

---

**DÉVELOPPEMENT D'UN PROTOCOLE RÉSEAU  
ÉCOÉNERGÉTIQUE BASÉ SUR QUIC POUR UN TRANSFERT  
PARALLÈLE ET DISTRIBUÉ DE DONNÉES ENTRE SITES**

---

**MÉMOIRE DE FIN D'ÉTUDES**

Présenté et soutenu par :

**ANZIE SEVERIN BRADLEY**

En vue de l'obtention du :

**DIPLOME D'INGÉNIEUR DE CONCEPTION, GÉNIE INFORMATIQUE**

Sous la supervision de :

**Pr. Thomas Djotio Ndié**

**Pr. Remous Aris Koutsiamanis**

Devant le jury composé de :

Président : **Pettang Chrispin**, Professeur des Universités

Rapporteur : **Thomas Djotio**, Maitre de conférence UY1

Examinateur : **Akam Denis**, Chargé de cours UY1

Invité : **Remous Aris Koutsiamanis**, Maitre de conférence IMT Atlantique Nantes

Année académique 2023-2024

Le 18 septembre 2024



**IMT Atlantique**  
Bretagne-Pays de la Loire  
École Mines-Télécom



---

DÉVELOPPEMENT D'UN PROTOCOLE RÉSEAU  
ÉCOÉNERGÉTIQUE BASÉ SUR QUIC POUR UN  
TRANSFERT PARALLÈLE ET DISTRIBUÉ DE  
DONNÉES ENTRE SITES

---

**MÉMOIRE DE FIN D'ÉTUDES**

Présenté et soutenu par :

**ANZIE SEVERIN BRADLEY**

En vue de l'obtention du :

**DIPLÔME D'INGÉNIER DE CONCEPTION, GÉNIE INFORMATIQUE**

Année académique 2023-2024

Le 22 septembre 2024

## DÉDICACE



## REMERCIEMENTS

Il est difficile de trouver les mots justes pour exprimer ma profonde gratitude envers toutes les personnes qui ont contribué à mon éducation et à la réalisation de ce mémoire. Ce projet n'aurait pas été possible sans le soutien inestimable de ceux qui m'entourent.

Je tiens à remercier chaleureusement mes encadreurs de stage, Mcf. **Remous Aris Koutiamanis**, Pr. **Jean-Marc Menaud** et Pr **Thomas Djotio Ndié**, pour leur accompagnement exceptionnel, leur patience et leurs conseils. Un merci particulier à **Aris**, dont le soutien indéfectible et la passion pour la recherche ont été des sources d'inspiration. Je suis également reconnaissant au Pr **Bouetou Bouetou Thomas**, chef du département de Génie Informatique, pour son écoute et son engagement envers ses étudiants.

Je remercie aussi l'ensemble des enseignants et du personnel pédagogique de notre département. Leur enseignement de qualité m'a permis d'acquérir des connaissances solides et d'affiner ma réflexion, essentielles à l'élaboration de ce travail.

Un immense merci à mes camarades de la promotion **GI2024**. Votre camaraderie et votre soutien mutuel ont rendu cette expérience mémorable. Ensemble, nous avons partagé des moments précieux qui resteront gravés dans ma mémoire.

Je souhaite aussi remercier mes amis et camarades de stage, **Menra Romial et Guimapi Céleste**, pour les moments passés et pour leur soutien tout au long de cette période. Merci également aux ingénieurs **Arnaud TAMO** et **Gabin Fodop** pour leurs conseils constructifs et leur aide précieuse durant mon stage. Un remerciement spécial à mon ami **Marc Djiala**, pour son soutien, son attention et les moments partagés.

Je tiens à exprimer une reconnaissance particulière à mon amie **Mekena Viannie** pour son soutien inconditionnel et ses conseils avisés, qui ont été d'une grande aide tout au long de ce projet. A mes **Mojañ** qui sont avec moi dans ce parcours depuis le début et avec qui j'ai partagé l'essentiel de mon temps, merci à vous mes frères.

Je ne saurais oublier ma famille, ma plus grande source de force et de motivation. Merci à ma mère, **Mine'e Mi Oba**, pour son amour inconditionnel ; à ma grand-mère, **Anzie Rose**, et à mon grand-père, **Oba Olle Théophile**, pour leur sagesse et leurs précieux conseils ; ainsi qu'à mon oncle, **Oyono Oba Peter**, pour son soutien. Une mention spéciale à mes frères et sœurs, dont le soutien a été inestimable. Votre foi en moi a été essentielle dans ma réussite.

Enfin, je dédie ce travail à toutes les personnes passionnées par la technologie et l'innovation,



qui m'ont inspiré à explorer de nouveaux horizons. Ce mémoire est le fruit de nombreuses influences et de l'engagement d'une communauté qui m'a soutenu à chaque étape.

Merci à tous pour votre précieuse contribution à ce projet.



## **TABLE DES MATIÈRES**

<b>Dedication</b>	i
<b>Acknowledgements</b>	ii
<b>Abstract</b>	vi
<b>Résumé</b>	vii
<b>Tableaux</b>	viii
<b>Figures</b>	ix
<b>Table of Contents</b>	xi
<b>Dedication</b>	i
<b>Remerciements</b>	ii
<b>Sigles et abréviations</b>	vii
<b>Glossaire</b>	viii
<b>Résumé</b>	x
<b>Abstract</b>	xi
<b>Liste des tableaux</b>	xii
<b>Table des figures</b>	xiii
<b>Introduction Générale</b>	1
<b>1 Concepts généraux et État de l'art</b>	4
1.1 Généralité . . . . .	5

## TABLE DES MATIÈRES

---

1.1.1	Définition et Concepts de Base . . . . .	5
1.1.2	Rappels sur les Protocoles de Transfert de Données . . . . .	5
1.2	Présentation du protocole QUIC . . . . .	10
1.2.1	introduction . . . . .	10
1.2.2	Principe de fonctionnement . . . . .	11
1.2.3	Les principaux concepts autour du protocole QUIC . . . . .	12
1.2.4	Extension de QUIC pour le multipath : MP-QUIC . . . . .	14
1.2.5	Etude comparative des protocoles de transport QUIC, TCP et UDP . . . . .	15
1.3	Dimension Énergétique dans les Protocoles de Transport Décentralisés . . . . .	16
1.3.1	Vers des Protocoles Écoénergétiques . . . . .	17
1.4	État de l'art . . . . .	17
1.4.1	Étude des Protocoles Existantes . . . . .	17
1.4.2	HTTP . . . . .	17
1.4.3	FTP . . . . .	18
1.4.4	SFTP . . . . .	19
1.4.5	GridFTP . . . . .	20
1.4.6	BitTorrent . . . . .	20
1.4.7	FDT . . . . .	21
1.4.8	MDTMFTP . . . . .	22
1.4.9	Limites des Protocoles Existantes . . . . .	22
1.4.10	Impact du Protocole QUIC . . . . .	23
1.4.11	Étude Comparative des protocoles de transfert de données existant . . . . .	23
1.5	Bilan du chapitre et Positionnement . . . . .	25
1.5.1	Bilan . . . . .	25
1.5.2	Positionnement . . . . .	25
<b>2</b>	<b>Conception du protocole DataStreamX</b>	<b>26</b>
2.1	Rappel du problème et des objectifs poursuivis . . . . .	27
2.1.1	Objectif général . . . . .	27
2.1.2	Modélisation mathématique du problème d'optimisation . . . . .	27
2.1.3	Contraintes du problème . . . . .	28
2.1.4	Complexité du problème . . . . .	29
2.2	Modélisation UML du protocole DataStreamX . . . . .	30
2.2.1	Spécification des exigences . . . . .	30
2.2.2	Analyse . . . . .	31
2.2.3	Conception détaillée . . . . .	43
2.2.4	Simulation et Validation . . . . .	50
2.3	Algorithmes d'orchestration du transfert des données . . . . .	51
2.3.1	Calcul des méta-paramètres statiques . . . . .	52

---



## TABLE OF CONTENTS

---

2.3.2 Sélection des relais . . . . .	52
2.3.3 Répartition optimale des segments . . . . .	54
<b>3 Implémentation et résultats</b>	<b>56</b>
3.1 Choix des technologies et des outils . . . . .	57
3.1.1 Les langages de programmation et les bibliothèques . . . . .	57
3.1.2 Les outils utilisés . . . . .	57
3.2 Implémentation du Protocole . . . . .	59
3.2.1 Architecture Modulaire . . . . .	59
3.2.2 Gestion des Connexions Multiplexées . . . . .	60
3.2.3 Calcul Adaptatif des Fenêtres et des Segments . . . . .	60
3.2.4 Répartition Dynamique des Segments de Données . . . . .	60
3.2.5 Gestion des Paquets . . . . .	61
3.2.6 Aspect énergétique . . . . .	61
3.3 Environnement de simulation . . . . .	62
3.3.1 Le serveur . . . . .	62
3.3.2 Les tâche . . . . .	64
3.3.3 Le réseau . . . . .	64
3.4 Résultats des test et simulation du Protocole . . . . .	64
3.4.1 Configuration du Benchmark . . . . .	64
3.4.2 Indicateurs de Performance . . . . .	65
3.4.3 Résultats du Benchmark . . . . .	65
3.4.4 Comparaison par rapport aux autres protocoles existants . . . . .	66
3.4.5 Analyse des Résultats . . . . .	66
<b>Conclusion Générale et Perspectives</b>	<b>69</b>
<b>Références bibliographiques</b>	<b>71</b>



## SIGLES ET ABRÉVIATIONS

<b>FTP</b>	File Transfer Protocol
<b>HTTP</b>	Hypertext Transfer Protocol
<b>IETF</b>	Internet Engineering Task Force
<b>IMT</b>	Institut Mines et Télécommunications
<b>IA</b>	Intelligence Artificielle
<b>IOT</b>	Internet Of Things
<b>IP</b>	Internet protocol
<b>ISO</b>	International Organization for Standardization
<b>ENSPY</b>	Ecole Nationale Supérieure Polytechnique de Yaoundé
<b>FDT</b>	Fast Data Transfer
<b>MDTMFTP</b>	Modified Data Transfer Model File Transfer Protocol
<b>NIC</b>	Network Interface Card
<b>QUIC</b>	Quick UDP Internet Connections
<b>RTT</b>	Round-trip Time
<b>SFTP</b>	Secure Shell File Transfer Protocol
<b>SMTP</b>	Simple Mail Transfer Protocol
<b>SSH</b>	Secure Shell
<b>TCP</b>	Transmission Control Protocol
<b>TLS</b>	Transport Layer Security
<b>UDP</b>	User Datagram Protocols

## GLOSSAIRE

**ACK (Acknowledgment)** Message envoyé par le récepteur d'un paquet pour confirmer la bonne réception de celui-ci. Utilisé dans les protocoles de communication pour garantir la fiabilité des transmissions.

**Bandé passante** Quantité de données qui peuvent être transmises sur une connexion réseau dans un temps donné. Elle est généralement exprimée en bits par seconde (bps).

**Buffer** Zone de mémoire utilisée pour stocker temporairement les données en transit entre deux entités, par exemple, entre un client et un serveur. Les buffers permettent de compenser les différences de vitesse de traitement entre l'émetteur et le récepteur.

**Client** Dans le contexte du protocole, il s'agit de l'entité qui initie une connexion avec un serveur pour demander ou recevoir des données.

**Connexions Multiplexées** Technique permettant de gérer plusieurs flux de données sur une même connexion réseau. Elle optimise l'utilisation de la bande passante en réduisant les temps d'attente et en augmentant la flexibilité des échanges.

**Contrôle de Flux (Flow Control)** Mécanisme utilisé pour gérer le taux de transmission des données entre deux entités afin de prévenir la surcharge des buffers et éviter la perte de paquets.

**Contrôle de Congestion** Méthode permettant de réguler l'envoi de données sur un réseau afin de prévenir et de gérer la congestion. Il ajuste dynamiquement le débit de transmission en fonction des conditions du réseau.

**Fenêtre de Transfert** Segment d'un fichier déterminé pour être envoyé en un seul bloc. Le protocole décrit divise les fichiers en plusieurs fenêtres, chaque fenêtre étant ensuite segmentée pour différents relais.

**MP-QUIC** Variante du protocole QUIC permettant d'utiliser plusieurs chemins de réseau simultanément pour le transfert de données. Améliore la résilience et la performance du protocole en multipliant les routes possibles pour les paquets.

**Packet (Paquet)** Unité de données transmise sur un réseau. Un paquet contient généralement une en-tête (avec des informations de contrôle) et une charge utile (les données elles-mêmes).



**Paquet de Contrôle** Type de paquet utilisé pour gérer la communication et la synchronisation entre les entités du protocole. Peut inclure des informations telles que des identifiants de connexion ou des notifications de fin de session.

**Paquet de Données** Paquet contenant la charge utile de la communication, c'est-à-dire les données effectivement transférées entre le client, le serveur, et les relais.

**Protocole de Transport** Ensemble de règles et de conventions qui régissent l'envoi et la réception de données entre deux entités communicantes. QUIC est un exemple de protocole de transport utilisé dans le protocole décrit ici.

**QUIC** Protocole de transport orienté connexion, conçu pour être plus performant que TCP en réduisant la latence et en intégrant le chiffrement dès le départ. Utilisé comme base pour le protocole décrit.

**QUIC-go** Implémentation du protocole QUIC en langage Go. Fournit une infrastructure pour le développement et le test de protocoles de transport basés sur QUIC.

**Relais** Entité intermédiaire dans le protocole décrit qui reçoit des segments de données du serveur central et les transmet au client. Le relais aide à répartir la charge et à optimiser le transfert des données.

**Round-Trip Time (RTT)** Temps total nécessaire pour qu'un signal soit envoyé et qu'un accusé de réception revienne. Mesure essentielle pour évaluer la performance des connexions réseau.

**Segment** Partie d'une fenêtre de transfert assignée à un relais spécifique. Les segments sont définis en fonction des performances de chaque relais pour optimiser le transfert des données.

**Sérialisation/Désérialisation** Processus de conversion des données en une forme qui peut être stockée ou transmise (sérialisation) et de conversion inverse pour les rendre exploitables (désérialisation).



## RÉSUMÉ

**L**es transferts de données à grande échelle entre centres de données posent des défis majeurs, notamment en termes de performances et de consommation énergétique. **DataStreamX** propose une approche décentralisée qui répartit les flux sur plusieurs points de connexion pour éviter les limitations liées à la centralisation des transferts. Grâce à une architecture distribuée et à des algorithmes adaptatifs, le protocole optimise la bande passante disponible et assure un débit stable même en période de forte demande. En parallèle, DataStreamX se distingue par son engagement écologique, en favorisant l'utilisation de relais alimentés par des énergies renouvelables pour réduire l'empreinte carbone des transferts. Les résultats expérimentaux démontrent une amélioration des performances de transfert tout en réduisant la consommation énergétique par rapport aux approches classiques, ce qui fait de DataStreamX une solution innovante pour les besoins croissants des infrastructures numériques.

**Mots clés** : Protocole de transfert de données, Protocole QUIC, Décentralisation, Optimisation énergétique, Architecture distribuée.

## ABSTRACT

**L**arge-scale data transfers between data centers face significant challenges, particularly regarding performance and energy consumption. **DataStreamX** offers a decentralized approach that distributes data flows across multiple connection points to overcome the limitations of centralized transfers. Through a distributed architecture and adaptive algorithms, the protocol optimizes available bandwidth and maintains stable throughput even during high-demand periods. Additionally, DataStreamX stands out for its ecological commitment, prioritizing the use of renewable energy-powered relays to reduce the carbon footprint of data transfers. Experimental results show improved transfer performance while reducing energy consumption compared to traditional methods, positioning DataStreamX as an innovative solution for the growing needs of digital infrastructures.

**Keywords :** Data transfer protocol, QUIC Protocol, Decentralization, Energy optimization, Distributed architecture

## **LISTE DES TABLEAUX**

1.1	Tableau comparatif des protocoles QUIC, TCP et UDP . . . . .	16
1.2	Comparaison des protocoles de transfert de données existant à QuicPlex . . . . .	24
3.1	Résultats du Benchmark . . . . .	66
3.2	Comparaison des protocoles de transfert de données existant à QuicPlex . . . . .	67

## TABLE DES FIGURES

1.1	Architecture client-serveur vs architecture P2P . . . . .	6
1.2	QUIC architecture.[4] . . . . .	11
1.3	QUIC Connection establishment 0-RTT and 1-RTT handshake.[4] . . . . .	12
1.4	QUIC stream flow control.[4] . . . . .	14
1.5	Processus de communication HTTP.[18] . . . . .	18
1.6	Fonctionnement modes actif et passif du protocole FTP.[23] . . . . .	19
2.1	<i>Diagramme d'activité global du protocole DataStreamX</i> . . . . .	32
2.2	<i>Diagramme de séquence DataStreamX</i> . . . . .	34
2.3	<i>Diagramme de séquence Ouverture de session</i> . . . . .	35
2.4	<i>Diagramme de séquence Refus de session</i> . . . . .	36
2.5	<i>Diagramme de séquence Acceptation de session</i> . . . . .	37
2.6	<i>Diagramme de séquence envoi de données</i> . . . . .	38
2.7	<i>Liste des signaux échangée dans DataStreamX</i> . . . . .	39
2.8	<i>Interfaces des signaux échangés dans DataStreamX</i> . . . . .	40
2.9	<i>Structures de Composant des entités de base de DataStreamX</i> . . . . .	40
2.10	<i>Structure interne des composants DataStreamX</i> . . . . .	41
2.11	<i>Structures de Composant des entités de base de DataStreamX</i> . . . . .	42
2.12	<i>Architecture fonctionnelle du protocole DataStreamX</i> . . . . .	43
2.13	<i>Structures d'un échange entre deux composants DataStreamX</i> . . . . .	44
2.14	<i>Ouverture de session</i> . . . . .	46
2.15	<i>Refus de session</i> . . . . .	47
2.16	<i>Acceptation de session</i> . . . . .	47
2.17	<i>Envoi de données</i> . . . . .	48
2.18	<i>Structures d'un paquet DataStreamX</i> . . . . .	49
2.19	<i>Simulation et validation de DataStreamX</i> . . . . .	50
2.20	<i>Simulation et validation de DataStreamX (suite)</i> . . . . .	51

## INTRODUCTION GÉNÉRALE

### CONTEXTE

**L**e transfert de grands volumes de données entre centres de données, dans des secteurs tels que le Cloud Computing, l'Internet des Objets (IoT), le Big Data et l'Intelligence Artificielle (IA), représente un défi considérable. L'augmentation exponentielle des échanges de données, associée à la nécessité croissante de rapidité et d'efficacité, soulève des préoccupations majeures en termes de performance et d'impact environnemental. Bien que les avancées en matière de bande passante permettent désormais d'atteindre des débits de plusieurs centaines de Gb/s, la centralisation des transferts de données dans les infrastructures actuelles continue de limiter l'efficacité globale. Cette centralisation impose une récupération des données, souvent stockées sur des disques distribués, vers un point unique avant leur transfert. Ce processus crée des limitations structurelles qui ralentissent les flux de données et réduisent l'efficacité des opérations, en particulier lors de transferts massifs.

Parallèlement, les questions environnementales ont pris une importance cruciale. L'optimisation de la consommation énergétique des infrastructures numériques est désormais indispensable, surtout à mesure que les centres de données se multiplient et consomment d'énormes quantités d'énergie. La transition vers l'utilisation de sources d'énergie renouvelables et l'optimisation des infrastructures pour minimiser l'empreinte carbone deviennent des priorités dans la conception des solutions de transfert de données. Il est donc impératif de développer des protocoles qui non seulement améliorent les performances de transfert, mais qui répondent également aux exigences d'efficacité énergétique pour une meilleure durabilité.

Face à ces défis, il est nécessaire de repenser les protocoles actuels, en particulier ceux basés sur la centralisation, et d'adopter des approches décentralisées capables de maximiser la bande passante tout en optimisant l'utilisation des énergies renouvelables. C'est dans ce contexte que DataStreamX propose un protocole distribué, visant à surmonter ces obstacles et à ouvrir la voie à des transferts de données plus performants et respectueux de l'environnement.



## PROBLÉMATIQUE

Malgré les progrès substantiels dans les technologies réseau, notamment en matière de bande passante et d'interconnexion des centres de données, la centralisation des transferts de données constitue toujours un obstacle majeur à l'optimisation des performances et de l'efficacité énergétique. La centralisation limite les débits, crée des points de congestion et accroît la consommation énergétique des infrastructures. Cela soulève plusieurs questions critiques :

- **Comment concevoir un protocole de transfert de données distribué et parallèle capable de surmonter les limitations structurelles liées à cette centralisation ?**
- **Comment maximiser l'efficacité énergétique des transferts tout en assurant des performances élevées, notamment en termes de débits ?**
- **Comment optimiser la répartition des flux de données en prenant en compte l'impact énergétique des relais, en favorisant l'utilisation de sources d'énergie renouvelables ?**

Ces problématiques sont au cœur des infrastructures numériques modernes. Elles nécessitent des solutions qui concilient la performance des transferts de données avec une gestion efficace de l'énergie, dans un cadre où la demande de rapidité et d'évolutivité est croissante, mais où l'impact environnemental doit être minimisé.

## OBJECTIF

L'objectif principal de ce travail est de développer un protocole de transfert de données distribué, basé sur le protocole QUIC, permettant de surmonter les limitations liées à la centralisation des transferts et d'améliorer l'efficacité énergétique. Ce protocole vise à maximiser les performances de transfert tout en minimisant l'empreinte carbone des infrastructures en favorisant l'utilisation de relais alimentés par des sources d'énergie renouvelables. Pour atteindre cet objectif global, trois sous-objectifs guideront notre démarche :

- ☛ **La décentralisation des transferts de données :** Concevoir un protocole de transfert de données décentralisé qui évite les goulets d'étranglement liés à la centralisation, permettant un transfert parallèle, rapide et fiable des données. Il s'agira de développer une répartition intelligente des segments de fichiers entre différents relais en fonction de leurs capacités réseau.
- ☛ **L'optimisation dynamique des flux de données :** Développer et mettre en œuvre des algorithmes de synchronisation des flux de données capables de s'adapter aux fluctuations de la bande passante. Ces algorithmes devront coordonner les interfaces réseau des relais et des clients pour garantir une transmission fluide même en présence de variations dans la disponibilité des ressources.





- ☛ **La sélection énergétique des relais :** Mettre en place des mécanismes de sélection des relais en fonction de leur consommation énergétique et de leur source d'énergie. L'objectif est de favoriser les relais utilisant des sources d'énergie renouvelables, dans le but de réduire l'empreinte carbone tout en maintenant des performances de transfert élevées.

## PLAN DU MÉMOIRE

Ce mémoire est structuré en trois principaux chapitres :

- ☛ **Chapitre 1 Généralités et État de l'art :** Dans premier ce chapitre, nous présentons les concepts fondamentaux des protocoles de transfert de données, nécessaires à la compréhension des enjeux liés à la centralisation et de la solution proposée. Une introduction détaillée au protocole QUIC et à la notion de multipath sera fournie, suivie d'une étude fonctionnelle des protocoles de transfert existants.
- ☛ **Chapitre 2 Conception du protocole DataStreamX :** Ce chapitre est dédié à la conception du protocole. Nous y développerons un modèle mathématique permettant de résoudre le problème d'optimisation lié aux transferts décentralisés. Une modélisation détaillée du protocole à l'aide du langage UML sera proposée, accompagnée d'une description des différents algorithmes utilisés pour coordonner les transferts parallèles et garantir une répartition efficace des flux de données entre les relais. L'optimisation énergétique et la gestion des connexions y seront également abordées.
- ☛ **Chapitre 3 Implémentation et évaluation résultats :** Ce dernier chapitre détaille l'implémentation du protocole DataStreamX. Nous y présentons les technologies choisies, l'environnement de développement et de test, ainsi que la méthodologie employée. Les résultats expérimentaux obtenus seront discutés, notamment en termes de performances de transfert et de réduction de la consommation énergétique, comparés aux méthodes classiques.

Nous terminerons par une conclusion générale, qui fait un bilan de ce travail, ressort ses limites et fournit quelques perspectives.



## CONCEPTS GÉNÉRAUX ET ÉTAT DE L'ART

**D**ans ce premier chapitre, nous présentons les concepts de base nécessaires à la compréhension du protocole proposé. Nous aborderons les fondamentaux des **protocoles de communication**, en particulier les caractéristiques du **protocole QUIC** et son importance dans l'amélioration des performances des transferts de données. Les notions de **débit**, de **latence** et de **multipath** seront également détaillées. Nous réaliserons une analyse des principaux **protocoles de transfert de données** actuellement utilisés, en mettant l'accent sur leurs forces et limitations, et nous conclurons par une **étude comparative**.



## 1.1 Généralité

### 1.1.1 Définition et Concepts de Base

Les **protocoles** réseau sont les ensembles de **règles** et de **conventions** qui permettent la communication entre différents dispositifs au sein d'un réseau. Ils définissent la manière dont les données sont envoyées, reçues, et interprétées sur le réseau. Les protocoles réseau se classifient en différentes couches selon le modèle OSI (Open Systems Interconnection) ou le modèle TCP/IP, qui sont des frameworks utilisés pour comprendre et concevoir des systèmes de communication en réseau.

Les protocoles de transport sont responsables de la livraison fiable et ordonnée des données entre les applications qui communiquent sur un réseau. Ils assurent la gestion des flux de données, le contrôle des erreurs, et la gestion de la congestion. Les deux principaux protocoles de transport sont

### 1.1.2 Rappels sur les Protocoles de Transfert de Données

#### 1.1.2.1 Architectures des Protocoles de Transfert de Fichiers

Les architectures de protocoles de transfert de fichiers sont variées, mais elles partagent un objectif commun : optimiser le transfert de données entre un ou plusieurs nœuds dans un réseau. Ces architectures peuvent être centralisées ou décentralisées. Les systèmes centralisés reposent sur un serveur unique qui gère les requêtes de transfert, tandis que les systèmes décentralisés, comme BitTorrent, permettent une distribution des tâches de transfert entre plusieurs pairs, améliorant ainsi l'efficacité et la résilience. Les quatre architectures les plus courantes sont les suivantes :

#### Architecture Client-Serveur

L'architecture client-serveur est l'une des plus classiques et largement utilisées pour les transferts de données. Dans ce modèle, un serveur centralisé héberge les données et gère les requêtes des clients, qui se connectent pour télécharger ou envoyer des fichiers. Dans cette architecture, on retrouve des protocoles tels que FTP, HTTP/HTTPS et GridFTP. Cette architecture présente des avantages parmi lesquels :

- **La centralisation** qui permet un contrôle strict de l'accès aux données et une gestion simplifiée.
- **La facilité** de mise en œuvre et de gestion pour les administrateurs.

L'architecture client-serveur, bien que simple et intuitive, présente néanmoins des limitations sévères, à savoir :

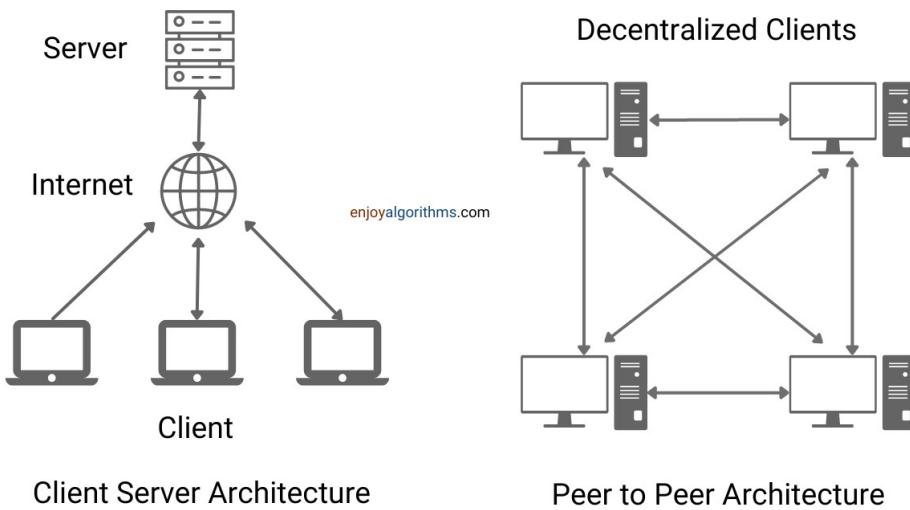




- **les goulets d'étranglement** possibles au niveau du serveur.
- **La vulnérabilité** accrue en cas de défaillance du serveur central.

### Architecture Peer-to-Peer (P2P)

FIGURE 1.1 – Architecture client-serveur vs architecture P2P



L'architecture Peer-to-Peer (P2P) est une architecture décentralisée, permettant à chaque nœud du réseau d'agir à la fois comme client et serveur. Les données sont partagées directement entre les pairs, sans besoin de serveur central. Par rapport à l'architecture client-serveur, cette architecture permet de résoudre la vulnérabilité due à la présence d'un seul serveur pouvant paralyser le transfert en cas de panne. Deux principaux protocoles de transfert de données utilisent cette architecture **BitTorrent**[3], Un protocole populaire qui divise les fichiers en morceaux, lesquels sont distribués entre les utilisateurs qui partagent et téléchargent simultanément et [?] Utilisé dans les premières applications de partage de fichiers, où les utilisateurs se connectaient directement pour partager des fichiers. Parmi ses avantages, on distingue entre autres :

- **Résilience** accrue grâce à l'absence d'un point unique de défaillance.
- **Haute scalabilité**, chaque nouveau pair augmentant la capacité totale du réseau.

malgré ces avantages, cette architecture présente des limitations notoires :

- **Complexité** accrue dans la gestion et le suivi des ressources.
- **Problèmes potentiels de sécurité et de fiabilité** des données.





### Architecture Hybride (Client-Serveur + P2P)

L'architecture hybride combine les avantages des architectures client-serveur et P2P. Un serveur central peut gérer la coordination ou l'indexation, tandis que le transfert réel des données se fait directement entre les pairs. On distingue deux principaux protocoles dans cette architecture à savoir **eMule** qui utilise un serveur central pour l'indexation des fichiers, mais les fichiers eux-mêmes sont transférés entre les utilisateurs via P2P et **Skype** qui dans ses versions antérieures utilisait une architecture hybride où les serveurs centraux géraient les connexions, mais les appels audio/vidéo étaient transférés directement entre les utilisateurs. En termes d'avantages, cette approche assure

- **Une meilleure gestion des ressources** grâce à la coordination centralisée.
  - **Une scalabilité améliorée** tout en maintenant un certain niveau de contrôle centralisé.
- parmi ses inconvénients, on retrouve :
- **Complexité** plus élevée en raison de la gestion des deux types de connexions.
  - **Vulnérabilité** partielle au niveau du serveur central en cas de défaillance.

### Architecture Multi-Source/Segmented Download

Cette architecture permet de télécharger simultanément des segments d'un fichier à partir de multiples sources, ce qui peut considérablement augmenter la vitesse de téléchargement. Elle est souvent utilisée dans des environnements distribués ou avec des systèmes de mise en cache. Comme protocole implémentant cette architecture, on retrouve **HTTP avec CDN** bien que HTTP soit traditionnellement client-serveur, les CDN utilisent une architecture multi-source pour servir des segments de contenu à partir de serveurs répartis géographiquement et **MDTMFTP** qui permet de transférer des fichiers en parallèle à partir de plusieurs sources, optimisant ainsi la bande passante disponible. cette architecture présente des avantages significatifs à savoir :

- Amélioration significative des performances de téléchargement.
- Résilience accrue en cas de défaillance d'une source.

mais aussi des difficultés considérables :

- **Complexité** dans la reconstitution des fichiers à partir de segments provenant de sources multiples.
- **Potentielle incohérence** des données si les sources ne sont pas bien synchronisées.

#### 1.1.2.2 Les Protocoles de Transport comme Substrat

Les protocoles de transfert de données permettent de coordonner la progression du transfert des données, mais pour assurer la livraison des données d'un point à un autre, ils se basent sur des protocoles de transport.





Les protocoles de transport forment la couche critique sur laquelle reposent les transferts de données sur Internet. Ils déterminent la manière dont les données sont segmentées, envoyées, et réassemblées à destination, tout en gérant la fiabilité, la congestion, et la latence. Les deux principaux protocoles de transport, TCP et UDP, offrent différentes approches avec des avantages et des inconvénients spécifiques pour le transfert de données.

### 1.1.2.3 TCP

TCP est un protocole de transport orienté connexion, ce qui signifie qu'il établit une connexion fiable entre l'émetteur et le récepteur avant de commencer le transfert de données. Ce protocole assure l'ordonnancement et l'intégrité des paquets de données en utilisant des mécanismes de contrôle de flux, de détection et de correction d'erreurs. Parmi les principaux avantages qu'offre TCP on retrouve :

- **Fiabilité** : TCP garantit que toutes les données sont livrées, dans l'ordre, et sans duplication. Si des paquets sont perdus, TCP les retransmet automatiquement.
- **Contrôle de flux** : TCP ajuste dynamiquement le débit d'envoi pour éviter la congestion réseau, ce qui aide à prévenir les surcharges et à optimiser l'utilisation de la bande passante.
- **Large adoption** : En raison de sa fiabilité, TCP est le protocole de transport le plus utilisé pour des applications nécessitant une transmission sécurisée des données, comme HTTP/HTTPS, FTP, et l'email.

Ce protocole présente des inconvénients qui peuvent fortement ralentir un protocole de transfert des données, notamment :

- **Latence élevée** : L'établissement de la connexion (handshake à trois voies) et les mécanismes de retransmission peuvent introduire une latence importante, surtout dans les réseaux avec des délais de propagation élevés.
- **Surcharge de bande passante** : Les mécanismes de contrôle de flux et de correction d'erreurs ajoutent une surcharge de bande passante, ce qui peut être inefficace pour les petites transmissions ou les réseaux à faible bande passante.
- **Scalabilité limitée** : TCP est moins adapté aux environnements où une scalabilité massive est requise, comme dans le cas des architectures décentralisées avec de nombreux nœuds.

### 1.1.2.4 UDP

UDP est un protocole de transport sans connexion qui envoie les données sous forme de datagrammes, sans vérifier si elles sont correctement arrivées à destination. Il n'y a pas d'établissement de connexion préalable, ce qui le rend beaucoup plus rapide que TCP pour l'envoi de



petits volumes de données ou dans des situations où la latence est critique. cette simplicité du protocole UP lui accorde un certain nombre d'avantages par rapport à TCP tels que

- **Faible latence** : Sans les procédures d'établissement de connexion et de retransmission, UDP minimise la latence, ce qui le rend idéal pour les applications nécessitant une transmission rapide, comme le streaming vidéo, les jeux en ligne, et la VoIP.
- **Simplicité** : L'absence de mécanismes complexes (comme le contrôle de flux) réduit la surcharge, ce qui le rend plus efficace pour les transmissions où la fiabilité n'est pas critique.
- **Flexibilité** : UDP permet aux développeurs de gérer manuellement la fiabilité et la correction d'erreurs, ce qui est avantageux dans les systèmes où le contrôle fin du transfert de données est nécessaire.

c'est également cette simplicité du protocole UDP qui fait ses principales faiblesses

- **Aucune garantie de livraison** : UDP n'offre aucune garantie que les paquets arrivent à destination ni qu'ils soient dans le bon ordre, ce qui peut entraîner une perte de données ou une désynchronisation.
- **Absence de contrôle de congestion** : Sans mécanismes intégrés de gestion de la congestion, UDP peut aggraver les problèmes de surcharge réseau, surtout dans les environnements très sollicités.
- **Risques de sécurité** : La simplicité de UDP le rend vulnérable à des attaques comme le spoofing et les inondations, ce qui nécessite des mesures de sécurité supplémentaires lors de son utilisation.

#### 1.1.2.5 Notion de Multi-chemin et Protocoles de transport Associés

Le concept de multi-chemin (ou multipath) consiste à permettre à une connexion réseau d'utiliser plusieurs chemins simultanément pour le transfert de données. Cela permet d'améliorer plusieurs aspects du transfert, notamment la résilience, la performance et l'équilibrage de charge. Les protocoles basés sur le multi-chemin sont particulièrement adaptés aux environnements où la connectivité peut être variable, comme les réseaux mobiles, les environnements de cloud, ou les architectures décentralisées. Parmi ces protocoles on retrouve principalement

#### MPTCP

MPTCP est une extension du protocole TCP qui permet à une seule connexion TCP d'utiliser plusieurs chemins simultanément. Il a été développé pour surmonter les limitations de TCP dans des environnements où la connectivité et la bande passante peuvent varier, comme dans les réseaux mobiles ou les architectures multi-homed (dispositifs ayant plusieurs interfaces réseau). Il fonctionne en subdivisant une connexion TCP en plusieurs sous-flux, chacun étant capable d'utiliser un chemin réseau différent. Cela permet d'optimiser l'utilisation des ressources réseau disponibles, en utilisant la capacité totale de plusieurs liens.





## SCTP

SCTP est un protocole de transport orienté connexion, conçu à l'origine pour le transport de signaux de télécommunication sur IP, mais qui a trouvé d'autres applications dans des environnements nécessitant une transmission fiable de données en flux multiple. Il permet la transmission de plusieurs flux indépendants de données entre deux hôtes, chacun pouvant être transmis sur un chemin différent. SCTP intègre également des mécanismes de gestion de la congestion et de tolérance aux pannes, similaires à ceux de TCP. Ces deux protocoles présentent une avancé considérable en terme de protocoles de transport s'accompagnant ainsi de plusieurs avantages

- **Multiplexage de flux** : SCTP permet à plusieurs flux indépendants de données d'être transmis simultanément sur une même connexion, évitant ainsi les blocages tête de ligne qui peuvent survenir avec TCP.
- **Tolérance aux pannes** : SCTP est capable de détecter une défaillance de chemin et de rerouter les flux de données vers un autre chemin, similaire à MPTCP.
- **Sécurité accrue** : SCTP intègre des mécanismes de protection contre les attaques de type SYN-flooding, plus robustes que ceux de TCP.

et des inconvénients inhérents :

- **Adoption limitée** : SCTP est moins largement déployé que TCP et UDP, ce qui limite son utilisation dans les environnements réseau courants.
- **Complexité** : Comme MPTCP, SCTP est plus complexe à implémenter et à gérer que TCP ou UDP, ce qui peut compliquer son adoption et son déploiement.

Bien que performants pour le transport des données, ces protocoles souffrent des diverses difficultés d'adoption et de mise en œuvre. C'est par exemple le cas de TCP qui est fortement couplé au noyau du système d'exploitation hôte[19].

## 1.2 Présentation du protocole QUIC

### 1.2.1 introduction

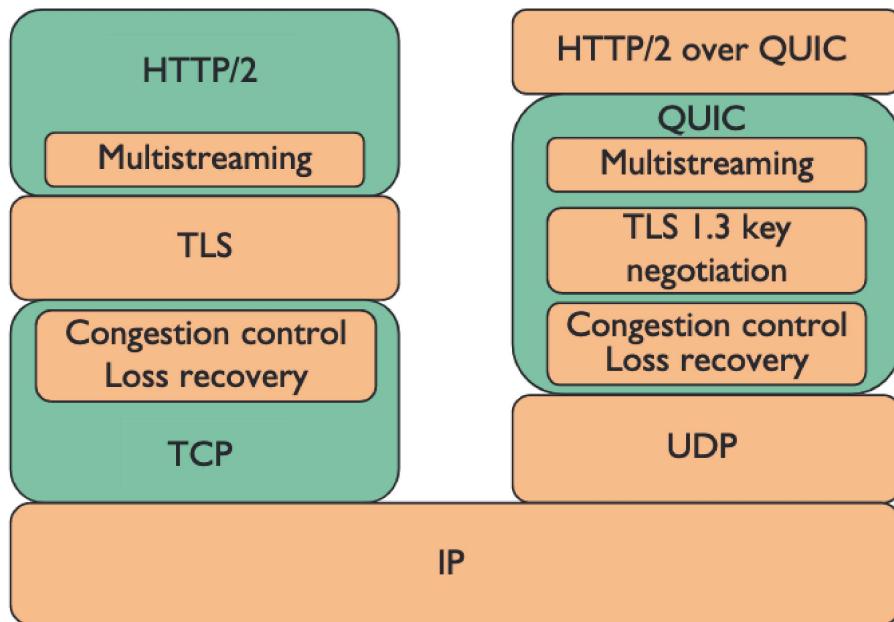
QUIC est un protocole de transport initié par Google en 2012, avec une première version officielle introduite par l'Internet Engineering Task Force (IETF) en 2021. Il a été développé pour résoudre plusieurs problèmes associés au protocole TCP. Premièrement, dans TCP, la perte d'un paquet empêche tous les paquets suivants d'atteindre l'application en raison de la livraison séquentielle et de la gestion des flux basée sur les fenêtres. Deuxièmement, TCP utilise l'adresse IP et le numéro de port pour identifier une connexion, ce qui nécessite une négociation coûteuse en cas de changement d'adresse IP ou de numéro de port, car il ne permet pas la réutilisation du contexte de communication. Troisièmement, TCP ne prend pas en charge nativement le multiplexage. Plutôt que d'améliorer TCP, il a été décidé de développer un nouveau protocole.



En effet, TCP est implémenté dans le noyau, alors que QUIC est implémenté sur UDP, ce qui le rend plus flexible. De plus, QUIC intègre des fonctionnalités telles que le multiplexage, la cryptographie, le contrôle de congestion et la récupération des pertes, contrairement à TCP où ces fonctionnalités sont fixes [4, 19].

### 1.2.2 Principe de fonctionnement

FIGURE 1.2 – QUIC architecture.[4]



QUIC fonctionne en remplaçant la majorité de la pile HTTPS traditionnelle, combinant les fonctionnalités de HTTP/2, TLS et TCP. Il utilise UDP comme substrat, permettant à ses paquets de traverser les middleboxes (dispositifs intermédiaires) tout en maintenant une sécurité et une intégrité des données élevées grâce à un chiffrement et une authentification intégrée. Cette utilisation d'UDP permet également une plus grande flexibilité et rapidité dans la mise à jour et le déploiement du protocole. Pour mieux comprendre le fonctionnement de QUIC, examinons en détail ses principales composantes et la manière dont elles améliorent les performances et la sécurité des communications réseau.

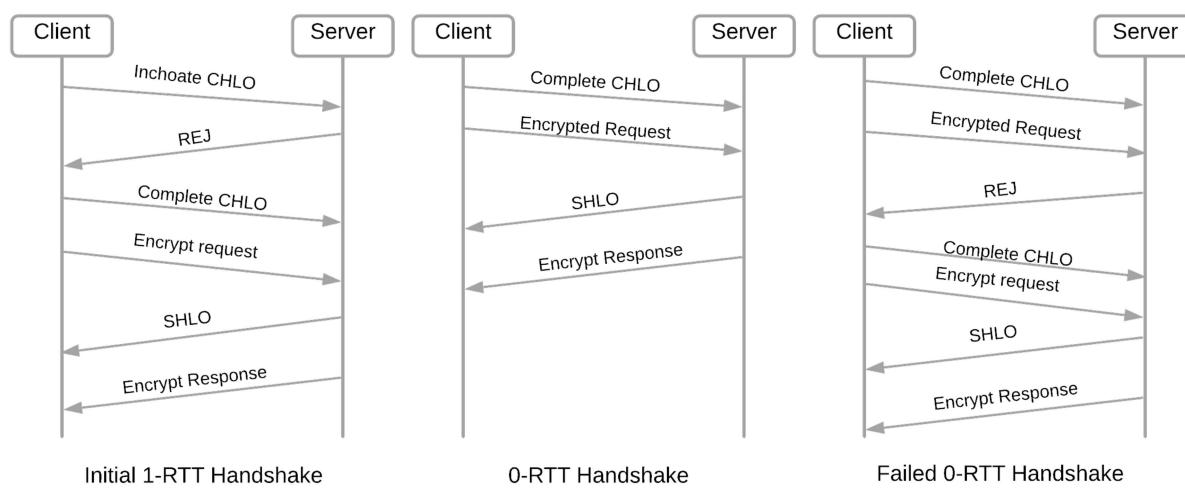


### 1.2.3 Les principaux concepts autour du protocole QUIC

#### 1.2.3.1 Établissement de connexion

QUIC s'appuie sur une poignée de main (handshake) cryptographique et de transport combinée pour établir une connexion de transport sécurisée. Il utilise une approche de connexion 0-RTT et 1-RTT, permettant d'établir une connexion sécurisée avec une latence minimale [4, 19].

FIGURE 1.3 – QUIC Connection establishment 0-RTT and 1-RTT handshake.[4]



- **0-RTT :** Permet aux clients qui ont précédemment connecté de reprendre une session et d'envoyer des données dès le premier paquet. Cela réduit considérablement le temps nécessaire pour établir une connexion.
- **1-RTT :** Utilisé pour les nouvelles connexions, où une poignée de main initiale est effectuée pour échanger les clés de chiffrement et établir une connexion sécurisée en un seul aller-retour.

La sécurité de l'établissement de connexion est assurée par l'utilisation de TLS, qui garantit que toutes les communications sont chiffrées dès le début de la connexion. QUIC inclut également une négociation de version pour éviter des retards et garantir que le client et le serveur utilisent la même version du protocole. Cela améliore l'efficacité et la sécurité de l'établissement de la connexion.

#### 1.2.3.2 Authentification et Chiffrement

QUIC garantit que presque tous les paquets sont authentifiés et principalement chiffrés, sauf quelques paquets initiaux de la poignée de main et les paquets de réinitialisation[19]. Les parties



non chiffrées de l'en-tête, telles que les indicateurs de paquets, le Connection ID, et le numéro de paquet, sont essentielles pour le routage et le déchiffrement. Toute tentative de manipulation des paquets de poignée de main non chiffrés entraînera l'échec de la connexion, assurant ainsi une sécurité robuste dès l'initialisation.

### 1.2.3.3 Multiplexage de Flux

Le multiplexage de flux permet à plusieurs flux de données indépendants d'être envoyés simultanément sur une seule connexion QUIC. Chaque flux est identifié de manière unique, ce qui permet une transmission parallèle des données sans interférence entre les différents flux. Cette fonctionnalité élimine le blocage tête de ligne (head-of-line blocking). Contrairement à TCP, où la perte d'un paquet peut bloquer la livraison de tous les paquets suivants, dans QUIC, cette perte de paquets dans un flux n'affecte pas les autres flux, ce qui améliore grandement l'efficacité et la rapidité des transmissions [19].

### 1.2.3.4 Récupération de Pertes

La récupération de pertes dans QUIC est basée sur des algorithmes modernes de détection et de correction des pertes de paquets, conçus pour minimiser l'impact sur la performance de la connexion. QUIC numérote tous les paquets y compris ceux des retransmissions et chaque paquet porte sans distinction un nouveau numéro. Ceci permet d'éviter la nécessité d'un mécanisme distinct les acquittements des retransmissions de ceux des transmissions originales comme c'est le cas dans TCP, il utilise également des décalages de flux dans les trames de flux pour assurer l'ordre de livraison permettant ainsi de séparer ces deux fonctions confondues dans TCP [4, 19].

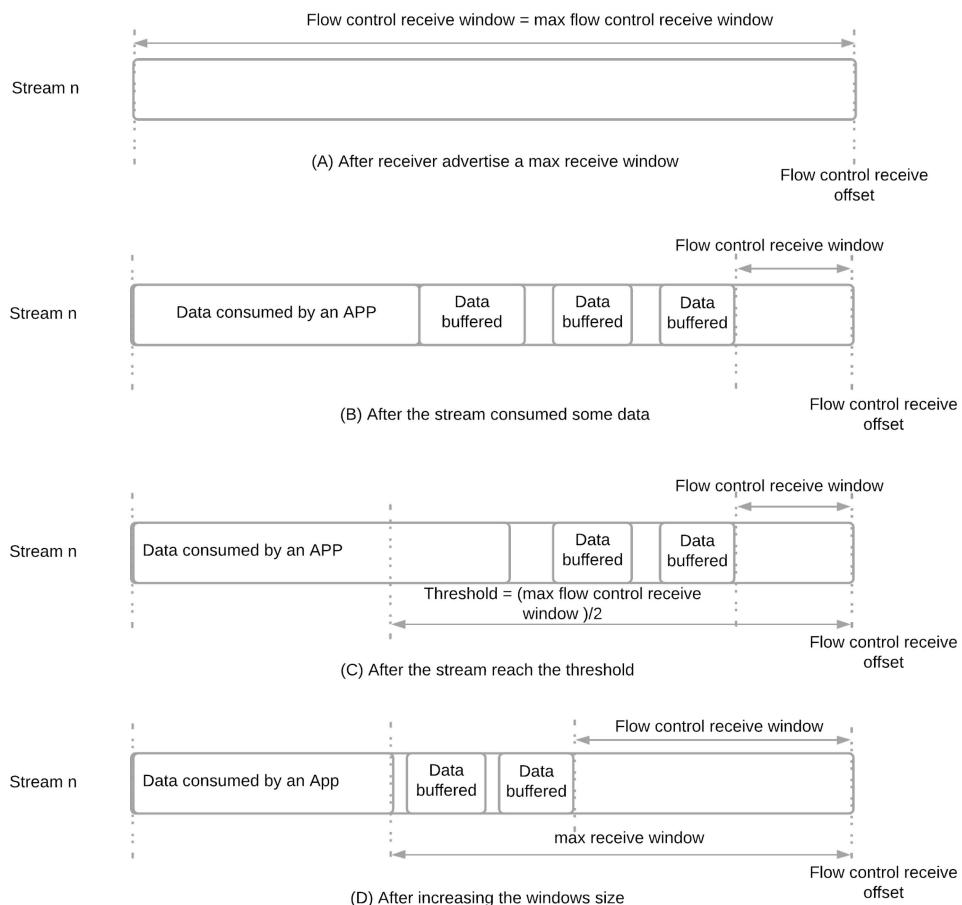
### 1.2.3.5 Contrôle de flux

QUIC utilise un contrôle de flux à deux niveaux pour éviter les blocages entre les flux : un au niveau de la connexion globale, et un autre spécifique à chaque flux. Le récepteur limite la quantité de données qu'il est prêt à recevoir en indiquant le décalage maximum d'octets pour chaque flux. Ce mécanisme permet de réguler efficacement le tampon utilisé, évitant ainsi la surcharge du récepteur et permettant une transmission fluide des données[19]. la figure 1.8 illustre ce processus de contrôle de flux.





FIGURE 1.4 – QUIC stream flow control.[4]



### 1.2.3.6 Contrôle de Congestion

Le contrôle de congestion dans QUIC est conçu pour maximiser le débit tout en s'adaptant aux conditions changeantes du réseau. Il intègre des algorithmes modernes de récupération des pertes, comme le RTO et le F-RTO, et fournit des retours détaillés grâce à l'utilisation de numéros de paquets monotones. QUIC utilise souvent l'algorithme NewReno ou Cubic pour la gestion de la congestion, avec une capacité à maintenir la connexion lors des changements d'adresse IP, grâce à l'ID de connexion, permettant ainsi une migration sans interruption.

### 1.2.4 Extension de QUIC pour le multipath : MP-QUIC

MP-QUIC (Multipath QUIC) est une extension de QUIC qui permet d'exploiter plusieurs interfaces réseau simultanément. MP-QUIC combine les avantages de QUIC avec les capacités de multipath. Cette extension introduit de nouvelles notions non présentes dans l'implementation



du QUIC traditionnel qui nous seront utiles pour concevoir notre solution.

MPQUIC présente des avantages conceptuels significatifs. En étant basé sur QUIC, il est plus facile à déployer, car ne dépend pas du système d'exploitation et intègre les informations de contrôle des multiples trajets avec un minimum d'interférence, tout en tirant parti des flux d'applications multiples et des priorités HTTP/2 pour optimiser la transmission des paquets dans des environnements variés [24]. Il existe cependant d'autres protocoles de transport intégrant les fonctionnalités de multipath comme MTCP, SCTP, mais qui nécessitent souvent une prise en charge par le système d'exploitation, ce qui ralentit leur adoption, notamment dans les appareils mobiles.

#### 1.2.4.1 Avantages de MP-QUIC

Bénéficiant déjà des avantages du QUIC traditionnels, le protocole MP-QUIC bénéficie en plus des avantages suivants :

- **Utilisation de multiples chemins réseau** : permet l'utilisation simultanée de plusieurs chemins réseau (par exemple, WiFi et LTE sur un smartphone), augmentant la bande passante globale disponible et améliorant la résilience en cas de défaillance d'une connexion.
- **Amélioration des performances** : Les évaluations montrent qu'il améliore le débit et réduit les temps de téléchargement par rapport à QUIC, TCP et MPTCP [24].
- **Flexibilité et Adaptabilité** : Il peut s'adapter dynamiquement aux conditions changeantes du réseau, basculant entre différentes connexions selon leur disponibilité et leur qualité. Ce qui assure une performance constante même dans des environnements réseau hétérogènes
- **Compatibilité avec les environnements variés** : MP-QUIC est conçu pour fonctionner efficacement dans divers environnements réseau, y compris les réseaux mobiles, les centres de données, et les réseaux domestiques et d'entreprises.

#### 1.2.5 Etude comparative des protocoles de transport QUIC, TCP et UDP

Afin de mettre en perspective les avantages et les inconvénients de QUIC par rapport à TCP et UDP, il est essentiel de comparer ces protocoles sur la base de critères pertinents qui justifient notre choix d'utiliser ce protocole dans notre travail plutôt que les autres. Dans les sections précédentes, nous avons axé notre rédaction de manière à ressortir pour chacun de ces protocoles les aspects utiles à cette section. Pour notre travail qui pour sa globalité portera sur un transfert sécurisé de données ordonnées et nécessitant à la fois une garantie de réception et une grande vitesse d'exécution, nous avons choisi les critères apparents de, la **latence**, la **sécurité**, **fiabilité**, **multiplexage de flux**, **contrôle de flux et de congestion**, la **récupération de**





TABLE 1.1 – Tableau comparatif des protocoles QUIC, TCP et UDP

Critere	QUIC	TCP	UDP
Type de Protocole	Orienté connexion sur UDP	Orienté connexion	Sans connexion
Fiabilité	Élevée (via les mécanismes de QUIC)	Élevée	Faible
Établissement de Connexion	Handshake 0-RTT rapide	Handshake en trois temps	Aucun
Multiplexage de Flux	Oui	Non	Non
Contrôle de Flux	Avancé	Standard	Aucun
Contrôle de Congestion	Oui	Oui	Non
Récupération de Perte	Numéros de paquets uniques, ACK explicites	Numéros de séquence, ACK	Aucun
Sécurité	Chiffrement et authentification intégrés	TCP+TLS externe	Aucun
Flexibilité de Déploiement	Haute, espace utilisateur	Faible, noyau système	Haute, espace utilisateur
Résilience	Migration de connexion possible	Non	Non

pertes, et la résilience. le tableau 1 . 4 présente cette comparaison entre les protocoles QUIC, TCP et UDP.

### 1.3 Dimension Énergétique dans les Protocoles de Transport Décentralisés

L'importance de l'efficacité énergétique dans les réseaux de télécommunication est de plus en plus reconnue, en raison des préoccupations croissantes concernant l'empreinte carbone des infrastructures numériques. Cependant, malgré cette prise de conscience, les protocoles réseau traditionnels n'intègrent généralement pas la dimension énergétique dans leur conception.

La consommation énergétique des protocoles actuels est principalement influencée par des facteurs tels que la surcharge de la bande passante, la gestion des connexions, la retransmission des paquets et le contrôle de la congestion au niveau de la couche transport. Ces opérations, bien que nécessaires pour assurer la fiabilité et la performance des transferts de données, peuvent conduire à une utilisation inefficace de l'énergie, en particulier dans les environnements distribués et décentralisés.

Quelques études ont tenté d'estimer l'impact énergétique des protocoles réseau, bien que les chiffres varient selon les méthodologies et les contextes étudiés. D'après Balakrishnan et al.,



2020, Les mécanismes de retransmission et de contrôle de la congestion de TCP, bien qu'essentiels pour la fiabilité, consomment des quantités significatives d'énergie. Une étude a estimé que dans certains environnements, TCP pourrait entraîner jusqu'à 10-15% de surconsommation d'énergie par rapport à des protocoles optimisés pour des tâches spécifiques.

### 1.3.1 Vers des Protocoles Écoénergétiques

Un des moyens d'améliorer l'efficacité énergétique d'un protocole de transfert de données décentralisé est d'introduire l'usage de sources d'énergies vertes et de prioriser les nœuds fonctionnant aux énergies renouvelables (par exemple, alimentés par le solaire ou l'éolien) pour participer au transfert de données. Cette approche repose sur l'idée que, dans un réseau distribué, tous les nœuds ne sont pas égaux en termes d'empreinte énergétique. Pour améliorer l'efficacité énergétique des transferts dans un mix énergétique (renouvelable + non renouvelable) des mécanismes peuvent être implémenté :

- **Sélection dynamique des nœuds** : Les protocoles pourraient être conçus pour sélectionner les nœuds participant au transfert de données en fonction de leur source d'énergie. Par exemple, un nœud alimenté par l'énergie solaire pourrait être préféré à un autre alimenté par une source d'énergie plus polluante, surtout pendant les périodes de production maximale d'énergie renouvelable.
- **Routing vert** : En fonction de la disponibilité des nœuds écoénergétiques, le protocole pourrait ajuster les routes de transfert de données pour minimiser l'impact énergétique global, même si cela signifie contourner des routes plus directes, mais plus énergivores.

## 1.4 État de l'art

L'état de l'art dans le domaine des protocoles de transfert de données décentralisés et distribués se concentre sur les solutions qui visent à améliorer la performance, la résilience et l'efficacité énergétique des réseaux modernes. Cette section examine les protocoles actuels, leurs forces et leurs limitations.

### 1.4.1 Étude des Protocoles Existantes

Dans cette section, nous analysons les protocoles de transfert de données en fonction de leur architecture et de leur capacité à répondre aux besoins des environnements décentralisés et distribués.

### 1.4.2 HTTP

Le HTTP est un protocole de communication destiné aux échanges d'informations sur le Web. Il repose sur un modèle client-serveur où un client envoie des requêtes à un serveur, qui répond



avec les données demandées. Il a été conçu pour faciliter la communication sur le Web. HTTP permet la récupération de documents hypertextes (pages web) et de fichiers multimédias. Son principal objectif est de fournir un mécanisme flexible pour interagir avec les ressources en ligne.

**Principe de Fonctionnement** Le HTTP fonctionne sur une connexion TCP et utilise une architecture de requête-réponse. Le client, généralement un navigateur web envoie des requêtes HTTP (par exemple, GET, POST) et le serveur répond avec des ressources, comme des pages HTML ou des fichiers JSON. Chaque transaction HTTP est stateless (sans état), c'est-à-dire que chaque requête est indépendante de la précédente.

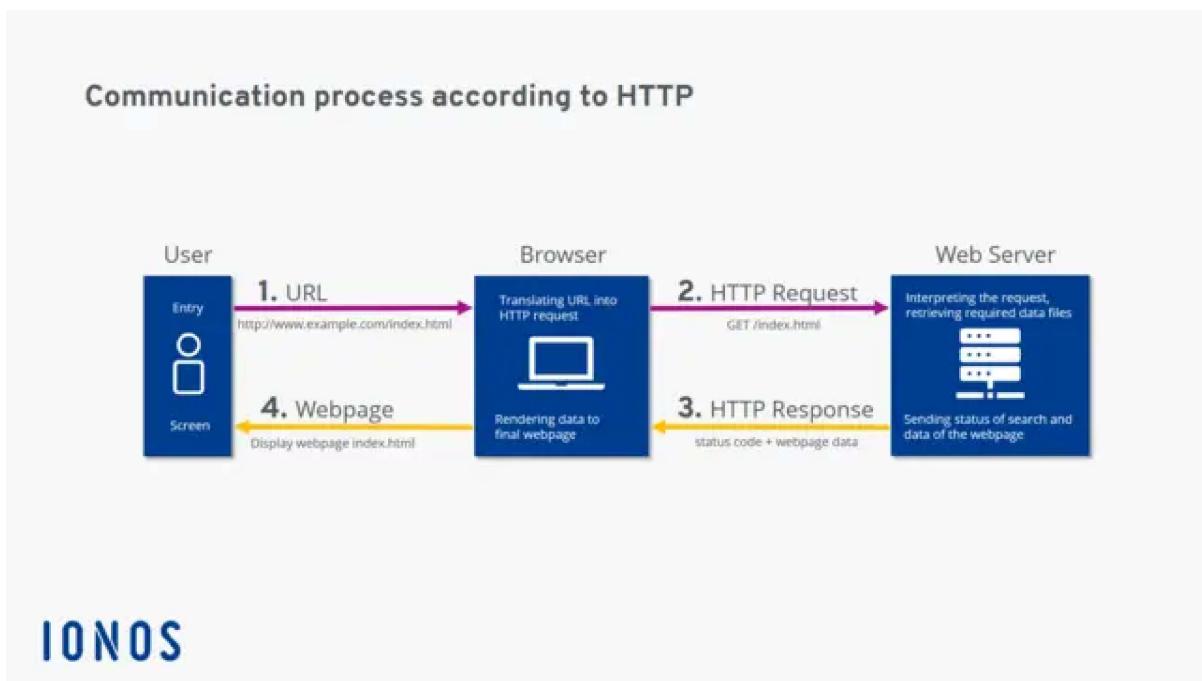


FIGURE 1.5 – Processus de communication HTTP.[18]

### 1.4.3 FTP

FTP[14] est un protocole standard utilisé pour transférer des fichiers entre un client et un serveur sur un réseau TCP/IP. Il permet à un utilisateur d'uploader ou de télécharger des fichiers. FTP a été conçu pour permettre un transfert fiable de fichiers volumineux entre des systèmes distants, tout en offrant des fonctionnalités basiques d'authentification.

**Principe de Fonctionnement** FTP utilise deux canaux de communication[23] : un canal de commande (port 21) pour échanger des commandes et un canal de données (port 20 ou dynamique) pour le transfert des fichiers. Il fonctionne en mode actif (où le client accepte une connexion depuis le serveur pour les données) ou passif (où le client initie la connexion de données).



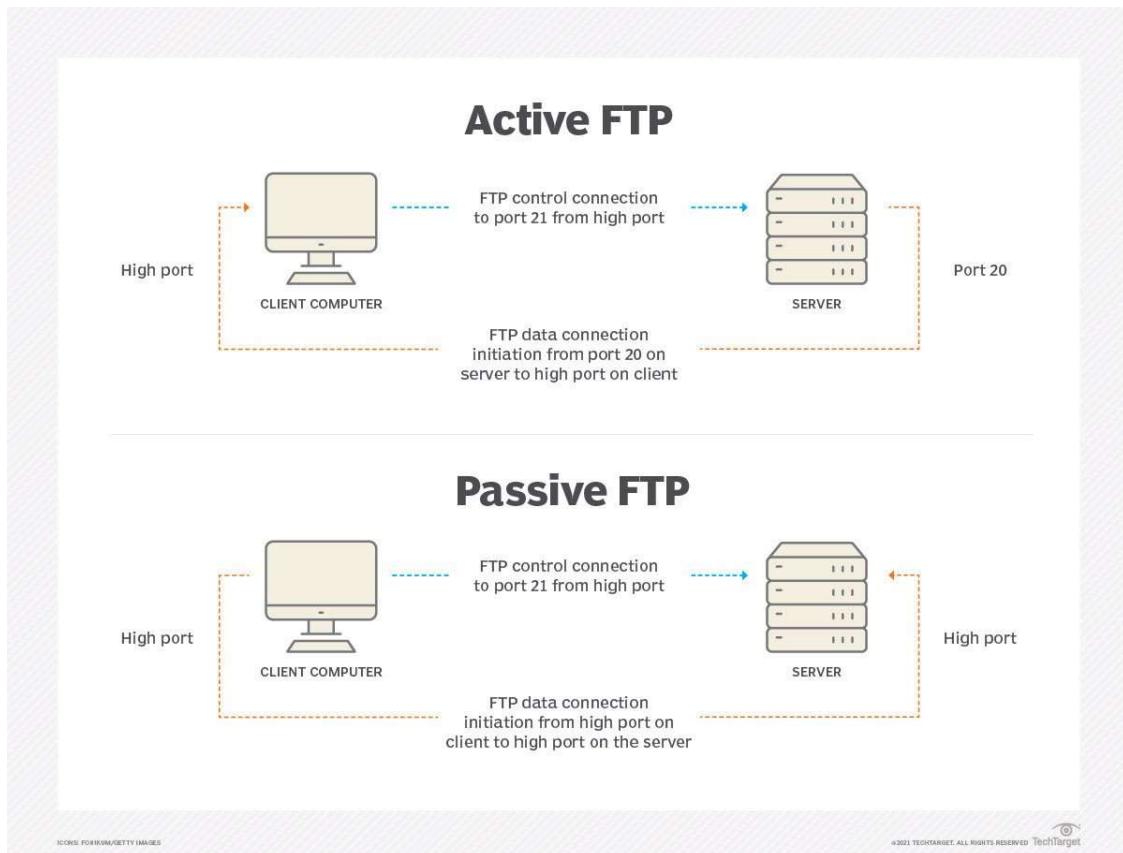


FIGURE 1.6 – Fonctionnement modes actif et passif du protocole FTP.[23]

#### 1.4.4 SFTP

SFTP est une version sécurisée de FTP fonctionnant au-dessus du protocole SSH (Secure Shell). Il assure le transfert de fichiers avec des mécanismes de sécurité avancés. SFTP a été conçu pour combiner les fonctionnalités de FTP avec la sécurité de SSH, garantissant ainsi la confidentialité et l'intégrité des fichiers échangés.

**Principe de Fonctionnement** SFTP utilise une seule connexion sécurisée sur le port 22, contrairement à FTP qui en utilise deux. Toutes les données, y compris les commandes et les fichiers, sont chiffrées via SSH. Les utilisateurs doivent s'authentifier via SSH pour accéder aux fichiers distants, et toutes les actions de gestion de fichiers sont effectuées de manière sécurisée. L'architecture du protocole repose sur les éléments clés suivants :

- **Client SFTP** : Le logiciel utilisé pour établir une connexion SSH sécurisée et gérer les fichiers.
- **Serveur SFTP** : L'hôte sécurisé qui accepte les connexions SFTP et assure le transfert sécurisé des fichiers.





- **SSH** : Le protocole sous-jacent qui fournit le chiffrement et l'authentification des connexions.

#### 1.4.5 GridFTP

GridFTP[22] est une extension du protocole FTP, conçue pour le transfert de fichiers à grande échelle dans des environnements distribués comme les grilles de calcul et les centres de données. Il améliore FTP en y ajoutant des fonctionnalités adaptées aux réseaux à large bande passante et à haute latence[5, 10], permettant ainsi une meilleure gestion des transferts massifs de données.

**Principe de fonctionnement** GridFTP utilise plusieurs connexions parallèles pour optimiser la bande passante disponible lors du transfert de données volumineuses. Il inclut des mécanismes tels que la reprise automatique des transferts après interruption et la gestion des connexions multiples. Il est largement utilisé dans les grilles de calcul scientifique, où le transfert rapide et fiable de données massives est critique[5].

#### Caractéristiques principales

- **Parallélisme** : Utilisation de multiples connexions pour augmenter le débit de transfert.
- **Reprise automatique** : Les transferts interrompus peuvent reprendre là où ils ont été arrêtés.
- **Optimisation pour haute latence** : Conçu pour fonctionner efficacement sur des réseaux à haute latence.
- **Sécurité** : Supporte les extensions de sécurité comme GSI (Grid Security Infrastructure) pour l'authentification et le chiffrement.

**Cas d'usage typique** GridFTP est principalement utilisé dans les environnements de calcul intensif, les centres de données, et les réseaux distribués pour le transfert de fichiers volumineux dans des infrastructures de grilles, où la fiabilité et l'efficacité sont essentielles.

#### 1.4.6 BitTorrent

BitTorrent est un protocole peer-to-peer (P2P) conçu pour le partage de fichiers de manière distribuée[3, 11]. Contrairement aux approches centralisées, BitTorrent permet à chaque participant (appelé "pair") de télécharger et de partager simultanément des parties d'un fichier, ce qui rend le protocole extrêmement efficace pour la distribution de fichiers volumineux à un grand nombre d'utilisateurs.





**Principe de fonctionnement[11]** Le fichier à partager est divisé en petits morceaux. Chaque pair télécharge les morceaux disponibles auprès des autres pairs et, en parallèle, partage les morceaux qu'il a déjà téléchargés. Cela réduit la dépendance à un seul serveur central et améliore la robustesse et la vitesse des transferts à mesure que le nombre de pairs augmente.

### Caractéristiques principales

- **Décentralisation** : Aucun serveur centralisé n'est requis, la distribution des fichiers est répartie entre les pairs.
- **Partage simultané** : Les utilisateurs téléchargent et partagent des morceaux de fichier en même temps.
- **Évolutivité** : Plus il y a de pairs dans le réseau, plus le transfert est rapide.
- **Tolérance aux pannes** : La perte de quelques pairs n'affecte pas la capacité à télécharger le fichier.

**Cas d'usage typique** BitTorrent est couramment utilisé pour la distribution de fichiers volumineux à grande échelle, comme les logiciels open source, les distributions Linux, ou encore des médias volumineux. Il est particulièrement adapté aux environnements où le téléchargement par un grand nombre de personnes est nécessaire sans surcharger un serveur central.

### 1.4.7 FDT

FDT (Fast Data Transfer) est un protocole optimisé pour le transfert de fichiers à haut débit sur des réseaux longue distance. Il est principalement utilisé dans les contextes où les volumes de données sont massifs, comme les grandes expériences scientifiques [20]. FDT exploite pleinement la bande passante disponible sur les réseaux à longue distance, en utilisant des connexions multiples pour maximiser le débit.

**Principe de fonctionnement** FDT utilise plusieurs canaux TCP parallèles pour assurer un transfert rapide des fichiers volumineux. Il est capable de segmenter un fichier en flux multiples et de gérer la synchronisation de ces flux sur des réseaux à haute latence. Il optimise également l'utilisation de la bande passante, ce qui permet d'éviter les goulots d'étranglement sur les réseaux longue distance.[22]

### Caractéristiques principales

- **Parallélisme des flux** : Utilisation de plusieurs flux TCP simultanés pour maximiser l'utilisation de la bande passante.
- **Optimisation pour longue distance** : Conçu pour des transferts à très haute vitesse sur des réseaux à grande distance géographique.





- **Reprise automatique** : Capacité à reprendre les transferts interrompus.
- **Gestion automatique des connexions** : Ajustement dynamique du nombre de connexions pour optimiser le débit.

**Cas d'usage typique** FDT est souvent utilisé dans les grandes infrastructures scientifiques et académiques, par exemple pour le transfert de données entre centres de recherche géographiquement éloignés ou pour les expériences dans le domaine de la physique des particules, où des volumes énormes de données doivent être transférés rapidement.

#### 1.4.8 MDTMFTP

MDTMFTP[22, 25] est une extension de FTP qui introduit des optimisations pour les transferts de données à travers des réseaux distribués. Il améliore les performances de FTP en exploitant le parallélisme et la gestion de flux multiples, tout en conservant la compatibilité avec les fonctionnalités de base de FTP, comme la gestion de fichiers et les contrôles d'accès.

**Principe de fonctionnement** MDTMFTP conserve le mécanisme de base de FTP mais optimise les transferts en gérant plusieurs connexions simultanées pour le transfert des fichiers. Il utilise des méthodes de compression et des ajustements dynamiques pour réduire les temps de transfert et gérer plus efficacement la bande passante disponible.

#### Caractéristiques principales

- **Parallélisme des connexions** : Utilisation de connexions multiples pour améliorer la vitesse de transfert.
- **Compression des données** : Réduction de la taille des fichiers pour accélérer les transferts.
- **Reprise automatique** : Support de la reprise des transferts en cas d'interruption.
- **Compatibilité FTP** : Conserve toutes les fonctionnalités de base de FTP (gestion des fichiers, authentification).

**Cas d'usage typique** MDTMFTP est particulièrement adapté aux environnements où il est nécessaire de transférer des fichiers volumineux de manière fiable et rapide sur des réseaux distribués. Il est couramment utilisé dans les centres de données, les infrastructures cloud, et les réseaux de calcul intensif.

#### 1.4.9 Limites des Protocoles Existantes

L'analyse des protocoles existants met en lumière plusieurs limitations, particulièrement dans les environnements décentralisés :





- **Scalabilité** : Les protocoles basés sur une architecture client-serveur, comme FTP, HTTP, et même SFTP, rencontrent des limites de scalabilité, en raison de leur dépendance à des serveurs centraux. Bien que des extensions comme GridFTP et MDTMFTP introduisent des améliorations, elles restent contraintes par leur architecture centralisée.
- **Résilience** : Les protocoles P2P comme BitTorrent offrent une meilleure résilience grâce à leur nature décentralisée, mais cette architecture expose également à des risques accrus de sécurité, tels que les attaques malveillantes.
- **Efficacité énergétique** : La consommation énergétique n'est pas un critère pris en compte dans la conception de la plupart des protocoles actuels, même ceux optimisés pour des transferts rapides comme FDT. Dans un contexte où l'empreinte carbone des infrastructures numériques devient un enjeu majeur, cette lacune doit être adressée.

#### 1.4.10 Impact du Protocole QUIC

Le protocole de transport moderne QUIC, vise à surmonter certaines des limitations des protocoles existants, en offrant des améliorations significatives en termes de performance, sécurité et résilience. En particulier, les extensions comme MP-QUIC introduisent des capacités de multi-chemin, améliorant la performance dans des environnements distribués tout en offrant une meilleure gestion de la congestion et du flux de données. QUIC est ainsi un bon support pour bâtir un protocole de transfert de données, car il est plus performant pour le transport que TCP et UDP mais à l'heure actuelle, nous n'avons pas connaissance des travaux d'amélioration sur les protocoles existants pour prendre en compte QUIC à l'exception des travaux sur HTTP3[4, 19] qui sont en cours.

#### 1.4.11 Étude Comparative des protocoles de transfert de données existant



TABLE 1.2 – Comparaison des protocoles de transfert de données existant à QuicPlex

Caractéristiques	GridFTP	BitTorrent	FDT	MDTMFTP	HTTP/2	FTP
Type de protocole	FTP optimisé pour les grilles	P2P (Pair-à-Pair)	Optimisé pour transfert à haut débit	FTP amélioré pour réseaux larges	Protocole d'application (HTTP)	Protocole d'application
Multiplexage	Oui, via connexions multiples	Oui, via les pairs	Oui, plusieurs flux TCP	Oui, plusieurs flux FTP	Oui, via une seule connexion TCP	Non
Transfert P2P	Non	Oui	Non	Non	Non	Non
Bandé passante	Optimisée avec plusieurs connexions	Partagée entre pairs	Maximisée via flux multiples	Maximisée via flux multiples	Optimisée par la gestion des flux	Fixe, sans optimisation
Tolérance aux pannes	Oui, reprise automatique	Oui, via les pairs	Oui, reprise automatique	Oui, reprise automatique	Non	Non
Sélection des relais	Non applicable	Dynamique (pairs aléatoires)	Non applicable	Non applicable	Non applicable	Non applicable
Gestion de l'énergie verte	Non	Non	Non	Non	Non	Non
Priorisation des flux	Non	Non	Oui	Non	Oui	Non
Sécurité	Sécurité optionnelle	Variable (dépend des clients)	Sécurité optionnelle	Sécurité optionnelle	Chiffrement (TLS)	Sécurité de base (authentification)
Latence	Moyenne, selon les conditions	Variable (dépend du réseau)	Faible (optimisé pour réseaux à longue distance)	Faible (optimisé pour grandes distances)	Faible (optimisé pour le web)	Haute
Usage typique	Transfert massif dans les grilles	Partage de fichiers volumineux	Transferts massifs scientifiques	Transfert de fichiers distribués	Communication client-serveur (web)	Transfert de fichiers simple





## 1.5 Bilan du chapitre et Positionnement

### 1.5.1 Bilan

Ce chapitre a examiné les principaux protocoles de transport et de transfert de données, avec un focus sur QUIC. Nous avons montré comment QUIC surmonte les limitations de TCP et UDP grâce à ses mécanismes avancés de contrôle de flux, de congestion, et de récupération des pertes. En comparant QUIC avec d'autres protocoles, nous avons mis en évidence ses avantages en termes de performance et de sécurité. De plus, notre analyse des protocoles comme HTTP, FTP, SFTP, BitTorrent, FDT, et MDTMFTP a révélé un manque de prise en compte des considérations énergétiques et la non utilisation du protocole QUIC comme protocole de transport[5, 14, 25] soulignant un besoin crucial pour des recherches futures dans ce domaine.

### 1.5.2 Positionnement

Notre travail vise à combler les lacunes identifiées, notamment l'absence de mécanismes d'optimisation énergétique et l'intégration insuffisante de QUIC dans les environnements décentralisés. Nous proposons de développer un nouveau protocole qui exploitera les avantages de QUIC tout en intégrant des mécanismes de décentralisation et d'efficacité énergétique. Cette approche répondra aux besoins croissants en matière de transfert de données tout en minimisant la consommation d'énergie, un aspect critique rarement abordé par les protocoles actuels.

