Même si les performances de l'IPFS sont satisfaisantes dans certaines conditions, il est difficile de conclure qu'il convient mieux ou non de l'utiliser plutôt qu'un autre protocole pour des raisons de performances.

En conclusion, les informations retirées lors de ce travail de recherche ont permis de mettre en avant le fonctionnement de l'InterPlanetary File System, son contexte d'utilisation, ses moyens de mise en œuvre et d'aboutir à une proposition d'arbre de décision permettant à un informaticien de comprendre lorsque l'InterPlanetary File Système peut être utilisé comme système de stockage décentralisé.
De plus, la mise en perspective de l'utilisation de ce protocole pour la réalisation d'une application client-serveur classique, requérant néanmoins de manipuler des données d'utilisateurs hautement privées a permis de pousser l'identification des contraintes et des défis qui relèvent de l'utilisation d'une telle technologie pour la manipulation de telles données, ainsi que d'identifier des solutions adaptées pour y répondre. En l'occurrence, il a pu être mis en perspective le fait que l'IPFS est un système de stockage de fichiers proposant un protocole dédié au Web décentralisé partiel ou complet (distribué), ainsi qu'à la publication et à la diffusion de données entre pairs. Les données partagées avec le protocole IPFS peuvent être de tout type et nécessiter ou non d'être sécurisées à l'aide de systèmes de chiffrement ; être rendues hautement disponibles à l'aide de systèmes de réplication ou de stockage tel que Filecoin pour assurer qu'elles restent présentes en tout temps sur les réseaux ; nécessiter que l'utilisateur s'authentifie tout en favorisant la confidentialité de ses données à l'aide des identités décentralisées. Les données peuvent rester intègres grâce aux identificateurs de contenu (CID) qui les identifient de manière unique afin de les rendre authentiques et versionnées à l'aide des arbres de Merkle, tout en étant possiblement reliées aux systèmes *blockchains* qui assurent leur traçabilité.

Par ailleurs, les réponses aux sous-questions résultant de la question de recherche, résolues par la réponse à la problématique de l'état de l'art et à travers le développement d'une application dans le travail de recherche ont permis d'aboutir à la création d'un arbre de décisions représenté par la figure 5.1. Ce dernier doit permettre d'orienter les choix d'un informaticien pour comprendre dans quel cas l'InterPlanetary peut être utilisé pertinemment comme système de stockage décentralisé dans un contexte de développement logiciel. L'ensemble de ce travail additionné à cet arbre de décision répondent implicitement à la question « Pourquoi mettre en œuvre une solution de stockage décentralisé avec l'IPFS ? ».

5.1 Perspectives de recherches plus poussées

Les résultats de la recherche actuelle pourraient être améliorés en développant une solution semblable au *Proof of Concept* réalisé dans le cadre de la recherche, mais destinée au web distribué ; donc respectant les principes du Web 3.0 en utilisant notamment les systèmes *blockchains*, les contrats intelligents, et les identités décentralisées. Le développement d'une application *serverless* utilisant le Ceramic Network avec une combinaison des *blockchains* serait une approche complémentaire adéquate à l'utilisation de l'IPFS. IPFS serait utilisé pour faire du stockage hors-chaîne.
Une telle réalisation permettrait d'identifier si l'arbre de décision proposé peut être amélioré, tout en vérifiant la faisabilité de la création d'une application web distribuée semblable à l'application web réalisée ; cela permettrait d'interagir avec des données privées en assurant un niveau de sécurité élevé, mais en utilisant le stockage de données avec IPFS dans un système complètement distribué.
De plus, comparer les performances de ces deux applications, partiellement et complètement décentralisées, en les déployant dans des contextes cohérents sur l'IPFS et équivalents (nombre de nœuds, bande passante) serait un moyen pertinent de vérifier leur efficacité respective.
De plus, il pourrait être intéressant d'identifier la faisabilité du déploiement d'une application client-serveur utilisant IPFS, combinée à un système de stockage distribué complémentaire, afin de ne pas dépendre de nœuds de confiance.

D'autre part, lors de la réalisation de ce travail, il s'est avéré que, la technologie IPFS étant récente, des lacunes sont présentes au niveau de la sécurité, des performances, de la complexité de réalisation, des cas d'application de la technologie. Il conviendrait de recenser les lacunes de cette technologie de manière exhaustive et de chercher des pistes pour les résoudre.

Par ailleurs, un travail de recensement et de comparaison exhaustif des différents systèmes de stockage décen-

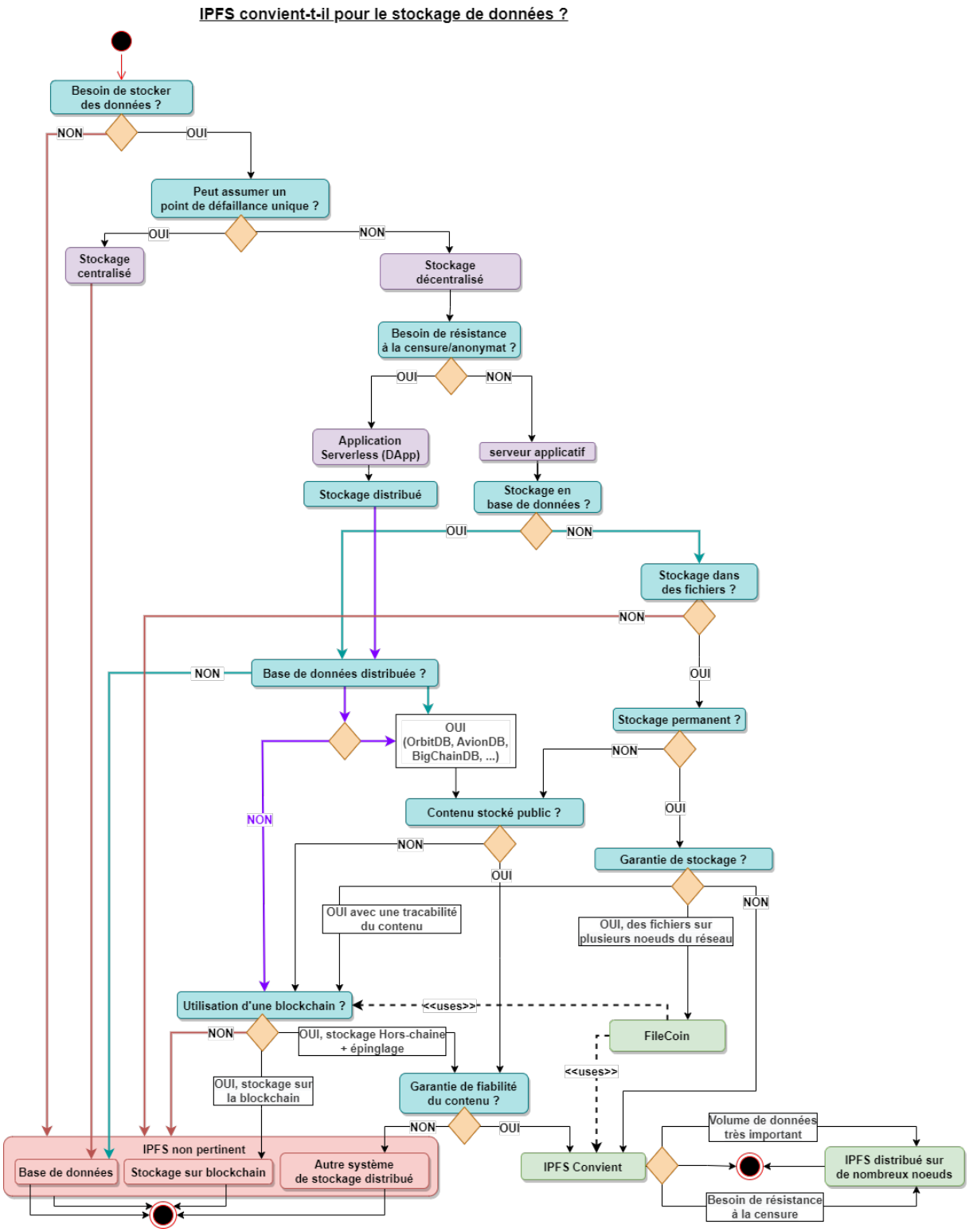


FIGURE 5.1 – Arbre de décisions permettant de déterminer lorsque IPFS peut être utilisé pertinemment pour le stockage de données

tralisé pourrait permettre de comprendre pourquoi un système de stockage est plus adapté qu'un autre à un cas d'application pour le développement d'un système.

Enfin, étudier les performances d'IPFS pour l'échange de fichiers de tailles variables dans un très large spectre, ainsi qu'avec un nombre très élevé de nœuds distribués et répartis dans le monde serait idéal. En effet, les tests de performances ont été réalisés dans des conditions avec peu de nœuds *clusterisés*, offrant un faible taux de réplication. Il est difficile de réaliser des tests dans des conditions offrant un taux élevé de réplication depuis un ordinateur personnel. Une réalisation dans des conditions réelles permettrait de mettre en œuvre des tests de

performances dans des conditions réalistes pour la distribution de contenu de tailles diverses, sur un nombre élevé d'un réseau de nœuds IPFS distribué complet. Pour ce faire, il faudrait augmenter le nombre de nœuds IPFS de manière conséquente avec cinquante à cent nœuds et partager des fichiers de plus de 500Mo pour augmenter la taille des flux de données depuis plusieurs serveurs localisés à des endroits physiques différents qui répliquent des confitions réelles d'un web distribué. Il conviendrait néanmoins d'exécuter ces tests dans des conditions environnementales identiques afin de comparer les résultats sans dépendre de variations de performances matérielles ou logicielles.

Bibliographie

[1] Omar Abdullah Lajam and Tarek Ahmed Helmy. Performance evaluation of ipfs in private networks. In *2021 4th International Conference on Data Storage and Data Engineering*, pages 77–84, 2021.

[2] Sawood Alam, Mat Kelly, and Michael L Nelson. Interplanetary wayback : The permanent web archive. In *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries*, pages 273–274, 2016.

[3] Muhammad Salek Ali, Koustabh Dolui, and Fabio Antonelli. Iot data privacy via blockchains and ipfs. In *Proceedings of the seventh international conference on the internet of things*, pages 1–7, 2017.

[4] Muneeb Ali, Ryan Shea, Jude Nelson, and Michael J Freedman. Blockstack : A new decentralized internet. *Whitepaper, May*, 2017.

[5] Saqib Ali, Guojun Wang, Bebo White, and Roger Leslie Cottrell. A blockchain-based decentralized data storage and access framework for pinger. In *2018 17th IEEE international conference on trust, security and privacy in computing and communications/12th IEEE international conference on big data science and engineering (TrustCom/BigDataSE)*, pages 1303–1308. IEEE, 2018.

[6] Morteza Alizadeh, Karl Andersson, and Olov Schelén. Efficient decentralized data storage based on public blockchain and ipfs. In *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, pages 1–8. IEEE, 2020.

[7] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM computing surveys (CSUR)*, 36(4) :335–371, 2004.

[8] Alex Auvolat and François Taïani. Merkle search trees : efficient state-based crdts in open networks. In *2019 38th Symposium on Reliable Distributed Systems (SRDS)*, pages 221–22109. IEEE, 2019.

[9] Oscar Avellaneda, Alan Bachmann, Abbie Barbir, Joni Brenan, Pamela Dingle, Kim Hamilton Duffy, Eve Maler, Drummond Reed, and Manu Sporny. Decentralized Identity : Where Did It Come from and Where Is It Going ? *IEEE Communications Standards Magazine*, 3(4) :10–13, 2019.

[10] Achmad Muhaimin Aziz, Adityas Widjajarto, and Avon Budiyono. Analysis and implementation of nodes communication between interplanetary file system (ipfs) in smart contract ethereum. *ICORE*, 5(1), 2020.

[11] Naimul Basher, Aniket Mahanti, Anirban Mahanti, Carey Williamson, and Martin Arlitt. A comparative analysis of web and Peer-to-Peer traffic. *Proceeding of the 17th International Conference on World Wide Web 2008, WWW'08*, pages 287–296, 2008.

[12] Ammar Ayman Battah, Mohammad Moussa Madine, Hamad Alzaabi, Ibrar Yaqoob, Khaled Salah, and Raja Jayaraman. Blockchain-based multi-party authorization for accessing ipfs encrypted data. *IEEE Access*, 8 :196813–196825, 2020.

[13] Ingmar Baumgart and Sebastian Mies. S/kademlia : A practicable approach towards secure key-based routing. In *2007 International Conference on Parallel and Distributed Systems*, pages 1–8. IEEE, 2007.

[14] Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv :1407.3561*, 2014.

[15] Juan Benet and Nicola Greco. Filecoin : A decentralized storage network. *Protoc. Labs*, pages 1–36, 2018.

[16] Nazanin Zahed Benisi, Mehdi Aminian, and Bahman Javadi. Blockchain-based decentralized storage networks : A survey. *Journal of Network and Computer Applications*, 162 :102656, 2020.

[17] BigchainDB GmbH. Whitepaper : BigchainDB 2.0 The Blockchain Database. *Whitepaper*, (May) :1–14, 2018.

[18] Jakob Blomer. A survey on distributed file system technology. In *Journal of Physics : Conference Series*, volume 608, page 012039. IOP Publishing, 2015.

[19] Lorenz Breidenbach, Christian Cachin, Benedict Chan, Alex Coventry, Steve Ellis, Ari Juels, Farinaz Koushanfar, Andrew Miller, Brendan Magauran, Daniel Moroz, Sergey Nazarov, Alexandru Topliceanu, Florian Tramèr, and Fan Zhang. Chainlink 2.0 : Next Steps in the Evolution of Decentralized Oracle Networks. pages 1–136, 2021.

[20] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4) :398–461, 2002.

[21] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. Bigtable : A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2) :1–26, 2008.

[22] Yongle Chen, Hui Li, Kejiao Li, and Jiyang Zhang. An improved p2p file system scheme based on ipfs and blockchain. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2652–2657. IEEE, 2017.

[23] Bram Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72. Berkeley, CA, USA, 2003.

[24] Introduction Compared, Interplanetary Linked Data, and Background We. A protocol & event-sourced database for decentralized user-siloed data. *Textile.io*, page 21, 2019.

[25] Bastien Confais, Adrien Lebre, and Benoît Parrein. Performance analysis of object store systems in a fog/edge computing infrastructures. In *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 294–301. IEEE, 2016.

[26] Mauro Conti, E Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys & Tutorials*, 20(4) :3416–3452, 2018.

[27] Alexandru-Gabriel Cristea, Lenuta Alboaie, Andrei Panu, and Vlad Radulescu. Offline but still connected with ipfs based communication. *Procedia Computer Science*, 176 :1606–1612, 2020.

[28] Ernesto Damiani, Sabrina De Capitani Di Vimercati, Stefano Paraboschi, Pierangela Samarati, and Fabio Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 207–216, 2002.

[29] Erik Daniel and Florian Tschorsch. Ipfs and friends : A qualitative comparison of next generation peer-to-peer data networks. *IEEE Communications Surveys & Tutorials*, 2022.

[30] Alfonso De la Rocha, David Dias, and Yiannis Psaras. Accelerating content routing with bitswap : A multi-path file transfer protocol in ipfs and filecoin. 2021.

[31] Richard Dennis and Gareth Owen. Rep on the block : A next generation reputation system based on the blockchain. *2015 10th International Conference for Internet Technology and Secured Transactions, ICITST 2015*, pages 131–138, 2016.

[32] David Dias and Juan Benet. Distributed web applications with ipfs, tutorial. In *International Conference on Web Engineering*, pages 616–619. Springer, 2016.

[33] Mallikarjun Reddy Dorsala, VN Sastry, and Sudhakar Chapram. Blockchain-based solutions for cloud computing : A survey. *Journal of Network and Computer Applications*, 196 :103246, 2021.

[34] Michael J Freedman, Eric Freudenthal, and David Mazieres. Democratizing content publication with coral. In *NSDI*, volume 4, pages 18–18, 2004.

[35] J Hao, Yan Sun, and Hong Luo. A safe and efficient storage scheme based on blockchain and ipfs for agricultural products tracking. *Journal of Computers*, 29(6) :158–167, 2018.

[36] Ragib Hasan, Zahid Anwar, William Yurcik, Larry Brumbaugh, and Roy Campbell. A survey of peer-to-peer storage techniques for distributed file systems. In *International Conference on Information Technology : Coding and Computing (ITCC'05)-Volume II*, volume 2, pages 205–213. IEEE, 2005.

[37] Ethan Heilman. One weird trick to stop selfish miners : Fresh bitcoins, a solution for the honest miner. In *International Conference on Financial Cryptography and Data Security*, pages 161–162. Springer, 2014.

[38] Mark Robert Henderson, Samuli Pöyhtäri, Vesa-Ville Piiroinen, Juuso Räsänen, Shams Methnani, and Richard Littauer. The OrbitDB Field Manual.

[39] Jordi Herrera-Joancomartí and Cristina Pérez-Solà. Privacy in bitcoin transactions : new challenges from blockchain scalability solutions. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 26–44. Springer, 2016.

[40] Van-Hoan Hoang, Elyes Lehtihet, and Yacine Ghamri-Doudane. Privacy-preserving blockchain-based data sharing platform for decentralized storage systems. In *2020 IFIP Networking conference (networking)*, pages 280–288. IEEE, 2020.

[41] Alex Hoffman, Eric Becerril-Blas, Kevin Moreno, and Yoohwan Kim. Decentralized security bounty management on blockchain and ipfs. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0241–0247. IEEE, 2020.

[42] Huawei Huang, Jianru Lin, Baichuan Zheng, Zibin Zheng, and Jing Bian. When blockchain meets distributed file systems : An overview, challenges, and open issues. *IEEE Access*, 8 :50574–50586, 2020.

[43] Tam T Huynh, Thuc D Nguyen, and Hanh Tan. A survey on security and privacy issues of blockchain technology. In *2019 international conference on system science and engineering (ICSSE)*, pages 362–367. IEEE, 2019.

[44] Christos Karapapas, Iakovos Pittaras, Nikos Fotiou, and George C Polyzos. Ransomware as a service using smart contracts and ipfs. *arXiv preprint arXiv :2003.04426*, 2020.

[45] Manpreet Kaur, Mohammad Zubair Khan, Shikha Gupta, and Abdullah Alsaeedi. Adoption of blockchain with 5g networks for industrial iot : Recent advances, challenges, and potential solutions. *IEEE Access*, 2021.

[46] Shreya Khatal, Jayant Rane, Dhiren Patel, Pearl Patel, and Yann Busnel. Fileshare : A blockchain and ipfs framework for secure file sharing and data provenance. In *Advances in Machine Learning and Computational Intelligence*, pages 825–833. Springer, 2021.

[47] Soohyeong Kim, Yongseok Kwon, and Sunghyun Cho. A Survey of Scalability Solutions on Blockchain. *9th International Conference on Information and Communication Technology Convergence : ICT Convergence Powered by Smart Intelligence, ICTC 2018*, pages 1204–1207, 2018.

[48] Soohyeong Kim, Yongseok Kwon, and Sunghyun Cho. A survey of scalability solutions on blockchain. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1204–1207. IEEE, 2018.

[49] Daniel Kraft. Difficulty control for blockchain-based consensus systems. *Peer-to-peer Networking and Applications*, 9(2) :397–413, 2016.

[50] Bruno Krebs. The openid connect handbook. *https ://auth0.com/*.

[51] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, et al. Oceanstore : An architecture for global-scale persistent storage. *ACM SIGOPS Operating Systems Review*, 34(5) :190–201, 2000.

[52] Randhir Kumar, Ningrinla Marchang, and Rakesh Tripathi. Distributed off-chain storage of patient diagnostic reports in healthcare system using ipfs and blockchain. In *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 1–5. IEEE, 2020.

[53] Randhir Kumar and Rakesh Tripathi. Implementation of distributed file storage and access framework using ipfs and blockchain. In *2019 Fifth International Conference on Image Information Processing (ICIIP)*, pages 246–251. IEEE, 2019.

[54] Avinash Lakshman and Prashant Malik. Cassandra : a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2) :35–40, 2010.

[55] Vu Le, Ramin Moazeni, and Melody Moh. Improving security and performance of distributed ipfs-based web applications with blockchain. In *International Conference on Advances in Cyber Security*, pages 114–127. Springer, 2021.

[56] Eliezer Levy and Abraham Silberschatz. Distributed file systems : Concepts and examples. *ACM Computing Surveys (CSUR)*, 22(4) :321–374, 1990.

[57] Peter Macko and Jason Hennessey. Survey of distributed file system design choices. *ACM Transactions on Storage (TOS)*, 18(1) :1–34, 2022.

[58] Warren Matthews and Les Cottrell. The pinger project : active internet performance monitoring for the henp community. *IEEE Communications Magazine*, 38(5) :130–136, 2000.

[59] Petar Maymounkov and David Mazieres. Kademlia : A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer, 2002.

[60] David Mazieres. Self-certifying file system, 2000.

[61] David Mazieres and M Frans Kaashoek. Escaping the evils of centralized control with self-certifying pathnames. In *Proceedings of the 8th ACM SIGOPS European workshop on Support for composing distributed applications*, pages 118–125, 1998.

[62] Shicong Meng, Cong Shi, Dingyi Han, Xing Zhu, and Yong Yu. A statistical study of today's gnutella. In *Asia-Pacific Web Conference*, pages 189–200. Springer, 2006.

[63] Ralph C. Merkle. Protocols for public key cryptosystems. In *1980 IEEE Symposium on Security and Privacy*, pages 122–122, 1980.

[64] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.

[65] Ralph C Merkle. Protocols for public key cryptosystems. In *Secure communications and asymmetric cryptosystems*, pages 73–104. Routledge, 2019.

[66] Shapna Muralidharan and Heedong Ko. An interplanetary file system (ipfs) based iot framework. In *2019 IEEE international conference on consumer electronics (ICCE)*, pages 1–2. IEEE, 2019.

[67] Satoshi Nakamoto. Bitcoin : A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[68] Nishara Nizamuddin, Haya R Hasan, and Khaled Salah. Ipfs-blockchain-based authenticity of online publications. In *International Conference on Blockchain*, pages 199–212. Springer, 2018.

[69] Nishara Nizamuddin, Khaled Salah, M Ajmal Azad, Junaid Arshad, and MH Rehman. Decentralized document version control using ethereum blockchain and ipfs. *Computers & Electrical Engineering*, 76 :183–197, 2019.

[70] Robert Norvill, Beltran Borja Fiz Pontiveros, Radu State, and Andrea Cullen. Ipfs for reduction of chain size in ethereum. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1121–1128. IEEE, 2018.

[71] Emmanuel Nyaletey, Reza M Parizi, Qi Zhang, and Kim-Kwang Raymond Choo. Blockipfs-blockchain-enabled interplanetary file system for forensic and trusted data traceability. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 18–25. IEEE, 2019.

[72] Maxwell Ogden, Karissa McKelvey, Mathias Buus Madsen, et al. Dat-distributed dataset synchronization and versioning. *Open Science Framework*, 10, 2017.

[73] White Paper. Swarm fund the blockchain for private equity. 2018.

[74] White Paper. The ThreeFold Grid and Token. 2019.

[75] Constantinos Patsakis and Fran Casino. Hydras and ipfs : a decentralised playground for malware. *International Journal of Information Security*, 18(6) :787–799, 2019.

[76] Morgen E Peck. Blockchain world-do you need a blockchain ? this chart will tell you if the technology can solve your problem. *IEEE Spectrum*, 54(10) :38–60, 2017.

[77] Doriane Perard, Lucas Gicquel, and Jérôme Lacan. Blockhouse : Blockchain-based distributed storehouse system. In *2019 9th Latin-American Symposium on Dependable Computing (LADC)*, pages 1–4. IEEE, 2019.

[78] Michael Piatek, Tomas Isdal, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. Do incentives build robustness in bittorrent. In *Proc. of NSDI*, volume 7, page 4, 2007.

[79] Eugenia Politou, Efthimios Alepis, Constantinos Patsakis, Fran Casino, and Mamoun Alazab. Delegated content erasure in IPFS. *Future Generation Computer Systems*, 112 :956–964, 2020.

[80] Chaitanya Rahalkar and Dhaval Gujar. Content addressed p2p file system for the web with blockchain-based meta-data integrity. In *2019 International Conference on Advances in Computing, Communication and Control (ICAC3)*, pages 1–4. IEEE, 2019.

[81] Matei Ripeanu, Adriana Iamnitchi, and Ian Foster. Mapping the gnutella network. *IEEE Internet Computing*, 6(1) :50–57, 2002.

[82] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. P2p mixing and unlinkable bitcoin transactions. *Cryptology ePrint Archive*, 2016.

[83] Hector Sanjuan, Samuli Poyhtari, Pedro Teixeira, and Ioannis Psaras. Merkle-crdts : Merkle-dags meet crdts. *arXiv preprint arXiv :2004.00107*, 2020.

[84] Rüdiger Schollmeier and Gero Schollmeier. Why peer-to-peer (p2p) does scale : An analysis of p2p traffic patterns. In *Proceedings. Second International Conference on Peer-to-Peer Computing,*, pages 112–119. IEEE, 2002.

[85] Meet Shah, Mohammedhasan Shaikh, Vishwajeet Mishra, and Grinal Tuscano. Decentralized cloud storage using blockchain. In *2020 4th International conference on trends in electronics and informatics (ICOEI)(48184)*, pages 384–389. IEEE, 2020.

[86] Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah. A review of smart contract-based platforms, applications, and challenges. *Cluster Computing*, pages 1–27, 2022.

[87] Jiajie Shen, Yi Li, Yangfan Zhou, and Xin Wang. Understanding i/o performance of ipfs storage : a client's perspective. In *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE, 2019.

[88] LinFei Shi, Hong Luo, XueMei Yang, and Yan Sun. A high-availability data backup strategy for ipfs. In *2019 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, pages 1–2. IEEE, 2019.

[89] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, pages 1–10. Ieee, 2010.

[90] Mathis Steichen, Beltran Fiz, Robert Norvill, Wazen Shbair, and Radu State. Blockchain-Based, Decentralized Access Control for IPFS. *Proceedings - IEEE 2018 International Congress on Cybermatics : 2018 IEEE Conferences on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, iThings/GreenCom/CPSCom/SmartData/Blockchain/CIT 2018*, pages 1499–1506, 2018.

[91] Mathis Steichen, Beltran Fiz, Robert Norvill, Wazen Shbair, and Radu State. Blockchain-based, decentralized access control for ipfs. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1499–1506. IEEE, 2018.

[92] Storj. Storj : Decentralized cloud storage network framework. *Storj Labs*, 2018.

[93] Jin Sun, Xiaomin Yao, Shangping Wang, and Ying Wu. Blockchain-based secure storage and access scheme for electronic medical records in ipfs. *IEEE Access*, 8 :59389–59401, 2020.

[94] Antonio Tenorio-Fornés, Samer Hassan, and Juan Pavón. Open peer-to-peer systems over blockchain and ipfs : An agent oriented framework. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pages 19–24, 2018.

[95] Tran Doan Thanh, Subaji Mohan, Eunmi Choi, SangBum Kim, and Pilsung Kim. A taxonomy and survey on distributed file systems. In *2008 Fourth International Conference on Networked Computing and Advanced Information Management*, volume 1, pages 144–149. IEEE, 2008.

[96] David Vorick and Luke Champine. Sia : Simple decentralized storage. *Retrieved May*, 8 :2018, 2014.

[97] Shaohua Wan, Meijun Li, Gaoyang Liu, and Chen Wang. Recent advances in consensus protocols for blockchain : a survey. *Wireless networks*, 26(8) :5579–5593, 2020.

[98] Liang Wang and Jussi Kangasharju. Measuring large-scale distributed systems : case of bittorrent mainline dht. In *IEEE P2P 2013 Proceedings*, pages 1–10. IEEE, 2013.

[99] Shangping Wang, Yinglong Zhang, and Yaling Zhang. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. *Ieee Access*, 6 :38437–38450, 2018.

[100] Oscar Wennergren, Mattias Vidhall, and Jimmy Sörensen. Transparency analysis of distributed file systems : With a focus on interplanetary file system, 2018.

[101] Shawn Wilkinson, Tome Boshevski, Josh Brandoff, and Vitalik Buterin. Storj a peer-to-peer cloud storage network. *Whitepaper*, 2014.

[102] Shawn Wilkinson and Jim Lowry. Metadisk a blockchain-based decentralized file storage application. *Whitepaper*, 2014.

[103] Gavin Wood. An Introduction to Polkadot state actors .". *Lightpaper*, page 17, 2019.

[104] Gavin Wood et al. Ethereum : A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014) :1–32, 2014.

[105] Jie Xu, Kaiping Xue, Shaohua Li, Hangyu Tian, Jianan Hong, Peilin Hong, and Nenghai Yu. Healthchain : A blockchain-based privacy preserving scheme for large-scale health data. *IEEE Internet of Things Journal*, 6(5) :8770–8781, 2019.

[106] Quanqing Xu, Zhiwen Song, Rick Siow Mong Goh, and Yongjun Li. Building an ethereum and ipfs-based decentralized social network system. In *2018 IEEE 24th international conference on parallel and distributed systems (ICPADS)*, pages 1–6. IEEE, 2018.

[107] Sourabh Yadav, Nonita Sharma, Monika Mangla, and Asmita Mahajan. Blockchain and ipfs based framework for secure student document record keeping. *Journal of Educational Multimedia and Hypermedia*, 30(2) :165–181, 2021.

[108] Qiuhong Zheng, Yi Li, Ping Chen, and Xinghua Dong. An innovative ipfs-based storage model for blockchain. In *2018 IEEE/WIC/ACM international conference on web intelligence (WI)*, pages 704–708. IEEE, 2018.

Annexe A

Glossaire

Adressage de localisation : Le Web centralisé utilise des adresses permettant d'accéder à des données en créant des liens sur le Web. Ce sont des liens hypertextes de type URL (Uniform Resource Locators). Ces adresses se basent sur l'emplacement du stockage des données.

Agent : Logiciel qui agit de façon autonome (automate). [94]

Attaque byzantine (ou panne byzantine volontaire) : Attaques volontaires visant à faire échouer le système, causant des pannes byzantines.

Blockchain : Suite de structures de données (des blocs) qui sont en écriture seule et en lecture seule, sans permission de modification ou de suppression. Les blocs de données sont distribués dans un réseau *peer-to-peer*, et chaque bloc contient la fonction de hachage cryptographique du bloc dont il provient. Le bloc précédent est donc utilisé pour créer un lien avec le bloc qui vient d'être créé, et les blocs sont liés entre eux. Ils forment une chaine de blocs. [5]

Capacité (*blockchain*) : taille de toutes les transactions précédemment effectuées que le mineur doit stocker.

Contrat intelligent (*Smart contract*) : Programme informatique servant de transaction numérique, créée, déployée et mise en oeuvre par des utilisateurs du réseau blockchain (mineurs). Il doit faciliter le fonctionnement de la blockchain en dehors du concept des cryptoactifs. Un arrangement de transfert de valeurs quelconques est encodé entre deux pairs d'une blockchain qui ne se font pas confiance et sans l'intervention d'un service tiers. Les utilisateurs interragissent avec les contrats intelligents en envoyant des transactions au registre, qui déclenche des appels de fonctions dans le contrat. Les transactions s'exécutent automatiquement à partir de règles prédéfinies lorsque certaines contraintes d'exécution sont remplies au sein du système. [104, 15]

CID (Content Identifier) : Identifiant de contenu. Une adresse unique pour un bloc de données dans l'IPFS qui est dérivé de son contenu. [29]

Concurrence de flux : Le nombre maximum de flux TCP utilisé simultanément par un seul hôte pour transférer du contenu.

DAG (Graphe Acyclique Dirigé) : Les blocs dans IPFS forment un graphique car ils peuvent pointer vers d'autres blocs par leur CID. Ces liens ne peuvent pointer que dans une direction (dirigée) et sur l'ensemble du graphe, il n'y a pas de boucles ou de cycles (acycliques).

Durée de flux : L'intervalle de temps entre le début et la fin d'un flux TCP.

Epinglage : Fonctionnalité permettant de rendre du contenu permanent dans l'IPFS en signalant au *garbage collector* de ne pas supprimer les fichiers épinglés.

Fog/Edge Computing : infrastructure chargée de stocker et de traiter des données issues d'objets connectés. Le *Fog computing* renvoie à l'architecture ; l'*Edge computing* fait référence aux terminaux de traitement.

Graphe : abstraction mathématique utilisée pour représenter les relations entre une collection d'objets. Le plus souvent, les graphiques sont utilisés pour représenter les relations par paires entre les objets (exemple : organisation structurelle hiérarchique).

Graphe dirigé : Un graphe est dit orienté si chaque arête a un sens de direction. Les relations entre les nœuds ne s'associent correctement que dans une direction, et cette direction est indiquée par une flèche à une seule tête. Des termes généalogiques tels qu'ancêtre, descendant, parent (racine) et enfant (feuille) sont fréquemment utilisés pour désigner les nœuds dans un graphe orienté.

Graphe acyclique : Un graphe est appelé acyclique s'il n'y a pas de boucles dans le graphe - c'est-à-dire, étant donné n'importe quel nœud dans le graphe, il n'y a aucun moyen de naviguer de ce nœud vers lui-même le long des bords du graphe.

Hors-chaine (ou *off-chain*) (stockage) : Stockages de données traditionnels (en dehors de la blockchain), soit des stockages de fichiers soit des bases de données de différents types [48].

Hypermedia : Extension de l'hypertexte à des données multimédias, pour ajouter une surcouche de médias (images, sons, vidéos, données multimédia) aux données hypertexte.

Hypertexte : Document ou ensemble de documents informatiques accessibles à partir d'un système de liens hypertextes (ou hyperliens).

Liquid democracy : Les nœuds sont récompensés pour leur participation aux votes et leurs propositions sur le réseau, plutôt que de l'être pour une preuve de travail qui dépend de la puissance de CPU mise à disposition. [73]

Loi de puissance : Relation mathématique entre deux quantités, où un changement relatif d'une quantité entraîne un changement relatif proportionnel de l'autre quantité.

Minage (*blockchain*) : processus de création de nouveaux blocs, qui sont ajoutés à la fin de la *blockchain* [5].

Nœud (d'un graphe/arbre) : Désigne un objet dans un graphe.

Nonce : Chaine de bits, ou nombre aléatoire utilisé pour vérifier les hachages cryptographiques uniques d'un bloc afin de maintenir l'intégrité d'une blockchain de bout en bout [5].

Pair : participant dans un réseau décentralisé. Les pairs sont des participants tout aussi privilégiés et équipotents dans une application Peer-to-Peer.

Preuve de l'espace-temps (PoSt ou *Proof-of-Spacetime* : Algorithme qui permet aux fournisseurs de stockage de prouver aux clients que leurs données sont stockées durant une période spécifique. [16, 15]

Preuve de réplication : Algorithme qui permet aux fournisseurs de stockage de prouver aux clients qu'aucun nœud du réseau n'a dupliqué les données et qu'elles sont sécurisées dans leurs emplacements de stockage physiques dédiés. [16]

Preuve de stockage (PoS ou *Proof-of-Storage* : Algorithme qui permet aux fournisseurs de stockage de prouver aux clients qu'ils ont stocké les données pour lesquelles ils ont été payés à partir de preuves que les pairs du réseau vérifient. [15]

Pair-à-pair (système) : Un système pair-à-pair(*peer-to-peer* ou P2P), est un système d'égal à égal, dont aucun nœud n'est privilégié [14].

Panne byzantine : Comportement d'un système qui ne respecte pas des spécifications et qui donne des résultats inattendus. Elles peuvent être volontaires (attaque byzantine) ou involontaires (défaut/défaillance/échec). Dans le second cas, l'échec peut être lié au protocole de réponse face aux situations réseau, au consensus, à la validation, à la vérification des données.

Propagation par inondation : Un pair interroge tous ses voisins, et ceux-ci propagent la requête de proches en proches à leur tour avec des métadonnées liées à la requête. Ceci afin de permettre de poursuivre la diffusion ou l'arrêt de la propagation.

Réseau superposé réseau en surcouche d'un autre réseau, dont les noeuds sont interconnectés par des liens logiques du réseau sous-jacent. La complexité du réseau sous-jacent n'est pas visible par le réseau superposé.

Service web : programme logiciel permettant de traiter, manipuler, de sauvegarder des données, etc.

Service cloud : Service informatique de partage de ressources à la demande (logiciels, serveurs, stockage de données) accessible sur internet par l'intermédiaire d'un service web. Les données sont stockées et traitées de manière centralisée ou décentralisée. L'architecture de ce type de stockage n'est pas systématiquement distribuée. [45, 16]

Stockage cloud : Système répondant à un besoin de gestion de ressources de stockage, et disponible à l'aide de *services cloud*. Il facilitent la coopération entre des objets connectés à travers différentes formes d'applications [16].

Système de stockage décentralisé : groupe de personnes mettant à disposition les composants de stockage disponibles de leurs machines [16].

Système multi-agents (SMA) : système composé d'un ensemble d'*agents*. [94]

Temps de latence de flux : Intervalle de temps entre deux arrivées de flux consécutives. [11]

Tolérance aux pannes byzantines : capacité à résister aux défauts byzantins dans un système distribué.

Torrent : Ensemble des fichiers échangés par le protocole BitTorrent.

Volume de transfert : Le nombre total d'octets échangés entre deux hôtes pendant leur période d'activité.

Annexe B

Problématisation

B.1 Spécialisation du sujet depuis le sujet initial

Le sujet initial du travail de recherche « Le protocole IPFS » proposé pour la réalisation du mémoire est un thème de recherche étendu et qui m'était initialement totalement inconnu, ainsi que son contexte. Il a d'abord été convenu avec mon promoteur que j'analyse ce sujet en profondeur afin de comprendre dans un premier temps : le protocole IPFS, son contexte, ses avantages, ses inconvénients, ses applications dans un contexte de stockage décentralisé, et que je le compare avec d'autres protocoles et les moyens de stockage décentralisés existants.

Il aurait été difficile d'identifier une question de recherche initiale sans documentation préalable avec l'objectif d'apporter une vraie contribution à la recherche scientifique, sans comprendre tout le contexte extérieur au protocole IPFS et les sujets qui y sont associés.

B.1.1 Problématique de l'état de l'art

Le sujet « Le protocole IPFS » proposé pour le travail de mémoire, ainsi que la lecture d'articles scientifiques ont pu mener vers deux propositions de questions initiales qui pouvaient amener vers une première problématique pour la rédaction de l'état de l'art : « Quelles sont les solutions existantes permettant de faire du stockage décentralisé ? », ou « Quels sont les moyens de stockage décentralisés existants ? ». Cependant, mon promoteur m'a déconseillé ces questions afin de ne pas faire un recensement des solutions décentralisées. L'objectif du sujet est d'analyser l'intérêt d'utiliser IPFS par rapport aux autres systèmes semblables dans un contexte de stockage décentralisé et de mesurer ses performances. Après plusieurs échanges et certaines lectures sur le sujet d'IPFS, le *stockage décentralisé* dans le cadre d'IPFS a été identifié comme un thème de question de problématique plus précis.

Une question telle que « Quelles sont les motivations pour l'adoption d'une solution de stockage distribué ? » ; « Quels sont les principes de mise en oeuvre d'une solution de stockage distribué ? » ; « Quelles sont les difficultés et défis liés la mise en oeuvre d'une solution de stockage distribué ? » ont semblé peu pertinentes afin de pouvoir comprendre et expérimenter le contexte d'IPFS. En effet, le spectre de telles questions était trop étendu et peu relatif au sujet de l'InterPlanetary File System, mais tout en utilisant un terme très précis qu'est le « stockage distribué ». A l'aide d'un tel mot-clé, le nombre d'articles recherchés aurait été plus restreint, mais cela n'aurait pas permis de comprendre tout le cheminement qui mène à ce type de sujet alors que le thème de l'IPFS est relatif à un thème plus vaste qu'est le stockage décentralisé.

L'objectif exprimé par mon promoteur étant de comprendre le contexte du protocole IPFS, son fonctionnement, ses avantages, ses limites, ses performances et éventuellement ses différences avec d'autres systèmes de stockage distribués, la problématique « Quelles sont les solutions apportées par l'IPFS ? » a été retenue pour l'état de l'art.

Un suivi chronologique a été établi qui résume les étapes clés des échanges avec mon promoteur pour la résolution de la recherche B.1.3.

D'autre part, les mots-clés utilisés pour effectuer la recherche dans la littérature scientifique ont été répertoriés dans la section B.2.2.

B.1.2 Aboutissement à la question de recherche

Après avoir réalisé l'état de l'art qui présente le contexte d'utilisation de l'IPFS dans le cadre du stockage décentralisé pour une distribution complète du web, il convenait de trouver une approche que la littérature ne présente pas à propos du protocole IPFS et de son système de fichiers.
Un manque identifié dans la littérature scientifique était la manière dont l'IPFS peut être utilisé (« Comment ? »), ainsi que le cheminement qui peut mener un informaticien à l'utiliser (« Pourquoi ? »).

La question de recherche identifiée est donc « Pourquoi et comment mettre en œuvre une solution de stockage décentralisé ; le cas d'IPFS ? ». La question comprend plusieurs sous-questions qui permettent notamment d'aider à y répondre : « Pourquoi faire du stockage décentralisé ? » ; « Quelles solutions de stockage décentralisé existent ? » ; « Quels défis ces solutions soulèvent-elles ? » ; « Quels sont les principes de fonctionnement d'IPFS ? » ; « Quelles sont les performances d'IPFS ? ».

Afin de réaliser la recherche il a été convenu avec mon promoteur qu'il serait intéressant de connaître les possibilités qu'apporte le protocole IPFS. Cela peut être identifié en démontrant comment l'utiliser dans un contexte de développement logiciel, ainsi que d'identifier dans quel contexte il est pertinent de l'utiliser. Créer un *Proof of concept* a permis d'identifier ce qu'il est possible de faire avec le système. Le tout a donc mené au titre principal du mémoire : « Motivations, défis et fonctionnement du stockage distribué : le cas de l'InterPlanetary File System ».

B.1.3 Chronologie des échanges avec mon promoteur

Date	**Description**	**Prise en compte des conseils**
Juin - Juillet 2021	Premiers contacts avec Monsieur Colin pour connaître les attentes du sujet du mémoire relatif au Protocole IPFS. Le sujet initial étant vaste et m'étant totalement inconnu, il a d'abord été convenu avec mon promoteur que j'analyse d'abord ce sujet en profondeur afin de me familiariser avec le sujet et de pouvoir identifier des questions de recherches. Il m'est conseillé de ne pas m'étendre sur un sujet trop ambitieux, même si le sujet peut mener à de nombreux thèmes (implémentation de stockage au-dessus de IPFS ; évaluation de ses performances ; regard critique sur les fonctionnalités et les performances ; la relation avec la blockchain)	Recherche de littérature scientifique depuis IEEEXplore, ACM.org et Google Scholar afin de comprendre le protocole IPFS et son contexte, ses avantages et inconvénients, ses applications dans un contexte de stockage décentralisé et que je le compare avec d'autres protocoles et les moyens de stockage décentralisés existants. Sans me documenter initialement, il aurait été difficile d'identifier une question de recherche initiale avec l'objectif d'apporter une vraie contribution à la recherche scientifique, sans comprendre tout le contexte extérieur au protocole IPFS, et les sujets qui y sont associés. J'ai donc récupéré des articles à partir de plusieurs mots clés. Identification de mots clés pour la recherche.
Novembre 2021	Premières propositions de thèmes de mémoire afin de pouvoir concentrer la recherche (blockchains, moyens de paiements avec IPFS et évaluation des performances). Il m'est conseillé d'être prudent avec un tel sujet qui possède plusieurs facettes. Les systèmes de paiement ont besoin de sécurité et il ne faudrait pas proposer un sujet du risquerait de me piéger.	Poursuite des lectures sur le sujet global d'IPFS. Remise à plat des idées de sujet et d'implémentation.

Janvier - Février 2022	Première proposition de question pour l'état de l'art : "Quels sont les moyens de stockage décentralisés existants ?" Mon promoteur déconseille cette question trop générale, car cela conviendrait à faire un simple inventaire qui serait difficile à comparer (le choix de critères, choix de comparaison, manière d'évaluer ces critères). Il est conseillé de l'orienter vers un sujet plus en ligne avec le sujet principal du mémoire (IPFS). Un échange Teams est organisé afin de discuter d'une question plus en lien avec la description du protocole et de son implémentation. Il m'est conseillé de ne pas tenir compte uniquement des articles publiés dans des revues.	Recherche d'une problématique pour l'état de l'art permettant de décrire le fonctionnement d'IPFS, les contraintes du système et les services proposés, ainsi que de comprendre l'utilité du stockage décentralisé, son fonctionnement et les différences entre ces systèmes. Les mots-clés de la recherche sont affinés.
7 Avril 2022	Proposition d'idées d'un sujet de mémoire, l'orientant vers un scénario dans différents environnements qui seraient comparés sur différents critères afin de mesurer l'efficacité du système IPFS. L'idée présentée est l'utilisation de l'IPFS dans un scénario précis, lié au calcul distribué et à l'utilisation de la *blockchain* afin de rémunérer les contributeurs du système. Mon promoteur me signale que le sujet de la *blockchain* risque de m'embarquer très loin, mais que l'approche d'un scénario avec une comparaison dans plusieurs environnements est une bonne approche. La partie mémoire proposerait un *proof of concept* en prenant IPFS comme un 'exemple' d'un système de stockage distribué, dès lors que dans l'état de l'art les autres approches disponibles seraient mentionnées. Mon promoteur soulève que plusieurs métriques peuvent être prises en compte dès lors que l'on compare des systèmes centralisés et décentralisés (elapsed time, cpu time, network traffic) et qu'il faudra éclaircir ce point. Nous convenons du fait qu'il faut une problématique permettant de réaliser une étude approfondie sur IPFS : ce que c'est, comment cela fonctionne, pourquoi un tel système.	Remise à plat des idées de scénarios et recherche de questions de recherches. Afin de rester relativement général, des sujets trop ciblés été écartés dans le contexte de l'IoT ; les réseaux 4G ; 5G ; 6G ; les réseaux sociaux ; le domaine du forensic ; le domaine biomédical ; la bioinformatique ; le biohacking ; la biométrie ; l'intelligence artificielle ...
17-19 Avril 2022	Après plusieurs échanges, mon promoteur et moi convenons de plusieurs sujets potentiels. Les problématiques sous-jacentes retenues : — "Quelles sont les motivations pour l'adoption d'une solution de stockage distribué ?" — "Quels sont les principes de mise en oeuvre d'une solution de stockage distribué ?" (en restant au niveau générique) — "Quelles sont les difficultés et défis liés la mise en oeuvre d'une solution de stockage distribué ?" — "Quelles sont les solutions apportées par le protocole IPFS ?" Sujet de mémoire : "Motivations, défis et fonctionnement du stockage distribué : le cas de l'IPFS". Un échange en direct est organisé sur Microsoft Teams. Il m'est demandé de faire un plan de mémoire et de choisir une problématique, en plus de choisir d'un sujet de mémoire définitif.	Etablissement d'un plan de mémoire.

2 Mai 2022	Proposition d'un plan de mémoire à mon promoteur. Il le valide, mais relève certaines incohérences concernant l'ordre des étapes. Question de problématique pour l'état de l'art : "Quelles sont les solutions apportées par l'IPFS ?". L'objectif est de présenter : — Définir le web distribué et sa raison d'exister — Citer des systèmes existants pour le partage de fichiers et leurs objectifs — d'expliquer les protocoles P2P et énumérer leurs objectifs — de mettre une priorité sur l'IPFS en expliquant ce qu'il permet, son fonctionnement, ses avantages, ses limites, ses performances et le comparer avec d'autres systèmes Le mémoire doit avoir pour objet une utilisation plus précise du protocole IPFS. Sujet du mémoire confirmé : "Motivations, défis et fonctionnement du stockage distribué : le cas de l'IPFS".	Corrections de l'ordre des étapes dans le plan du mémoire et début de la rédaction en réponse à la problématique de l'état de l'art.
fin Mai 2022	Envoi du brouillon de l'état de l'art et attente de validation. Il m'est conseillé d'ajouter un paragraphe à propos du *White-paper* sur la solution ThreeFold , ainsi que de parler des problématiques soulevées par la *Decentralized Identity Foundation* . Rédaction de la partie mémoire.	Ajout de sections dédiées ThreeFold et les identités décentralisées. Des *White-papers* et articles associés sont ajoutés.
Juin 2022	Questions pour validation des questions de recherche et de la problématique. Il m'est conseillé de faire simple afin de comprendre pourquoi on a voulu construire IPFS, comment cela fonctionne, dans quels cas il peut être utile ou non.	La question de recherche utilisée est "pourquoi et comment mettre en oeuvre une solution de stockage décentralisé ; le cas d'IPFS". Elle implique plusieurs sous-questions : "Pourquoi du stockage décentralisé ?" ; "Quelles solutions existent ?" ; "Quels défis ces solutions soulèvent-elles ?" ; "Quels sont les principes de fonctionnement ? (IPFS)" ; "Quelles performances ?"
Juillet 2022	Envoi d'une version brouillon du mémoire pour valider la structure, la table des matières et certains contenus. Il est conseillé d'être plus précis concernant l'ajout de certains détails précis dans le contenu existant et d'arranger la structure du texte. Certains termes ambigüs sont présents dans le texte et il est demandé que j'ajoute des schémas supplémentaires pour rendre la lecture moins lourde.	Des schémas et améliorations du texte sont ajoutés. Le texte est amené vers la version finale.
Août 2022	Envoi de la version finale du mémoire.	

TABLE B.1 – Chronologie des échanges avec mon promoteur et étapes clés pour la réalisation du mémoire

B.2 Méthodologie de recherche de littérature scientifique

Cette section présente un examen transparent et reproductible de la méthodologie de recherche de la littérature scientifique qui a permis d'obtenir des articles pour la réalisation de l'état de l'art. Ces articles relatent des sujets très généraux que sont les systèmes décentralisés, le stockage décentralisé et le protocole IPFS. Ils ont par ailleurs amené vers des sujets supplémentaires liés aux systèmes distribués, aux systèmes pair-à-pair, aux *blockchains*, etc. Les articles scientifiques étant très nombreux dans ces domaines, il a été nécessaire de trouver des mots-clés permettant de cadrer un maximum le nombre de résultats, ainsi que de trouver des critères d'acceptation arbitraires lors de la lecture d'articles et de revues. Après un certain nombre de lectures, les informations devenant répétitives, il a pu être décidé qu'il n'était pas nécessaire d'utiliser davantage d'articles pour la rédaction de l'état de l'art ou du travail de recherche.

B.2.1 Ciblage des articles

Les articles recherchés ont été ciblés selon certains critères :

- Recherche d'articles de la *littérature blanche* rédigés en anglais provenant de *Springer*, *ACM*, *Science Direct*, *IEEE Xplore*.
- Recherche prioritaire d'articles scientifiques, de recueils d'articles uniquement, et de *review/survey* si ils sont suffisamment exhaustifs.
- Pour l'analyse et la rédaction de l'état de l'art, rechercher de sources qui parlent soit de *systèmes de stockage décentralisé*, soit de *protocoles de stockage décentralisé*. Avec un ciblage sur IPFS.
- Recherche d'articles qui abordent principalement le sujet du stockage décentralisé non spécialisé, la description du fonctionnement des différentes solutions, les différents algorithmes associés et leur comparaison.
- Des articles obtenus par *snowballing* sont des *white papers* issus de la littérature grise.
- Les articles de revue ont été favorisés par rapport aux articles non publiés afin d'essayer d'éliminer des articles moins pertinents compte tenu de la quantité de résultats importante relative à la thématique.
- L'année de publication est postérieure à l'année 2010, jusqu'à 2022, pour les systèmes et protocoles liés au *stockage décentralisé* utilisés en exemples et pour la description des systèmes de *stockage décentralisé*.
- Des articles obtenus par snowballing datent des années 1980. Accepter ponctuellement des sources datant de 1980 à 2010 pour des informations susceptibles d'augmenter la qualité d'une description d'un concept abordé avait donc du sens. Surtout lorsque l'article était cité par plusieurs articles.
- L'article analysé doit posséder un contenu et une conclusion riches et sans incohérences. Il doit être alimenté d'un nombre de références suffisant en fonction du nombre de page qu'il possède.
- Sauf exception, l'article possède un nombre de 40 pages maximum.
- S'il s'agit d'un résultat de recherches scientifiques d'une nouvelle méthode/d'un concept, ou tout autre outil utile pour le sujet mais indirectement lié au *stockage décentralisé*, le nombre de citations peut être moins important mais il doit être utilisé uniquement pour être cité s'il est peu cité, ou avec parcimonie si son nombre de citations est peu élevé.
- Si l'article a été peu cité mais qu'il a été téléchargé un grand nombre de fois depuis une bibliothèque numérique, il peut rester pertinent en fonction de la richesse de son contenu et de la renommée de son auteur.
- Les articles utilisé par *snowballing* sont doivent préférentiellement provenir d'une source de recherche (ACM, IEEE, Springer, Science Direct), mais s'ils sont vraiment pertinents, ils peuvent provenir d'une autre source que ces dernières si la source de référence est une source de recherche.

B.2.2 Mots clés recherchés

Les recherches d'articles scientifiques ont été faites avec l'outil de recherche avancée de *Google Scholar* afin de pouvoir rechercher des articles ciblés simultanément sur plusieurs moteurs de recherche de sites proposant de la littérature scientifique. Google Scholar renvoie en nombreux résultats, j'ai donc ciblé ma recherche prioritairement sur les articles publiés dans de revues scientifiques.

Les mots-clés sont recherchés et les résultats de recherches sont :

Mots-clés de la recherche	**Résultats**
"decentralized" + "distributed" + "data storage" + ("system" OR "protocol") + (site :acm.org OR site :ieeexplore.ieee.org OR site :springer.com OR site :sciencedirect.com) Seuls les articles rédigés en anglais, datant entre 2010 et 2022 ont été ciblés.	Le nombre de résultats est de 14400 articles. En ciblant uniquement les articles de revues scientifiques, le nombre de résultats est de 920 articles.
("ipfs" OR "p2p" OR "peer-to-peer" OR "peer to peer") + "decentralized" "distributed" "data storage" + ("system" OR "protocol") + (site :acm.org OR site :ieeexplore.ieee.org OR site :sciencedirect.com OR site :springer.com)	7200 résultats, dont 465 résultats articles de revues.

De la même manière, afin de réduire le nombre d'articles et étant donné que le sujet principal est le protocole IPFS et que je dois comparer ce système avec d'autres moyens de stockage distribués, j'ai modifié les mots clés comme suit : "decentralized" + "data" + "storage" + ("system" OR "protocol") AND (site :acm.org OR site :ieeexplore.ieee.org OR site :springer.com OR site : sciencedirect.com).	Le nombre de résultats est de 3300, et en ciblant les articles de revue, le nombre de résultats est de 238 articles.
Pour cibler d'avantage la recherche, j'ai ajouté le mot clé « ipfs » à la recherche : "ipfs" "decentralized" "distributed" "data storage" + ("system" OR "protocol") + (site :acm.org OR site :ieeexplore.ieee.org OR site :springer.com OR site :sciencedirect.com).	Le nombre de résultats total est de 1100 articles. Le nombre total d'articles de revue est de 77.
L'utilisation du snowballing pourrait aider à trouver des articles pertinents par la suite. La recherche de survey/review pourrait également aider à trouver des systèmes/protocoles à comparer : ("decentralized" + "data storage") + ("system" OR "protocol") AND (site :acm.org OR site :ieeexplore.ieee.org OR site :springer.com OR site :sciencedirect.com)	15700 résultats, dont 464 résultats articles de revues. L'ajout des mots clés « ("survey" OR "review") AND ... » permettent de réduire ces articles à 11000 résultats et 451 articles de revues.
allintitle : "decentralized" "storage" (site :acm.org OR site :ieeexplore.ieee.org OR site :sciencedirect.com OR site :springer.com)	213 résultats, dont 1 article de revues.
"decentralized storage" + "distributed" + ("system" OR "protocol") + (site :acm.org OR site :ieeexplore.ieee.org OR site :sciencedirect.com OR site :springer.com)	114 résultats
"decentralized storage system" + (site :acm.org OR site :ieeexplore.ieee.org OR site :sciencedirect.com OR site :springer. com)	9 résultats
"decentralized data storage" + (site :acm.org OR site :ieeexplore.ieee.org OR site :sciencedirect.com OR site :springer.com)	33 résultats
("distributed" AND "decentralized data storage") OR ("decentralized" AND "distributed data storage") + ("system" OR "protocol") + (site :acm.org OR site :ieeexplore.ieee.org OR site :springer.com OR site :sciencedirect.com)	31 résultats
"ipfs" ("system" OR "protocol") AND (site :acm.org OR site :ieeexplore.ieee.org OR site :springer.com OR site :sciencedirect.com)	obtient 4110 résultats et 179 résultats d'articles de revues.
"ipfs" + "decentralized storage"+ ("system" OR "protocol") + (site :acm.org OR site :ieeexplore.ieee.org OR site :sciencedirect.com OR site :springer.com)	37 résultats
Plusieurs mots-clés attirent mon attention lors de mes lectures d'articles liés à l'IPFS. Je fais donc une nouvelle recherche afin d'identifier des articles supplémentaires potentiels : blockchain ipfs framework secure + "storage" AND (site :acm.org OR site :ieeexplore.ieee.org OR site :springer.com OR site :sciencedirect.com)	1760 résultats et 114 articles de revue.

TABLE B.2 – Mots-clés recherchés depuis Google Scholar

Au fur et à mesure des recherches à l'aide des mots-clés, de nombreux résultats réapparaissent à plusieurs reprises. Ces articles ont été repris pour être analysés et indexés.
De plus, lors du filtrage des articles, certains d'entre eux initialement étudiés ont finalement été considérés comme étant trop généraux, ou hors contextes. Ils sont dont mis de côté, et les articles relatifs à IPFS sont favorisés. Certains ont été écartés au fur et à mesure des lectures et de la rédaction de l'état de l'art.
Des articles ont également été obtenus par *snowballing* au fur et à mesure des lectures.
Les articles ciblés, généralement des White-papers, ont été recherchés de manière ciblée lorsqu'un mot clé devait être trouvé. Tel que "decentralized identity foundation", "Threefold", "Web3 foundation", ou encore

dans le cas de la recherche des identité décentralisées ("decentralized identity").
Enfin, au cours des lectures, les articles se sont montrés de plus en plus répétitifs. Le travail ne pouvant pas être trop long, des articles n'ont pas été jugés suffisamment utiles pour ajouter une plus-value en réponse à la problématique.
Un filtrage a pu être fait initialement afin de garder les articles qui ne se basent pas sur une analyse de cas d'utilisation du stockage décentralisé très ciblés et qui n'ont pas trait directement au contexte d'IPFS.
Le nombre d'articles retenus après le filtrage est de 108 articles.

B.3 Analyse et lecture approfondie

B.3.1 Extraire les informations et les thèmes clés de chaque manuscrit

Les articles retenus pour une analyse ont été lus, surlignés, indexés, catégorisés dans l'outil Zotero et résumés afin d'en extraire un maximum d'informations utiles en relation avec le sujet final.
Les thèmes clés et le type de données qu'abordent les articles analysés ont été marqués dans l'outil *Zotero* afin que je garde une vue rapide du contenu des articles en fonction de leurs catégories.

Pour la rédaction de l'état de l'art, j'ai tenté de résumer les articles à l'aide de questions principales. Les résumés obtenus étaient destinés à être agrégés vers des sujets précis afin de répondre à la problématique et aux différentes sous-questions de la question de recherche, pour enfin aboutir à une réponse à la question de recherche dans la conclusion du mémoire.
Les questions pour la rédaction de ces résumés sont les suivantes :

- Quel est le type du texte analysé (un recueil/*survey*/*review*/autre) ?
- Est-ce un une nouvelle description d'un protocole/système de stockage/système communication/un système déjà répertorié, un cas d'utilisation d'IPFS, une description approfondie d'IPFS, etc. ?
- De quel(s) système(s)/protocole(s) parle-t-on dans le texte ?
- Quel est le type de système étudié ?
- S'il y a un moyen de communication dans le système étudié, duquel s'agit-il ?
- Est-ce orienté pour une centralisation, une décentralisation partielle/totale ?
- Comment les système(s)/protocole(s) identifiés s'organisent-ils entre eux ?
- Comment se décomposent/catégorisent les système(s)/protocole(s) identifiés ?
- Quels sont les avantages et les inconvénients du système/protocole détaillé ?
- Quels sont les éléments clés qui décrivent le système/protocole ?
- Quels sont les domaines d'application du système/protocole étudié ?
- Quels défis sont soulevés par le(s) système(s)/protocole(s) étudié(s)
- Qu'apporte-t-il et quels sont les résultats ?

Pour un texte scientifique qui décrit l'IPFS, en complément des questions ci-dessus :

- Quelles sont les performances de l'IPFS ?
- Comment l'IPFS fonctionne/s'organise-t-il ?
- Quels sont les cas d'application d'IPFS ?

B.3.2 Choix des informations qui répondent aux critères de sélection

Comme expliqué précédemment, lors de recherches initiales, j'ai identifié des termes de recherche supplémentaires depuis une première liste d'articles analysés afin d'alimenter mon sujet et j'en ai récupéré des références supplémentaires à l'aide de mots-clés et de références vers d'autres articles.

J'ai obtenu une grande quantité de références, que j'ai filtrées en fonction :

- De leur fiabilité, car j'ai éliminé des sources après avoir fait des recherches sur la fiabilité/renommée de sources peu connues/citées.
- Du nombre de citations. J'ai néanmoins validé des articles peu cités pour lesquels les auteurs étaient forts présents la littérature scientifique, ou bien pour lesquels la présentation d'un sujet récent ou très ciblé pouvait être pertinent. Certains autres articles étaient parfois beaucoup cités, ce qui selon moi ajoute de la fiabilité. Les articles peu pertinents ont été écartés.
- De l'utilité des articles pour mon sujet en fonction de la plus-value qu'il pouvait ajouter. J'ai donc écarté de nombreux articles en suivant les répétitions des contenus déjà abordés dans d'autres articles.
- De la présence de citations/descriptions d'un article original B depuis un article A. Sur cette base, j'ai tenté de toujours comparer les textes entre l'article B et l'article qui y fait référence. Et si possible,

d'utiliser au moins un troisième article C afin d'identifier d'éventuelles contradictions dans l'article A ou C avec l'article B.
— Lors de la rédaction de l'état de l'art, j'ai vérifié que les informations ajoutées n'étaient pas en contradiction avec ce qui était déjà rédigé. Si c'était le cas, je recherchais des informations provenant d'autres sources pour corriger les erreurs et éventuellement écarter un article en erreur. Cela m'a permis de retravailler mes analyses à plusieurs reprises (processus itératif) et d'éliminer des sources non fiables avant de rédiger l'état de l'art.

B.3.3 Utilisation d'outils pour grader une vue des travaux utiles

Lors de la recherche et de la récupération d'articles scientifiques, *Zotero* a été utilisé afin d'indexer les différents articles et d'identifier d'éventuels doublons.
De plus, comme expliqué précédemment, durant mes analyses, j'ai maintenu les articles indexés dans *Zotero* à jour pour les organiser et les filtrer.

Cela m'a aidé à remplir plusieurs objectifs :
— Identifier l'article par son titre, son année de publication, sa source, sa fiabilité, le type de données, le nombre de citations, etc.
— Classer l'article suivant son type de contenu (description, type de communication, type de système, les sujets relatifs, *survey*, etc.).
— Filtrer les articles retenus/écartés.
— L'avancement du traitement de l'article.
— Identifier le lien avec un article source en cas de *snowballing*.
— Avoir une notion de fiabilité et de marqueurs permettant de garder une vue quant à l'utilisation ou non de l'article dans le travail.

Mise en évidence des meilleures pratiques de recherche dans la littérature scientifique

Lors de la préparation à la rédaction, il a été nécessaire que j'adapte mon organisation afin de travailler plus efficacement. Cela a été un travail conséquent, mais j'ai abouti à l'identification de certaines pratiques permettant une réalisation de l'analyse et de la rédaction de manière efficace :
— Pour la recherche, il est certain que plus le sujet est restreint, moins il sera compliqué d'avoir des recherches qui amènent à d'autres recherches en cascade. Il est donc primordial de rester dans le sujet et dans le cas d'une question de recherche trop vaste, il faut augmenter les critères de qualité de la recherche d'articles.
— Pour une collection d'articles, il est utile de les numéroter et de réaliser des résumés courts et rapides qui permettent de les classer.
— En parallèle, ajouter un outil permettant de classifier ces différents documents permet de les identifier et de pouvoir les comparer sur base de caractéristiques communes identifiées avec d'autres documents.
— Utiliser des classifications strictes, précises et les mettre à jour en continu est important.
— Utiliser strictement des marqueurs/codes couleurs pour la classification et pour identifier la fiabilité des textes permet d'avoir un aperçu visuel rapide et clair.
— En cas de *snowballing*, s'assurer qu'un lien est créé entre deux articles afin de pouvoir retracer sa recherche et pour remettre en question la pertinence des articles en cas de besoin.