**République Algérienne Démocratique et Populaire**

**Ministère de l'Enseignement Supérieur et de la**

**Recherche Scientifique**

**Université 20 Août 1955- Skikda**

**Faculté des Sciences**

**Département d'Informatique**

# Mémoire

En vue de l'obtention du diplôme de

Master en Informatique

Spécialité : Systèmes Informatiques

# Thème

---

## Amélioration du Protocole TCP pour les Applications Temps Réel Multimédia

---

**Présenté Par :**

Insaf SAADI

**Encadré Par :**

Dr. Mehdi BOULAICHE

**Année universitaire 2024/2025**

# DEDICACES

Je dédie ce mémoire à mes chers parents, qui ont veillé sur moi avec amour et m’ont accompagné tout au long de mon parcours scolaire.

# REMERCIEMENTS

Tout d’abord, toute ma gratitude va à Dieu clément  pour les bienfaits et les facilités qui m’ont portée tout au long de ce travail.

Mes sincères remerciements sont exprimés à DR BOULAICHE Mehdi pour son aide précieuse et ses conseils judicieux, malgré les contraintes de temps.

Je remercie également les membres du jury pour le temps qu’ils ont consacré à l’évaluation de ce travail, ainsi que pour la richesse de leurs remarques.

Je tiens à exprimer ma gratitude à mes chers parents qui m’ont aimée inconditionnellement et qui m’ont transmis tout ce qu’ils savaient et ont toujours tenté de me protéger de tout mal.

Ils m’ont conseillé avec le cœur, comme s’ils espéraient d’avoir  quelqu’un pour les conseiller et les protéger dans leur jeunesse.

Ils m’ont accompagnée tout au long de mon parcours scolaire, depuis la première année préparatoire jusqu’à ce moment de graduation. À eux, toute ma reconnaissance.

Je ne peux pas clore cette liste sans citer à ma chère sœur et ma meilleure amie Selma qui m’a aidé dans cette période avec ses précieux conseils et aussi son énergie positive.

# RESUME

Ce mémoire vise à adapter le protocole TCP aux exigences des applications temps réel, telles que la vidéo interactive et la voix sur Internet. Pour ce faire, nous avons adopté une approche de communication inter-couches, où la couche application envoie un signal à la couche transport l'informant que les données transmises sont en temps réel. À la réception de ce signal, TCP élargit la fenêtre de réception, permettant ainsi le passage d'un plus grand volume de données sans attendre d'accusé de réception (ACK), réduisant ainsi la latence.

Des expériences ont montré que cette modification améliore les performances de TCP dans ce contexte, bien qu'elle n'ait pas été initialement prévue pour les applications temps réel. Les graphiques ci-joints mettent en évidence la nette différence de comportement du protocole avant et après la modification, avec une analyse détaillée des différences de latence et de réponse.

Mots-clés : Applications multimédias temps réel, protocoles de transport, TCP.

# ملخص

تهدف هذه المذكرة إلى تكييف بروتوكول TCP مع متطلبات تطبيقات الزمن الحقيقي، كالفيديو التفاعلي والصوت عبر الإنترنت. لتحقيق ذلك، قمنا باعتماد مقاربة تعتمد على التواصل بين الطبقات (Cross-layer communication)، حيث تُرسل طبقة التطبيق إشارة إلى طبقة النقل لتُعلِمها بأن البيانات المُرسَلة هي بيانات زمن حقيقي. في حال استلام هذه الإشارة، يقوم بروتوكول TCP بتكبير نافذة الاستقبال، مما يسمح بتمرير كمية أكبر من البيانات دون انتظار الإقرار (ACK)، وبالتالي تقليل الكمون.

أظهرت التجارب أن هذا التعديل يُحسّن أداء TCP في هذا السياق، رغم أنه غير مخصص في الأصل لتطبيقات الزمن الحقيقي. كما تُبرز المنحنيات المرفقة الفرق الواضح في سلوك البروتوكول قبل وبعد التعديل، مع تحليل دقيق للفروقات من حيث الكمون والاستجابة.

الكلمات المفتاحية: تطبيقات الزمن الحقيقي ملتيميديا ،بروتوكولات النقل ،بروتوكول TCP.

# ABSTRACT

This thesis aims to adapt the TCP protocol to the requirements of real-time applications, such as interactive video and voice over the Internet. To achieve this, we adopted a cross-layer communication approach, where the application layer sends a signal to the transport layer informing it that the transmitted data is real-time. Upon receipt of this signal, TCP enlarges the receiving window, allowing a larger amount of data to pass through without waiting for an acknowledgment (ACK), thus reducing latency.

Experiments have shown that this modification improves TCP performance in this context, although it was not originally intended for real-time applications. The accompanying graphs highlight the clear difference in the protocol's behavior before and after the modification, with a detailed analysis of the differences in latency and response.

Keywords: Real-time multimedia applications, transport protocols, TCP.

# Table des matières

# Liste des figures

**Chapitre 01**



**Chapitre 02**



**Chapitre 03**



**Chapitre 04**

# Liste des tableaux

**Chapitre 03**



**Chapitre 04**

# Liste des abréviations

| | |
|---|---|
| **LAN** | Local Area Network |
| **WAN** | Wide Area Network |
| **MAN** | Metropolitan Area Network |
| **PAN** | Personal Area Network |
| **NIC** | Network Internet Card |
| **MAC** | Media Access Control |
| **FTP** | File Transfer Protocol |
| **SMTP** | Simple Mail Transfer Protocol |
| **HTTP** | Hypertext Transfer Protocol |
| **MTP** | Media Transfer Protocol |
| **PC** | Personal Computer |
| **OSI** | Open Systems Interconnection |
| **TCP** | Transmission Control Protocol |
| **UDP** | User Datagram Protocol |
| **IP** | Internet Protocol |
| **ISO** | International Organization for Standardization |
| **HTTPS** | Hypertext Transfer Protocol Secure |
| **POP3** | Post Office Protocol version 3 |
| **IPv4** | Internet Protocol version 4 |
| **IPv6** | Internet Protocol version 6 |
| **BGP** | Border Gateway Protocol |
| **OSPF** | Open Shortest Path First |
| **IS-IS** | Intermediate System to Intermediate System |
| **ARP** | Address Resolution Protocol |
| **RARP** | Reversed Address Resolution Protocol |
| **TLP** | Transport Layer Protocols |

| | |
|---|---|
| **IETF** | Internet Engineering Task Force |
| **SSH** | Secure Shell |
| **IMAP** | Internet Message Access Protocol |
| **POP** | Post Office Protocol |
| **SYN** | Synchronization |
| **ACK** | Acknowledgement |
| **FIN** | Finish |
| **ICMP** | Internet Control Message Protocol |
| **VoIP** | Voice over IP |
| **QoS** | Quality of Service |
| **4G** | Fourth-Generation Mobile Telecommunications Technology |
| **VPN** | Virtual Private Network |
| **RWIN** | Receive WINDOW |
| **NAT** | Network Address Translation |
| **GNU/Linux** | GNU's Not Unix |
| **Xfce** | XForms Common Environment |
| **APT** | Advanced Package Tool |
| **USB** | Universal Serial Bus |
| **BIOS** | Basic Input/Output System |
| **UEFI** | Unified Extensible Firmware Interface |

# INTRODUCTION GENERALE

Avec l'évolution rapide des technologies de l'information et des communications, les réseaux informatiques sont devenus le support principal d'un large éventail d'applications, allant de la navigation web classique jusqu'aux applications multimédias interactives telles que la visioconférence, le streaming en temps réel ou encore les jeux en ligne. Ces dernières imposent des contraintes strictes en termes de latence, de gigue et de débit, qui ne sont pas toujours bien prises en charge par les protocoles de transport traditionnels.

Le protocole TCP (Transmission Control Protocol), bien qu'il soit largement utilisé pour sa fiabilité et son contrôle d'erreurs, n'est pas conçu à l'origine pour répondre aux exigences temps réel. Ses mécanismes de contrôle de flux, d'acquittement et de congestion, conçus pour maximiser la fiabilité, peuvent introduire des délais non négligeables, nuisibles à la qualité de service (QoS) dans les applications sensibles au temps.

Dans ce contexte, notre projet vise à proposer une amélioration du protocole TCP en adoptant une approche Cross-Layer, qui permet une meilleure interaction entre la couche application et la couche transport. L'idée principale consiste à introduire une option personnalisée dans l'en-tête TCP, utilisée pour signaler que les données transmises sont de type temps réel. Cette signalisation permettrait à TCP d'adapter dynamiquement son comportement, notamment en augmentant la taille de la fenêtre de réception, réduisant les délais de traitement, et assouplissant certains mécanismes de contrôle, sans compromettre la stabilité du réseau.

L'objectif de ce travail est donc de concevoir, modéliser, puis tester cette extension du protocole TCP afin d'en évaluer les avantages potentiels pour les applications multimédias en temps réel.

La structure de cette mémoire s'articule comme suit le premier chapitre présent des généralités sur les réseaux et les types d'applications, en suite on passe au deuxième chapitre qui est consacré au modèle OSI et aux protocoles de transport, notamment TCP et UDP. Puis on a le troisième chapitre qui détaille notre proposition technique : conception de l'option TCP, principes de communication inter-couches, et impact attendu, et en fin on présente la mise en œuvre pratique, les tests réalisés et l'analyse des résultats dans le quatrième chapitre.

CHAPITRE 01

# ETAT DE L'ART SUR LES RESEAUX INFORMATIQUES

## 1.1. Introduction

Les réseaux informatiques ont un but bien défini est que de faciliter la communication et échanger les données entre les différents appareils et utilisateurs mais cette transmission d'informations ne peut pas être passer si ces réseau ne utilise pas les protocoles de communication qui sont des ensembles de règles qui définissent comment l'information est échangée, traitée et interprétée entre les systèmes de communication modernes, comme les réseaux téléphoniques, l'Internet, les réseaux mobiles et sans fil. Ces protocoles garantissent que tout fonctionne de manière cohérente, fiable et sécurisée.

Dans ce chapitre, nous explorerons les divers types et topologies de réseaux, en mettant en lumière leurs caractéristiques distinctes. Nous examinerons également les composants matériels essentiels qui les composent, ainsi que l'organisation qui permet de les structurer et d'assurer leur bon fonctionnement.

## 1.2. Définition de réseau

Les réseaux informatiques désignent la connexion électronique d'ordinateurs afin de partager des informations telles que des fichiers, des applications, des imprimantes et des logiciels, ils ont plusieurs avantages notamment en termes de sécurité, d'efficacité, de gestion simplifiée et de rentabilité, car elle facilite la collaboration entre des utilisateurs aux profils variés. Fondamentalement, un réseau est constitué de composants matériels tels qu'un ordinateur, des routeurs qui jouent un rôle important dans le transfert de données d'un point à un autre grâce à différentes technologies telles que les ondes radio et les câbles (Annmalai, 2014).

## 1.3. Types de réseaux informatiques

Il existe plusieurs catégories de réseaux informatiques, classées en fonction de leur taille et de leur portée. Cependant, dans cette partie, nous nous concentrerons uniquement sur les quatre types les plus courants.

### 1.3.1. Réseau local (LAN)

Un LAN relie des périphériques réseau sur une distance relativement courte. Un immeuble de bureaux, une école ou un domicile en réseau ne comporte généralement qu'un seul LAN, même si un bâtiment peut parfois en contenir plusieurs petits (par exemple un par pièce), et qu'un LAN peut parfois s'étendre sur plusieurs bâtiments voisins.

Les vitesses de transfert de données sur un réseau local peuvent atteindre 10 Mbit/s (par exemple, pour un réseau Ethernet) et 1 Gbit/s (par exemple, pour un réseau FDDI). Un réseau local peut atteindre jusqu'à 100, voire 1 000 utilisateurs.

### 1.3.2. Réseau étendu (WAN)

Les réseaux étendus permettent de connecter des serveurs et des ordinateurs répartis sur plusieurs continents afin de garantir une mise à jour constante des informations. Utilisés

dans le monde entier, ils sont interconnectés entre eux pour créer un réseau étendu géant. Ils utilisent la fibre optique comme moyen de communication. Le plus grand exemple de réseau étendu est Internet, qui connecte tous les utilisateurs aux informations et aux données disponibles sur Internet.

### 1.3.3. Réseaux métropolitains (RMM)

Le terme « RMM » désigne généralement un réseau qui s'étend sur une zone urbaine ou une ville. Les RMM sont plus vastes que les réseaux locaux traditionnels et utilisent principalement des supports haut débit, comme la fibre optique, pour leur dorsale, ils sont courants dans les organisations qui doivent relier plusieurs petites installations pour partager des informations.

Les réseaux MAN partagent bon nombre des mêmes menaces de sécurité que les réseaux locaux, mais à plus grande échelle. La situation critique d'un administrateur central autorisant l'accès à d'innombrables bureaux dispersés dans une ville est un défi majeur. Un processus complexe qui exige des mécanismes de contrôle d'accès stricts pour protéger les informations contre tout accès non autorisé.

### 1.3.4. Réseau personnel (PAN) ou réseau Bluetooth

Un terme plus récent utilisé pour décrire un type de réseau est le réseau personnel (PAN). Les réseaux PAN sont généralement sans fil, établis à la demande ou de manière ad hoc pour communiquer entre deux ou plusieurs appareils. Ils peuvent être utilisés entre des appareils appartenant à deux parties différentes, ou entre deux appareils appartenant à une même personne, comme un assistant numérique personnel et un ordinateur portable ou un téléphone portable. Ces réseaux sont généralement caractérisés par une courte portée, souvent limitée à 10 mètres ou moins.

Le réseau sans fil Bluetooth est un exemple de technologie PAN. Conçue comme une technologie de remplacement des câbles, Bluetooth permet aux utilisateurs de se passer des câbles série et USB utilisés par de nombreux périphériques actuels et de s'appuyer sur un PAN Bluetooth pour communiquer ( Anoop et Rai, 2014).

## 1.4. Topologies des réseaux informatiques:

### 1.4.1. Topologie bus

Une topologie physique en bus est une configuration où tous les périphériques sont reliés à un câble commun partagé. Ce type de réseau utilise généralement un câble long, appelé **dorsale**, auquel les ordinateurs (postes de travail et serveurs) sont directement connectés via des connecteurs hertziens terrestres. La dorsale est terminée à ses deux extrémités pour éviter que le signal ne soit réfléchi après avoir traversé tous les périphériques. La topologie en bus est la première utilisée pour connecter des ordinateurs dans un réseau, étant ainsi la forme la plus ancienne de topologie. Cependant, elle est vulnérable aux défaillances. Dans la plupart des configurations en bus, les signaux électriques ou électromagnétiques circulent dans les deux directions (Kumar & Deepa, 2015).

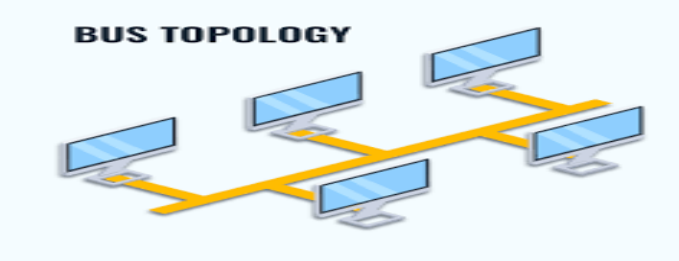


Figure 1.1 : Topologie bus [Web 01]

### 1.4.2. Topologie anneau

Les topologies en anneau sont configurées sous forme de cercle. Chaque nœud est relié à ses voisins, et les données circulent autour de l'anneau dans une seule direction. Chaque périphérique intègre un récepteur et un émetteur et sert de répéteur pour transmettre le signal au périphérique suivant sur l'anneau. Comme le signal est régénéré à chaque périphérique, sa dégénérescence est faible. Après un certain temps, la topologie en anneau est apparue. Pour éviter les inconvénients de la topologie en bus, la topologie en anneau a été inventée. Mais il s'agit également d'un modèle d'échec. Les topologies en anneau sont parfaitement adaptées aux méthodes d'accès par jeton. Le jeton circule sur l'anneau, et seul le nœud qui le détient peut transmettre des données. Les topologies en anneau sont assez rares (Kumar & Deepa, 2015).

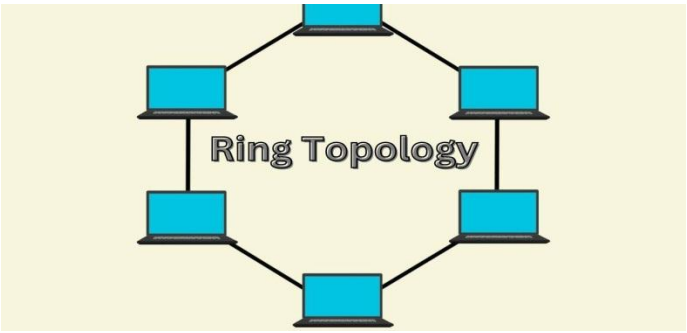


Figure 1.2 : Topologie anneau [Web 02]

### 1.4.3. Topologie en étoile

Les topologies en étoile utilisent un dispositif central avec des câbles de dérivation s'étendant dans toutes les directions. Chaque dispositif en réseau est connecté via une liaison point à point au dispositif central appelé concentrateur, répéteur multiport ou commutateur. De plus, les topologies en étoile peuvent être imbriquées dans d'autres étoiles pour former des topologies de réseau arborescentes ou hiérarchiques. Dans une topologie en étoile, les signaux électriques ou électromagnétiques circulent de l'appareil en réseau, via son câble de dérivation, jusqu'au commutateur, d'où ils sont transmis à un autre réseau. Pour éviter les inconvénients des topologies en bus et en anneau, la topologie en étoile a été inventée. Il ne s'agit pas d'un modèle de défaillance, mais d'un modèle standard, et cette topologie est aujourd'hui couramment utilisée partout (Kumar & Deepa, 2015).

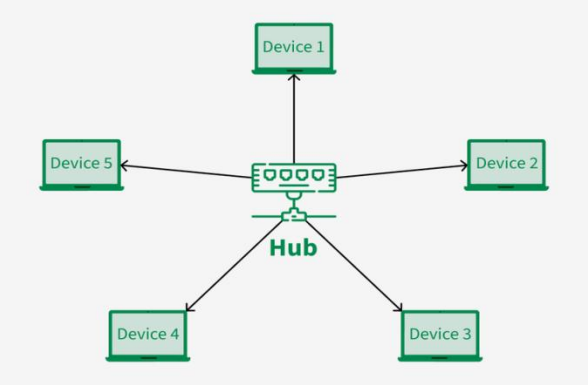


Figure 1.3 : Topologie étoile [Web 03]

### 1.4.4. Topologie en maillée

Un réseau maillé dispose d'une connexion point à point entre tous les appareils du réseau. Étant donné que chaque appareil nécessite une interface pour tous les autres appareils du réseau, les topologies maillées ne sont généralement pas considérées comme pratiques. Cependant, les réseaux maillés sont extrêmement tolérants aux pannes et chaque lien offre une capacité garantie (Kumar & Deepa, 2015).

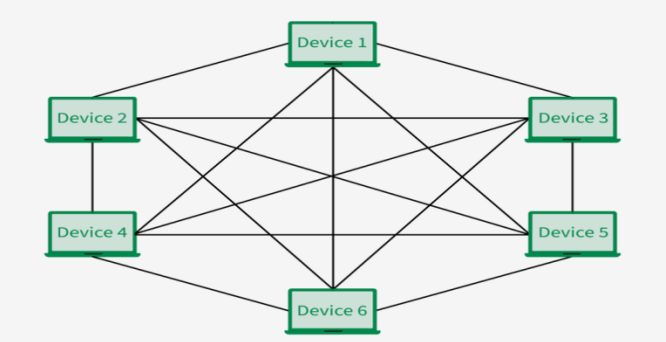


Figure 1.4 : Topologie maillée [Web 04]

### 1.4.5. Topologie hiérarchique

La topologie hiérarchique, ressemble beaucoup à la topologie en étoile, sauf qu'elle n'utilise pas de nœud central. Bien que Cisco préfère l'appeler « hiérarchique », on la désigne parfois sous le nom de topologie en arbre. Ce type de topologie présente le même défaut de centralisation que la topologie en étoile. Si l'appareil situé au sommet de la chaîne tombe en panne, c'est tout le réseau qui est hors service. Évidemment, cette méthode est peu pratique et peu utilisée dans les applications réelles (Jiang, 2015).

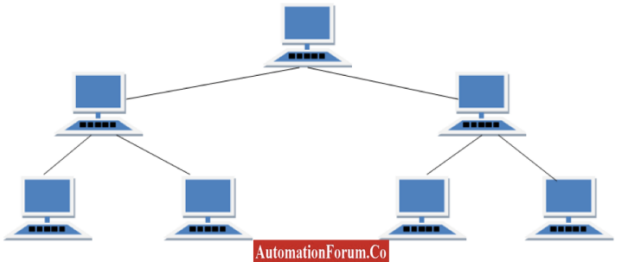


Figure 1.5 : Topologie hiérarchique [web 05]

### 1.4.6. Topologie arbre :

La topologie de réseau arborescent, utilise deux ou plusieurs réseaux en étoile interconnectés. Les ordinateurs centraux des réseaux en étoile sont connectés à un bus principal. Ainsi, un réseau arborescent est un réseau en bus de réseaux en étoile (Jiang, 2015).

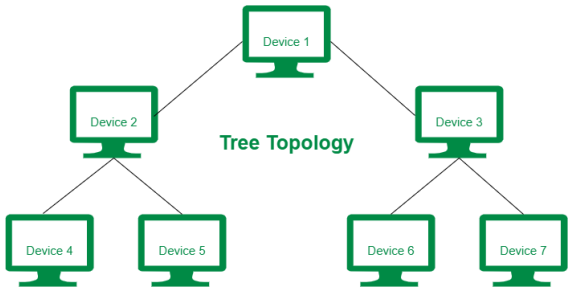


Figure 1.6 : Topologie arbre [web 06]

## 1.5. Le matériel

### 1.5.1. Les support de transmission

#### 1.5.1.1. Le cuivre : câbles coaxiaux ou à paires torsadées

Le **câble coaxial** est un type de câble utilisé pour la transmission de données, entre autres. Il se compose d'au moins deux conducteurs : une âme centrale, qui peut être soit un fil unique, soit plusieurs fils (en cuivre, cuivre argenté ou même acier cuivré). Cette âme est entourée d'un isolant diélectrique. Par-dessus, on trouve une tresse métallique conductrice (ou une feuille d'aluminium enroulée), suivie d'une gaine isolante servant de protection [WEB 07].

Un câble à **paires torsadées** décrit un modèle de câblage où une ligne de transmission est formée de deux conducteurs enroulés en hélice l'un autour de l'autre, Un câble peut contenir plusieurs paires torsadées [WEB 08]. Pour limiter les interférences, les paires torsadées sont souvent blindées. Comme le blindage est fait de métal, celui-ci constitue également un référentiel de masse. Le blindage peut être appliqué individuellement aux paires, ou à l'ensemble formé par celles-ci.

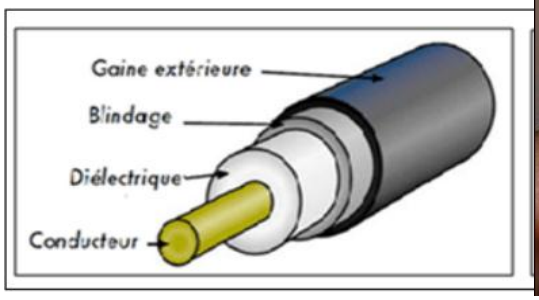


Figure 1.7 : Câble coaxiale (Bachir, 2016)

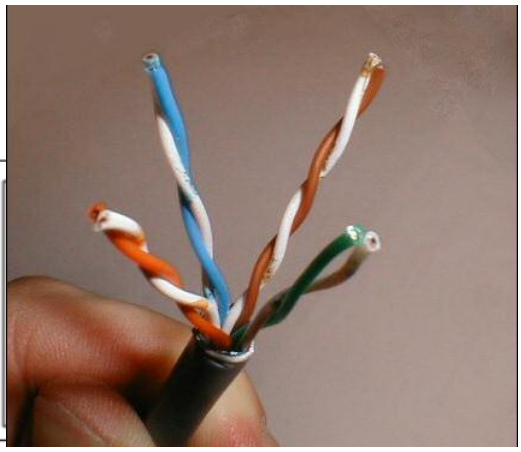

Figure 1.8 : Les paires torsadées [web 09]

### 1.5.1.2. Le verre : La fibre optique

La fibre optique, constituée d'un filament flexible en verre, convertit les signaux électriques en lumineux, permettant la transmission de données et de vidéos sur de longues distances (jusqu'à 100 km avant régénération). Elle est composée de plusieurs couches : un cœur en verre ou plastique qui transporte le signal, une gaine protectrice, un tampon pour protéger contre les dommages physiques, une couche métallique rigide (Amor) pour la sécurité physique, et enfin, une couche extérieure (Jack) qui protège contre l'humidité et l'abrasion (Onu, 2016).

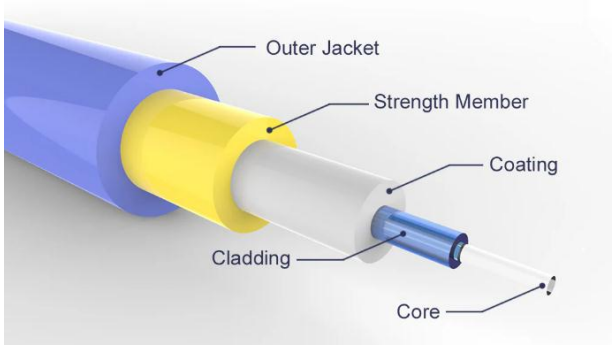


Figure 1.9 : Le câble de la fibre optique [WEB 10]

### 1.5.1.3. Ondes électromagnétiques : sans fil

Les supports sans fil utilisent des ondes électromagnétiques pour transmettre des signaux, mais leur portée peut être limitée par des obstacles comme les bâtiments ou le terrain. De plus, la transmission sans fil est sensible aux interférences et aux appareils courants. L'absence de câbles physiques facilite l'accès non autorisé au réseau, ce qui rend la sécurité essentielle dans les réseaux sans fil [WEB 07].

### 1.5.2. La carte réseau

La carte réseau (NIC) relie l'ordinateur au réseau et gère l'envoi et le contrôle des données. Elle possède généralement deux voyants lumineux : un vert pour l'alimentation et un orange ou rouge pour l'activité réseau. La carte utilise un transceiver pour convertir les données parallèles en série. Chaque carte a une adresse unique, appelée adresse MAC, pour l'identifier. Elle dispose également de paramètres configurables comme l'interruption matérielle et les adresses de mémoire [WEB 11].

### 1.5.3. Le commutateur ou switch

Un commutateur Ethernet est un pont multiport qui transfère de manière sélective les paquets d'un port LAN vers un l'autre (Walrand, J. et Varaiya, P. 2000). Il relie plusieurs segments de câbles ou de fibres optiques au sein d'un réseau informatique, permettant ainsi une communication fluide entre les différents appareils.

### 1.5.4. Le modem

Modem est l'abréviation de « Modulateur-Démodulateur ». Il s'agit d'un composant matériel qui permet à un ordinateur ou à un autre appareil, tel qu'un routeur ou un commutateur, de se connecter à l'internet. Il convertit ou « module » un signal analogique provenant d'un fil de téléphone ou de câble en données numériques (1s et 0s) qu'un ordinateur peut reconnaître. Aussi, il convertit les données numériques d'un ordinateur ou d'un autre appareil en un signal analogique qui peut être envoyé sur des lignes téléphoniques standard [WEB 12].

### 1.5.5. Routeur

Un routeur est un périphérique réseau qui transmet les paquets réseau d'un réseau à un autre (Tadimety, 2014) comme par exemple un réseau local et Internet. Souvent, un routeur inclut également une fonction de pare-feu (firewall) pour sécuriser le réseau en bloquant les connexions non autorisées et en filtrant le trafic indésirable.

### 1.5.6. Le firewall

Un pare-feu est un outil de sécurité qui régule les échanges entre deux réseaux distincts. Il est essentiel pour protéger les réseaux en contrôlant l'accès aux réseaux privés et publics, selon des règles prédéfinies. Sa principale fonction est de protéger les données internes et le bon fonctionnement du réseau, tout en surveillant et limitant le trafic de données. Le pare-feu est l'une des technologies de sécurité les plus utilisées et reconnues dans le domaine de la protection des réseaux (Wang, 2022).

### 1.5.7. Onduleur

Un onduleur, dans le contexte informatique, est un dispositif qui fournit une alimentation de secours aux équipements lors d'interruptions ou d'instabilités du réseau électrique, protégeant ainsi contre les pertes de données et les dommages matériels [WEB 13].

### 1.5.8. Le serveur

#### 1.5.8.1. Définition

Un serveur est un système qui répond à travers un réseau pour fournir, ou contribuer à fournir, un service réseau. Les serveurs peuvent être exécutés sur des ordinateurs dédiés, souvent appelés SERVEURS, mais de nombreux ordinateurs en réseau peuvent héberger des serveurs. Dans de nombreux cas, un ordinateur peut fournir de nombreux services et héberger plusieurs serveurs. Un serveur peut également être considéré comme un système qui répond aux requêtes d'un client (un autre système) via un réseau informatique pour fournir un service.

#### 1.5.8.2. Types de serveurs

**a. Serveurs d'applications**

Les serveurs d'applications fournissent aux clients du réseau le côté serveur des applications client/serveur, et souvent les données qui les accompagnent. Un serveur de base de données, par exemple, assure non seulement le traitement des requêtes et l'analyse des données, mais sert également de référentiel pour l'énorme volume de données souvent stocké dans les bases de données.

**b. Serveurs de fichiers et d'impression**

Les serveurs de fichiers et d'impression sont les piliers du monde des serveurs, car ils fournissent le stockage de fichiers réseau de base, des services de récupération et l'accès aux imprimantes en réseau – des fonctions qui définissent les utilisations fondamentales de la plupart des réseaux d'entreprise. Grâce à ces serveurs, l'utilisateur peut exécuter des applications localement tout en conservant les fichiers de données sur le serveur.

**c. Serveurs de courriers**

Les serveurs de courrier traitent les messages électroniques des utilisateurs ; cette fonction peut consister simplement à servir de centre d'échange pour les échanges locaux de messages. Cependant, les serveurs de courrier fournissent également généralement des services de stockage et de retransmission, dans lesquels les serveurs traitent les messages entrants en attendant que les utilisateurs y accèdent. De même, le serveur peut stocker les messages sortants jusqu'à l'établissement d'une connexion à un serveur de messagerie externe, puis les transférer à leur destination.

**d. Serveur web**

Un serveur web est conçu pour envoyer du contenu statique à un grand nombre d'utilisateurs. Les pages fournies par le serveur sont censées être identiques pour tous les visiteurs. Le serveur web est initialement conçu pour publier des documents statiques sur Internet. L'utilisateur demande une page web. Le serveur web recherche le fichier de la page web dans un répertoire local et le renvoie à l'utilisateur.

**e. Serveurs de base de données**

Un serveur de base de données sert principalement à stocker différents types de données afin que l'utilisateur puisse facilement accéder aux données stockées sur le serveur, localement ou à distance. Le serveur de base de données permet principalement à l'autre ordinateur d'accéder aux données qu'il contient et de les récupérer. La base de données fournit également d'autres services de traitement de données tels que l'analyse, la manipulation et l'archivage des données (Dave, 2017).

## 1.6. Organisation du réseau

On distingue deux types d'architecture de réseaux : le poste à poste et le client/serveur.

### 1.6.1. Le réseau client/serveur

#### 1.6.1.1. Définition

Un système client-serveur peut être défini comme une architecture logicielle composée d'un client et d'un serveur, où les clients envoient toujours des requêtes, tandis que le serveur répond aux requêtes envoyées. Ce système assure une communication interprocessus, car il implique l'échange de données entre le client et le serveur, chacun exécutant des fonctions différentes. Parmi les protocoles standardisés utilisés par les clients et les serveurs pour communiquer entre eux, on peut citer : le protocole de transfert de fichiers (FTP), le protocole SMTP (Simple Mail Transfer Protocol) et le protocole HTTP (Hypertext Transfer Protocol).

Le système client-serveur présente de nombreux avantages : il répartit le traitement des applications sur plusieurs machines et facilite le partage des ressources entre le client et les serveurs. Il réduit la réplication des données en les stockant sur chaque serveur plutôt que sur le client (Oluwatosin, 2014).

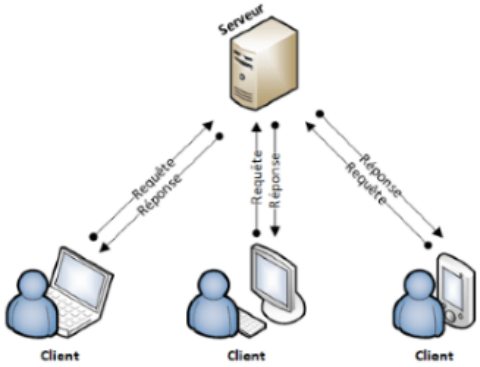


Figure 1.10 : Système client / serveur [Web 14]

#### 1.6.1.2. Exemple d'application client/serveur

**Transfert de fichiers :** Il s'agit de la transmission de fichiers entre le client et le serveur. Il permet également le stockage de fichiers sur le serveur. Il est possible d'y stocker des fichiers tels que des films, des images et de la musique.

**Transfert de courrier :** Il s'agit du transfert de messages tels que des e-mails à l'aide du protocole de transfert de courrier (MTP).

**Protocole de transfert hypertexte (HTTP) :** Transfert de fichiers multimédias tels que des images et du texte entre le client et le serveur. HTTP est utilisé pour améliorer la communication entre le client et le serveur, en servant de protocole de requête-réponse.

### 1.6.1.3. Types d'architecture client/serveur

Il existe différentes architectures client-serveur selon le nombre de serveurs impliqués dans l'implémentation. Parmi les architectures les plus courantes, on peut citer :

**a. Architecture système client-serveur à deux niveaux :** Cette architecture implique uniquement le serveur de base de données et un PC client. Dans une architecture à deux niveaux, les utilisateurs exécutent des applications sur leur PC (client), qui se connecte au serveur via un réseau. L'application cliente exécute le codage et la logique métier, puis affiche le résultat à l'utilisateur. On parle également de client lourd.

On parle de client lourd lorsque le client accède directement à la base de données sans intermédiaire.

On l'utilise également pour exécuter la logique applicative, le code applicatif étant attribué à chaque client du poste de travail.

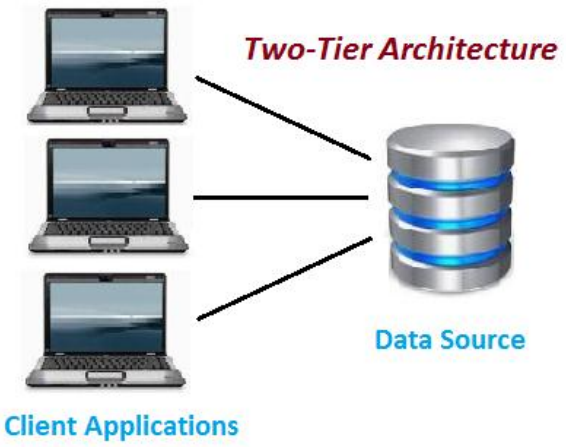


Figure 1.11 : architecture à deux tiers [Web 15]

**b. Architecture client-serveur à trois niveaux :** Cette architecture comprend le PC client, le serveur de base de données et le serveur d'applications. Elle peut être étendue à N niveaux, impliquant ainsi davantage de serveurs d'applications.

Dans cette architecture, le client ne contient que la logique de présentation, nécessitant ainsi moins de ressources et de codage.

Elle permet à un serveur de gérer plusieurs clients et de fournir davantage de ressources.

Elle implique un intermédiaire (serveur d'applications), également appelé middleware.

- **Middleware :** L'architecture à trois niveaux implique un serveur d'applications servant d'intergiciel entre le PC client et le serveur de base de données. Ce niveau d'intergiciel est un logiciel distinct exécuté sur une machine distincte et exécutant la logique applicative.

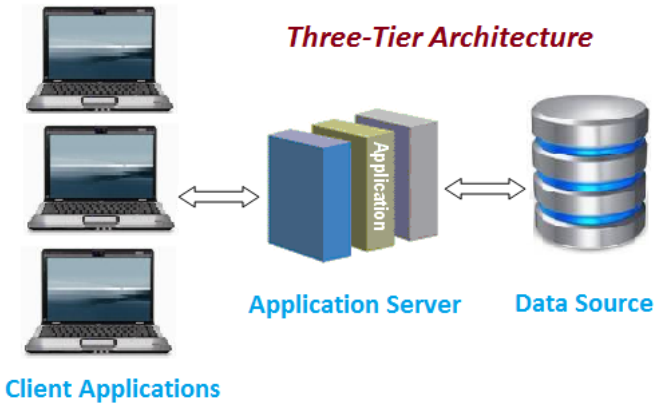


Figure 1.12 : architecture trois tiers [Web 15]

### 1.6.2. Le réseau poste à poste (peer to peer)

Dans cette architecture l'ordinateur ou logiciel joue le role du client et du serveur a la fois. Servent est un mot artificiel dérivé de la première syllabe du terme serveur (« Serv- ») et de la deuxième syllabe du terme client (« -ent »). Ce terme « Servent » représente donc la capacité des nœuds d'un réseau pair-à-pair à agir à la fois comme serveur et comme client. Ceci est totalement différent des réseaux client/serveur, au sein desquels les nœuds participants peuvent agir soit comme serveur, soit comme client, mais ne peuvent pas combiner les deux fonctions (Schollmeier, 2001).

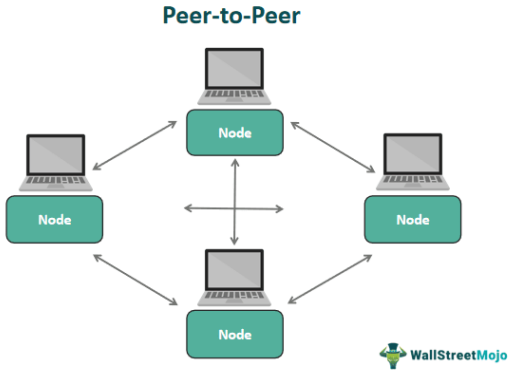


Figure 1.13 : architecture poste à poste [Web 16]

## 1.7.Conclusion

Dans ce chapitre nous avons présenté, le concept de réseau informatique en présentant les variétés de types et de topologies existantes. Nous avons également parlé des éléments matériels qui composent ces réseaux et en fin en a abordé l'organisation de ces réseaux, en nous intéressant à leur structure et à leur gestion, afin d'assurer leur bon fonctionnement et leur rendement maximal.

CHAPITRE 02

# MODELE OSI, TCP/IP ET PROTOCOLES DE TRANSPORT

## 2.1.Introduction:

Pour mieux gérer la complexité des nombreuses fonctions au sein des réseaux, les spécialistes ont structuré les opérations en plusieurs couches protocolaires, chacune remplissant des tâches spécifiques liées à la transmission des données. Parmi elles, la couche transport occupe une place centrale. Elle permet d'établir une communication logique entre hôtes distants, en fournissant des services à travers différents protocoles de transport, ce qui en fait un élément clé de l'architecture réseau.

Dans ce chapitre, nous explorerons le modèle OSI, avec un focus particulier sur la couche transport. Nous y analyserons en détail les deux protocoles fondamentaux qui la composent : TCP (Transmission Control Protocol) et UDP (User Datagram Protocol).

## 2.2. Définition :

Le modèle OSI constitue un cadre universellement reconnu qui sert de base à l'élaboration de normes ouvertes et complètes. Il facilite l'établissement de standards pour assurer l'interconnexion et la communication entre les systèmes, en particulier dans le secteur des technologies de l'information. Le modèle de référence OSI est conceptuellement divisé en sept couches, chacune ayant des fonctions réseau spécifiques. Ce modèle a été créé sur la base d'une proposition de l'ISO, comme première étape vers la normalisation des protocoles internationaux utilisés sur les différentes couches. (Permana & al, 2024) Le modèle de référence OSI adopte une structure en couches pour représenter les réseaux. Chaque couche correspond à un aspect distinct de la communication. Cette séparation permet de clarifier les interactions entre les logiciels et le matériel, tout en rendant plus aisée l'identification et la résolution des problèmes grâce à une méthode systématique pour analyser le fonctionnement des différents composants.

## 2.3. Les sept couches du modèle OSI :

Le modèle de référence OSI a été développé afin d'organiser la communication des données en étapes séparées et bien définies, appelées « couches ». Il est structuré en sept couches principales, chacune jouant un rôle spécifique dans le processus de transmission.

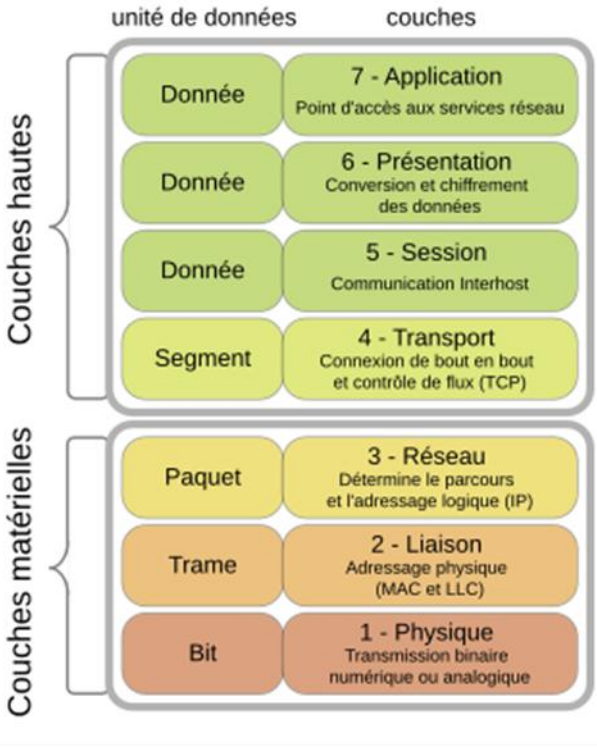


Figure 2.1 : Les couches de modèle OSI [Web 17]

**A. Couche application**

Il s'agit de la couche où l'utilisateur interagit avec l'ordinateur à l'aide de logiciels d'application comme Internet Explorer (Agrawal & al, 2021). Cette couche regroupe l'ensemble des services et protocoles nécessaires aux logiciels ou systèmes d'exploitation pour assurer la communication sur le réseau. Elle fournit les services qui interagissent directement avec les applications utilisées par l'utilisateur (comme Google Chrome, Zoom, Skype ou Outlook) et traite ce que l'utilisateur voit ou effectue. Elle permet, par exemple, la gestion de la messagerie électronique et des e-mails, ainsi que le transfert de fichiers. Les différentes applications utilisent différents protocoles de couche application. Les navigateurs web (Chrome, Firefox, etc.) utilisent un protocole de couche application appelé HTTP (Hyper Text Transfer Protocol) ou HTTPS pour naviguer sur le web. La messagerie électronique utilise le protocole POP3 (Post Office Protocol version 3) pour lire les e-mails et le protocole SMTP (Simple Mail Transport Protocol) pour en envoyer. Le protocole de transfert de fichiers (FTP) est utilisé pour transférer des fichiers (Amin et Rahman, 2023).

**B. Couche présentation :**

La couche de présentation reçoit les données de la couche application. Dans cette couche, les données passent par trois étapes ce sont conversion, compression, chiffrement. La conversion c-à-d les données sont converties en un format lisible par machine (combinaison de 0 et 1) pour les messages sortants, et en un format binaire lisible par machine (combinaison de 0 et 1) pour les messages entrants (Amin et Rahman, 2023), ce processus est aussi appelé traduction. Avant la transmission cette couche réduit le nombre de bit utilisés pour présenter les données d'origine cette réduction permet le fichier d'atteindre sa destination plus

rapidement. C'est très utile pour le streaming vidéo ou audio en temps réel. Avant d'être transmises, les données sont chiffrées pour garantir leur intégrité et renforcer leur sécurité. Ce chiffrement est réalisé du côté de l'expéditeur, tandis que le déchiffrement s'effectue chez le destinataire. Dans la couche de présentation, le protocole SSL (Secure Sockets Layer) est utilisé pour assurer ces opérations. Ce mécanisme protège les informations contre toute tentative d'interception.

**C. Couche session**

La couche session établit et gère la connexion entre deux appareils en autorisant l'envoi et la réception de données, puis l'ouverture et la fermeture de la session. Elle utilise des ports pour établir la communication entre deux appareils. Le temps écoulé entre l'ouverture et la fermeture de cette connexion est appelé session. Cette couche garantit que la session restera ouverte jusqu'à ce que toutes les données soient transférées, puis la ferme pour éviter le gaspillage de ressources (Amin et Rahman, 2023). La couche session facilite aussi la synchronisation des données à l'aide de points de contrôle. En cas d'interruption ou de déconnexion, la transmission peut reprendre à partir du dernier point de contrôle, évitant ainsi de devoir tout recommencer. Sans ces points de contrôle, l'intégralité du transfert devrait être relancée depuis le début.

**D. Couche transport**

La couche transport gère la transmission des données entre deux appareils en assurant leur découpage (segmentation), leur envoi et leur réassemblage. Chaque segment comprend des numéros de port pour identifier l'application concernée, ainsi qu'un numéro de séquence pour reconstituer les données dans le bon ordre. Elle régule le flux de données pour éviter la surcharge du récepteur et corrige les erreurs en retransmettant les segments perdus. Deux protocoles sont utilisés : TCP, qui garantit une transmission fiable avec accusé de réception (idéal pour le Web, les e-mails, ou FTP), et UDP, plus rapide mais sans vérification d'arrivée, utilisé pour le streaming et les jeux en ligne.

**E. Couche réseau**

La couche réseau récupère les segments de données de la couche transport et les transmet d'un appareil à un autre situé sur deux réseaux différents. Elle décompose les segments de données en unités plus petites, appelées paquets, dans l'appareil émetteur et les réassemble sur l'appareil récepteur. Cette couche remplit principalement trois fonctions : 1) l'adressage logique, 2) le routage et 3) la détermination du chemin (Amin et Rahman, 2023). L'adressage IP, qu'il s'agisse d'IPv4 ou d'IPv6, se déroule au niveau de la couche réseau et est appelé adressage logique. Chaque appareil connecté à un réseau dispose d'une adresse IP unique, qui sert à son identification. Ces adresses sont attribuées pour garantir que chaque paquet de données soit dirigé vers le bon destinataire. L'adresse IP peut être comparée à celle d'un appartement, permettant ainsi de localiser et d'atteindre un appareil précis. Le routage, quant à lui, est le processus de transfert des paquets de données de la source vers la destination en fonction de l'adresse IP. Un appareil peut se connecter à un serveur Internet ou à un autre appareil par différents chemins. La sélection du meilleur itinéraire pour la

transmission des données entre l'expéditeur et le récepteur est appelée détermination du chemin. Les protocoles comme le BGP (Border Gateway Protocol), l'OSPF (Open Shortest Path First) et l'IS-IS (Intermediate System to Intermediate System) sont utilisés pour définir le chemin optimal pour la circulation des données.

**F. Couche liaison de données**

Cette couche gère le transfert des données entrantes et sortantes de la couche physique, ainsi que les notifications d'erreur et la topologie du réseau. L'unité de données du protocole de cette couche est la trame (Agrawal & al, 2021). La couche liaison de données convertit les paquets reçus de la couche réseau en trames, un format adapté pour la transmission au niveau de la couche physique. Cette conversion permet de structurer les données de manière à ce qu'elles puissent circuler efficacement sur le support physique, qu'il s'agisse de câbles, de signaux sans fil ou d'autres types de transmission. En plus de cette fonction, la couche liaison joue un rôle essentiel dans la détection et la correction des erreurs qui peuvent survenir durant la transmission. Elle vérifie l'intégrité des données transmises et, lorsqu'elle identifie des erreurs, elle tente de les corriger ou de demander une retransmission, dans le but de garantir une communication fiable entre les appareils du réseau.

**G. Couche physique**

La couche physique englobe l'ensemble des composants matériels impliqués dans le transfert des données, tels que les câbles, les connecteurs, les cartes réseau ou encore les antennes. Elle est responsable de la transmission des flux binaires (suites de 0 et 1) sous forme de signaux adaptés au support de communication. Du côté de l'émetteur, elle convertit les données binaires en signaux physiques, tandis que du côté du récepteur, elle effectue l'opération inverse, en transformant les signaux reçus en flux binaires exploitables par les couches supérieures. Comme mentionné précédemment, les données issues de la couche application ont été segmentées par la couche transport, encapsulées sous forme de paquets dans la couche réseau, puis structurées en trames par la couche liaison de données. Ces trames, composées d'un flux binaire, sont finalement prises en charge par la couche physique, qui les convertit en signaux pour les transmettre à travers un support de transmission (cuivre, fibre optique, ondes radio, etc.). La nature du signal dépend du type de support utilisé : électrique pour les câbles en cuivre, lumineux pour les fibres optiques, et électromagnétique (ondes radio) pour les connexions sans fil.

**2.4. Encapsulation :**

L'encapsulation est le processus par lequel les informations d'une couche supérieure du modèle sont insérées dans le champ de données d'une couche inférieure. Lorsqu'un message quitte une station réseau, il transite de la couche 7 à la couche 1. Les données créées par la couche application sont transmises à la couche présentation. Cette dernière récupère les données de la couche application et y ajoute son en-tête et sa fin de bande. Ces données sont ensuite transmises à la couche session, qui ajoute son en-tête et sa fin de bande et les transmet à la couche transport. Le processus se répète jusqu'à ce que les données atteignent la couche physique. La couche physique ne se soucie pas de la signification des données. Elle se

contente de les convertir en bits et de les placer sur le support de transmission. (Shimonski & al, 2002).

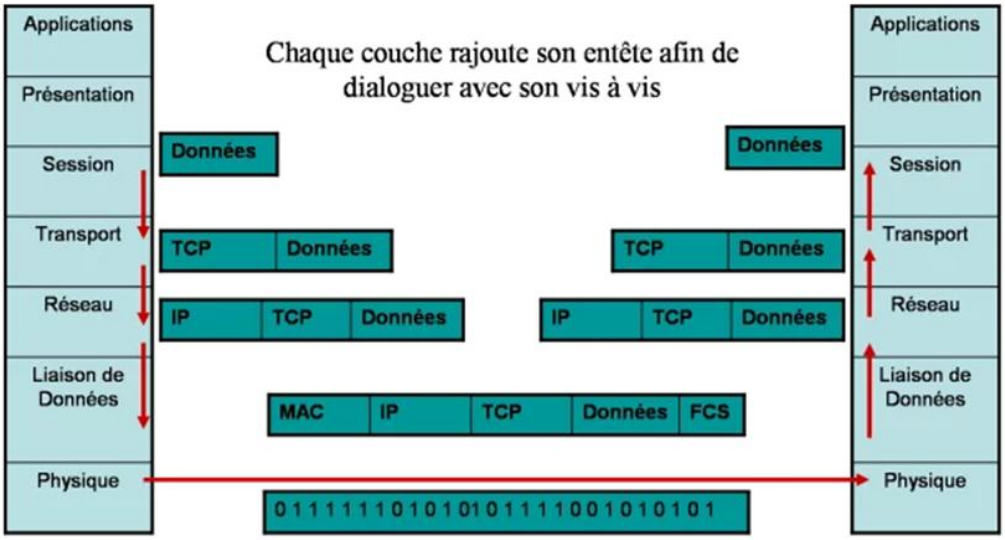


Figure 2.2 : Le processus d'encapsulation [Web 18]

## 2.5. Modèle TCP\IP :

### 2.5.1. Définition :

La suite de protocoles TCP/IP (Transmission Control Protocol/Internet Protocol) représente l'ensemble des normes et règles qui régissent les communications sur Internet. Grâce à sa flexibilité, sa fiabilité et ses performances, elle a été largement adoptée dans divers systèmes informatiques, y compris dans les milieux professionnels et institutionnels. Microsoft, qui avait initialement développé ses propres protocoles de communication, a progressivement intégré TCP/IP, d'abord pour le transport de données, puis pour prendre en charge un ensemble plus large de services réseau.

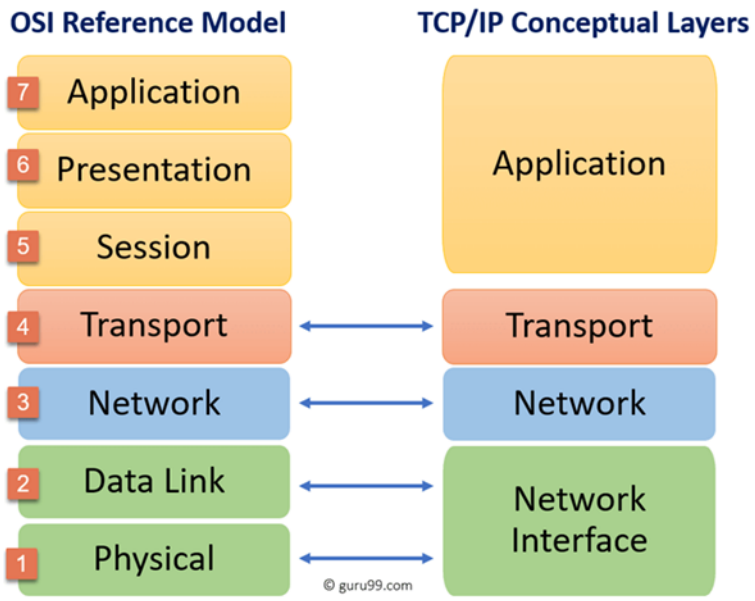


Figure 2.3 : Le modèle OSI vs TCP/IP [Web 19]

**A. Couche d'interface réseau** : la couche d'interface réseau (également appelée couche d'accès réseau) est chargée de placer les paquets TCP/IP sur le support système et d'accepter les paquets TCP/IP provenant de ce support. TCP/IP a été conçu pour être indépendant de la technique d'accès au système, de l'organisation du boîtier et du support (Mundra & Eltaeib, 2015).

**B. Couche Internet :** La couche Internet a pour mission d'échanger des datagrammes au-delà des limites du réseau. Elle fournit une interface réseau uniforme qui masque la topologie (disposition) réelle des connexions réseau sous-jacentes (Nath & Uddin, 2015). C'est pour cette raison qu'on la qualifie aussi de couche d'interconnexion des réseaux, car elle joue un rôle central dans la définition et la mise en place d'Internet. Elle spécifie les mécanismes d'adressage et de routage employés dans la suite de protocoles TCP/IP. Le protocole clé à ce niveau est le protocole Internet, responsable de l'attribution des adresses IP. Sa fonction de routage consiste à acheminer les datagrammes vers le routeur IP suivant, situé sur un réseau rapprochant les données de leur destination finale.

**C. Couche transport :** La couche transport (aussi appelée couche de transport d'hôte à hôte) est chargée de fournir la session et le datagramme à la couche application. Les principaux protocoles de cette couche sont TCP et UDP (Mundra et Eltaeib, 2015).

• TCP fournit un service de communication fiable, orienté connexion et un à un.

• UDP fournit un service de communication fiable, un à un ou un à plusieurs, sans connexion.

**D. Couche applicative :** La couche applicative du modèle TCP/IP combine les fonctionnalités des couches application, présentation et session du modèle OSI (Murkomen, 2024). La couche application est celle qui interagit directement avec les utilisateurs et assure le transfert des données à travers le réseau. Elle s'appuie sur des logiciels pour établir la communication. La couche présentation, quant à elle, est chargée de formater les données afin

qu'elles puissent être correctement interprétées par l'appareil de destination, en prenant en charge des opérations comme la compression, la décompression, le chiffrement et le déchiffrement. Cette couche caractérise les protocoles utilisés par les applications pour échanger des informations tel que :

- HTTP est utilisé pour échanger des documents constituant les pages Web.
- FTP est utilisé pour l'échange de données intelligentes.
- SMTP est utilisé pour l'échange de messages et de connexions.
- Telnet est utilisé pour la connexion à distance aux systèmes.

### 2.5.2. La couche de transport du modèle TCP\IP :

La couche transport est un composant essentiel de la suite de protocoles TCP/IP, offrant une solution complète de bout en bout pour des communications fiables. TCP/IP s'appuie sur la couche transport pour contrôler efficacement les communications entre deux hôtes. (Woodberg & al 2007) Elle est responsable de l'établissement et de la terminaison des sessions de communication IP et c'est là que les ports TCP/IP écoutent les connexions entrantes. La couche transport utilise des numéros de port standardisés pour faciliter la communication entre les applications courantes.

### 2.5.3. Les protocoles de la couche transport :

Les protocoles de la couche transport qui assurent la communication de bout en bout pour les services sont appelés protocoles de couche transport (TLP). Ces protocoles peuvent être largement divisés en deux types TCP et UDP.

#### 2.5.3.1. TCP :

Le protocole TCP (Transmission Control Protocol) est une norme de communication qui permet aux programmes applicatifs et aux dispositifs informatiques d'échanger des messages sur un réseau. Il est conçu pour envoyer des paquets sur Internet et assurer la transmission réussie des données et des messages sur les réseaux [Web 20].

Le protocole TCP constitue l'une des normes fondamentales qui régissent le fonctionnement d'Internet, faisant partie des standards définis par l'Internet Engineering Task Force (IETF). Il est largement utilisé dans les communications numériques en réseau et assure la transmission fiable des données d'un point à un autre. TCP structure les données afin qu'elles puissent être échangées entre un client et un serveur, tout en garantissant leur intégrité. Avant l'envoi, il établit une connexion entre la source et la destination, maintenant cette connexion active pendant toute la durée de la communication. Il segmente ensuite les volumes de données en petits paquets, en veillant à leur cohérence tout au long du transfert. En conséquence, les protocoles applicatifs qui doivent transmettre des données s'appuient généralement sur TCP. C'est le cas, par exemple, des systèmes de partage peer-to-peer comme FTP (File Transfer Protocol), SSH (Secure Shell) ou Telnet. TCP est également utilisé pour l'envoi et la réception d'e-mails via IMAP (Internet Message Access Protocol), POP (Post Office

Protocol) et SMTP (Simple Mail Transfer Protocol), ainsi que pour l'accès au Web via HTTP (Hypertext Transfer Protocol).

**2.5.3.2. Format paquet TCP :**

**Port source (16 bits) :** Contient le numéro de port de l'application source/transmetrice et permet de déterminer l'application vers laquelle la livraison des données est prévue.

**Port de destination (16 bits) :** Ce champ contient le numéro de port de l'application émettrice et permet d'envoyer les données à l'application appropriée.

**Numéro de séquence (32 bits) :** Il garantit la réception des données dans le bon ordre grâce à une segmentation ordonnée et à leur réassemblage à la réception.

**Numéro d'accusé de réception (32 bits) :** Ce champ contient le numéro de séquence à venir et accuse réception des retours jusqu'à ce numéro.

**Décalage des données (4 bits) :** Le champ de décalage des données indique le point de départ de la charge utile des données TCP et stocke également la taille de l'en-tête TCP.

**Indicateurs de contrôle (9 bits) :** TCP utilise plusieurs indicateurs de contrôle pour réguler la communication. Parmi les indicateurs importants, on peut citer :

**SYN (Synchronisation) :** Responsable de la connexion entre l'expéditeur et le destinataire.

**ACK (Accusé de réception) :** Son objectif est de transmettre l'accusé de réception des données par l'expéditeur.

**FIN (Finish) :** Indique si la connexion TCP est terminée ou non.

**RST (Reset) :** Utilisé principalement pour réinitialiser la connexion en cas d'erreur.

**Taille de la fenêtre (16 bits) :** Cette propriété spécifie la taille de la fenêtre de réception de l'expéditeur.

**Somme de contrôle (16 bits) :** Indique si l'en-tête a été endommagé pendant le transport.

**Pointeur urgent (16 bits) :** Ce champ pointe vers le premier octet de données urgentes du paquet.

**Options (longueur variable) :** Ce champ représente les différentes options TCP.

**Charge utile des données :** Ce champ contient principalement les informations relatives aux données applicatives transmises [Web 21].

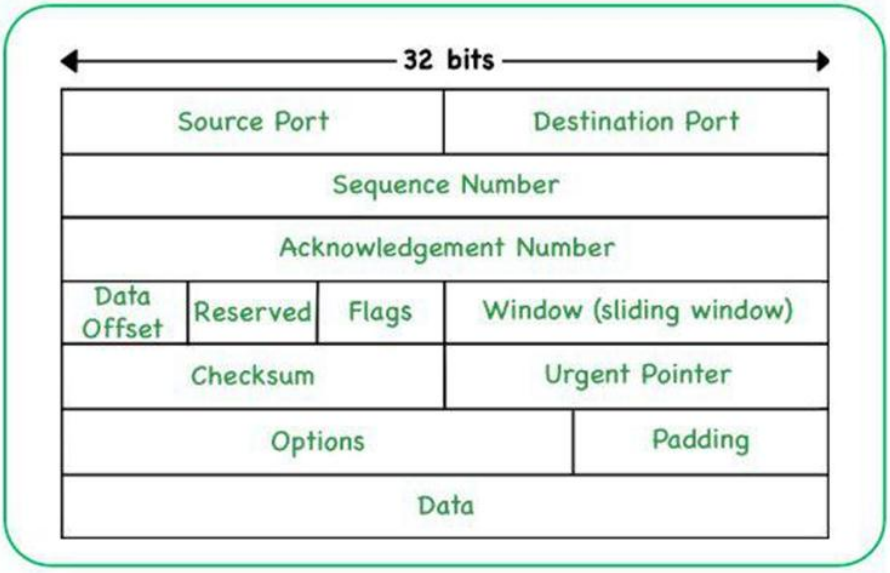


Figure 2.4 : Format paquet TCP [Web 22]

**2.5.3.3. Cycles de sessions TCP :**

**A. Établissement d'une connexion :**

L'algorithme utilisé par TCP pour établir et terminer une connexion est appelé «three-way handshake ». Deux parties qui établissent une connexion TCP échangent des numéros de séquence initiaux pour leurs flux respectifs. Le client envoie d'abord un segment avec le drapeau SYN et un numéro de séquence initial x. Le serveur répond avec un segment contenant les drapeaux SYN et ACK, accusant réception de x et indiquant son propre numéro de séquence y. Enfin, le client envoie un ACK pour y. Chaque accusé de réception (ACK) indique le prochain numéro de séquence attendu (ex. : x + 1), ce qui implique l'acceptation des données précédentes. Un minuteur est associé aux premiers segments pour permettre la retransmission en cas de non-réponse. Le choix aléatoire des numéros de séquence initiaux, exigé par TCP, permet d'éviter que d'anciennes connexions n'interfèrent avec de nouvelles utilisant les mêmes numéros (Peterson, Davie, 2022).

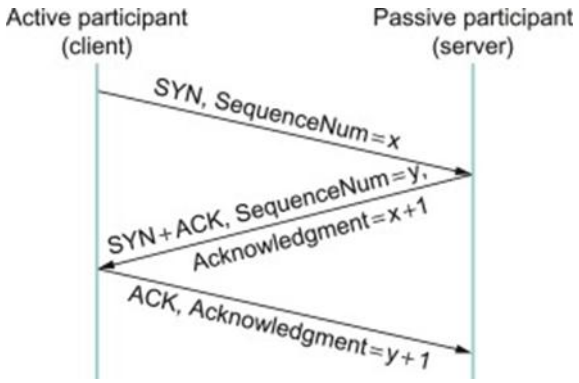

Figure 2.5: The three way hand shake (Peterson, Davie, 2022)

**B. Transfert de données :**

Pendant la phase de transferts de données, certains mécanismes clefs permettent d'assurer la robustesse et la fiabilité de TCP. En particulier, les numéros de séquence sont utilisés afin d'ordonner les segments TCP reçus et de détecter les données perdues, les sommes de contrôle permettent la détection d'erreurs, et les acquittements ainsi que les temporisations permettent la détection des segments perdus ou retardés [Web 22].

**C. Terminaison d'une connexion :**

La phase de terminaison d'une connexion utilise un handshaking en quatre temps, chaque extrémité de la connexion effectuant sa terminaison de manière indépendante. Ainsi, la fin d'une connexion nécessite une paire de segments FIN et ACK pour chaque extrémité [Web 22].

**2.5.3.2. UDP :**

**2.5.3.3. Définition :**

Le User Datagram Protocol (UDP) est un protocole de transport sans connexion, mis au point pour permettre une transmission simple et rapide des données entre deux entités. Il ne fournit aucun mécanisme de contrôle d'erreurs, ce qui signifie qu'il ne garantit ni la fiabilité, ni l'ordre, ni la détection de pertes de messages. De ce fait, UDP est considéré comme un protocole non fiable, mais il reste adapté aux applications où la vitesse est importante que la fiabilité, comme le streaming audio ou vidéo. (Larzon et al, 2004).

**2.5.3.4. Format paquet UDP :**

Le champ Source Port occupe 16 bits. Il indique :

- le numéro de port du processus émetteur,

- le numéro de port où on peut adresser les réponses lorsque l'on ne dispose d'aucun autre renseignements.

- si sa valeur est 0, cela signifie qu'aucun numéro de port n'est attribué.

Le champ Destination Port identifie le processus correspondant à l'adresse IP de destination auquel on envoie les données UDP. UDP effectue le démultiplexage des données à l'aide de numéros de port. Lorsque l'UDP reçoit un datagramme sans numéro de port, il génère un message d'erreur ICMP indiquant qu'il est impossible de contacter le port et il rejette le datagramme. Le champ Longueur contient la longueur du paquet UDP en octets (en-tête + données). La valeur minimale est 8 et correspond à un paquet où le champ de données est vide.

Le pseudo en-tête de préfixe de l'en-tête UDP contient l'adresse d'origine, l'adresse de destination, le protocole (UDP = 17) et la longueur UDP. Ces informations sont destinées à prévenir les erreurs de routage. (Christian bulfone).

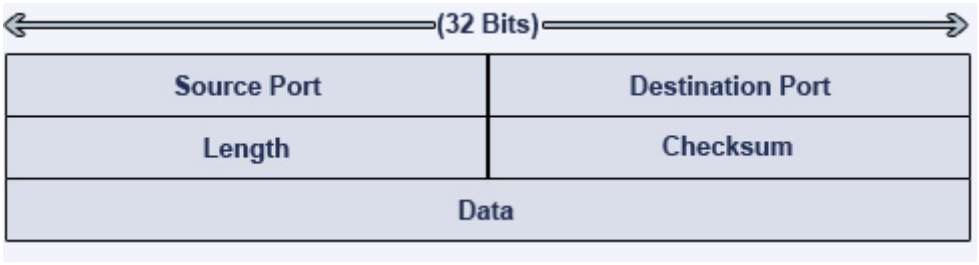


Figure 2.6: Format paquet UDP [Web 23]

## 2.6. Conclusion:

Ce chapitre débute par la présentation du modèle OSI et de ses sept couches, en expliquant le rôle spécifique de chacune dans le processus de communication réseau. Nous avons ensuite montré comment ce modèle a été adapté et simplifié pour donner naissance au modèle TCP/IP, plus utilisé dans les réseaux actuels. Une attention particulière a été portée sur le parcours de l'information à travers les couches du modèle TCP/IP, afin de mieux comprendre la manière dont les données sont transmises entre les dispositifs. La suite du chapitre est consacrée à une étude approfondie de la couche transport, considérée comme un élément central dans la gestion des communications réseau. Cette couche joue un rôle crucial en assurant le transfert des données entre les hôtes de manière fiable ou rapide selon le protocole utilisé. Dans ce contexte, nous avons détaillé les deux principaux protocoles de transport : TCP (Transmission Control Protocol) et UDP (User Datagram Protocol). Bien qu'ils appartiennent à la même couche, ces protocoles présentent des mécanismes de fonctionnement différents, adaptés à des besoins spécifiques.

CHAPITRE 03

# ADAPTATION DU PROTOCOLE TCP POUR LES APPLICATIONS MULTIMEDIAS TEMPS REEL CROSS-LAYER

## 3.1. Introduction :

Le protocole TCP (Transmission Control Protocol) est l'un des protocoles de la couche transport du modèle OSI. Il se distingue par sa fiabilité, car il garantit la livraison de tous les paquets en attendant un accusé de réception pour chacun d'eux. Cependant, cette fiabilité engendre une latence importante, ce qui constitue une limitation majeure pour les applications en temps réel, notamment les applications multimédias interactives telles que la visioconférence, le streaming en direct ou les jeux en ligne. L'objectif est donc de trouver un compromis entre les avantages de TCP (notamment la fiabilité) et les exigences de faible latence propres aux applications en temps réel.

Ce chapitre examine les limitations structurelles du TCP face à ces exigences, explore les approches d'adaptation possibles, notamment via une communication inter-couches (cross-layer), et présente une solution concrète permettant à la couche application de signaler à TCP la nature temps réel des données afin de modifier dynamiquement son comportement.

## 3.2. Les applications temps réel multimédias :

Une application temps réel multimédia est une application qui diffuse en temps réel des contenus multimédias (comme la vidéo ou l'audio), qui nécessite des performances réseau ininterrompues et un réseau réactif, notamment en termes de faible latence et de faible perte de paquets, pour assurer une qualité perçue optimale par les utilisateurs finaux (Mu, 2009). Les systèmes temps réel sont traditionnellement classés en temps réel dur et temps réel souple : dans la première catégorie, on trouve les systèmes temps réel critiques pour la sécurité, où manquer une échéance peut avoir des conséquences catastrophiques, tandis que dans la seconde, il s'agit de systèmes pour lesquels il faut optimiser la qualité de service fournie à l'utilisateur (Lipari & Palopoli, 2015). Les applications temps réel multimédias peuvent aussi être catégorisées en temps réel souple et temps réel dur, mais la pluparts de ces applications appartiennent à la catégorie temps réel souple. Il existe plusieurs types d'applications temps réel multimédias, parmi lesquels on peut citer : la visioconférence, les appels VoIP (voix sur IP), les jeux vidéo en ligne, la télémédecine (temps réel critique dans certains cas), ainsi que les véhicules autonomes ou les systèmes d'aide à la conduite.

## 3.3. Les exigences des applications temps réel multimédias :

Pour assurer une expérience utilisateurs fluide et satisfaisante ces applications nécessitent des performances réseau et système élevées. Ainsi, il est essentiel de comprendre les principales exigences qu'elles imposent aux infrastructures matérielles et logicielles.

### 3.3.1. Faible latence :

Latence c'est le temps de transmission d'un paquet entre l'émetteur et le récepteur, ce temps doit être minimal. Une latence élevé une qualité très basse surtout dans les appels audio et vidéos. Il est donc très essentiel que les mécanismes de sécurité réseau n'introduisent pas de délais supplémentaires importants.

### 3.3.2. Faible perte de paquets :

La perte de paquets et aussi un point sensible surtout dans les applications temps réels multimédias même si une simple perte de paquet sera affecté la qualité car elle produit des coupures audio ou geler l'image. Cette peut aggravée si les mécanismes de contrôle de flux ou de retransmission ne sont pas adaptés au temps réel.

### 3.3.3. Qualité de service (QoS) :

La qualité de service représente la capacité du réseau à fournir un service constant et prévisible. Les applications temps réel nécessitent une QoS élevée, même dans des environnements réseau hétérogènes donc le contrôle bout-en-bout est crucial.

### 3.3.4. Sécurité adaptée aux contraintes temps réel :

C'est-à-dire que les applications multimédias en temps réel nécessitent une sécurité robuste mais avec une faible latence, ça veut dire que chiffrement de bout-en-bout est indispensable mais sans ajout de retard. Les solutions de sécurité doivent également être compatibles avec les capacités de traitement des dispositifs utilisés, quel que soit dans les environnements mobiles ou embarqués.

### 3.3.5. Continuité de service pendant la mobilité (handover vertical) :

Dans un contexte de mobilité, comme le passage d'un réseau Wi-Fi vers une connexion 4G, il est crucial d'assurer un handover (transfert) rapide, sécurisé et sans interruption. Pour cela, des mécanismes d'authentification anticipée et de pré-négociation des connexions doivent être prévus afin de garantir la continuité du service sans dégradation perceptible.

### 3.3.6. Protection contre les attaques internes et externes :

Des attaques interne (utilisateurs compromis, logiciels malveillants) ou externe (attaques par déni de service, interceptions), tout en évitant les bouchons réseau ou le ralentissement dû à la surcharge de chiffrement.

### 3.3.7. Gestion efficace de la mobilité et de la signalisation :

Signalisation rapide pour établir les tunnels VPN sans délai perceptible (Bou Diab, 2010).

## 3.4. Limites TCP pour les applications temps réel :

### 3.4.1. TCP privilégie la fiabilité sur la temporalité :

La convention veut que TCP ne soit pas adapté aux applications temps réel, car il accorde plus d'importance à la fiabilité (assurer que les paquets arrivent sans erreur) qu'à la rapidité (assurer que les paquets arrivent dans les délais impartis).

### 3.4.2. Problèmes de latence :

TCP introduit des mécanismes de contrôle de congestion et de retransmission en cas de perte de paquets, ce qui peut entraîner des délais considérables, une latence élevée, et une perte de performance, des caractéristiques indésirables dans les applications temps réel où la réactivité est cruciale.

### 3.4.3. Limitation dans les applications en temps réel :

En raison de ces mécanismes de correction d'erreurs, TCP n'est pas conçu pour fonctionner de manière optimale dans des environnements nécessitant une faible latence et une faible perte de paquets, comme dans les systèmes multimédia en temps réel (vidéoconférence, VoIP, etc.) (Liang & Cheriton, 2002).

## 4.4. Adaptation TCP pour les applications temps réel multimédia par cross-layer :

### 4.4.1. Définition Cross-layer :

La conception inter-couches (cross-layer) s'est imposée comme une méthode viable pour améliorer les performances des réseaux de communication modernes. Elle permet d'accroître les performances globales du réseau, de minimiser les délais et d'améliorer la qualité de service des applications en combinant les fonctionnalités de plusieurs couches de la pile de protocoles réseau (Prakash, 2023).

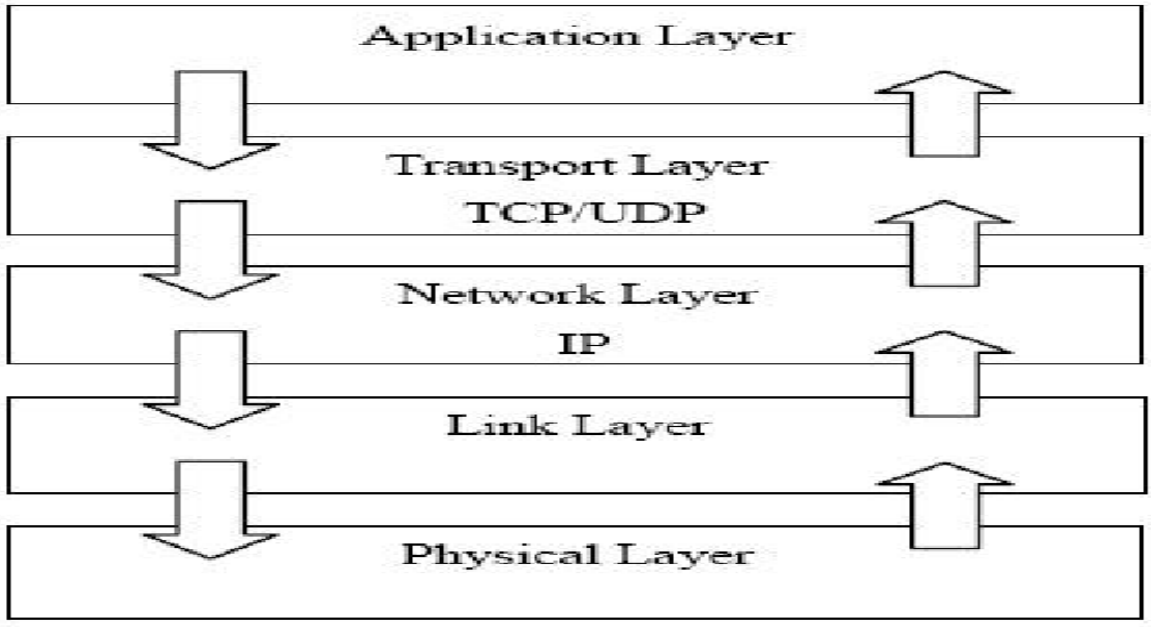


**Figure 3.1** Cross-layer TCP/IP (Lahane & Jariwala, 2018)

### 4.4.2. Différences entre architecture en couches et approche Cross-Layer :

Le modèle en couches, tel que défini dans le modèle OSI ou TCP/IP, favorise la modularité mais peut limiter l'efficacité dans des contextes dynamiques. Selon Srivastava et Motani [1], cette séparation stricte entre les couches empêche l'adaptation rapide nécessaire aux applications sensibles au délai, comme le multimédia temps réel. L'approche Cross-Layer vient justement répondre à ces limites en autorisant une collaboration inter-couches pour une meilleure performance globale (Srivastava & Motani, 2005).

Cependant, cette flexibilité n'est pas sans inconvénients. En brisant l'isolation stricte des couches, le Cross-Layer Design peut introduire de la complexité, rendre le débogage plus difficile, et menacer la portabilité des protocoles (Wang & Min, 2010) C'est pourquoi il est essentiel de concevoir des mécanismes Cross-Layer bien encadrés, où le flux d'information entre les couches est maîtrisé.

Dans notre cas d'étude, cette approche permet à la couche application d'indiquer à la couche transport (TCP) que les données transmises sont de type temps réel multimédia. Ainsi, TCP peut adapter dynamiquement son comportement (agrandir la fenêtre de réception, réduire les délais d'attente, etc.), ce qui serait difficile à faire dans un modèle purement en couches (Tsaoussidis & Georgiadis, 2007).

### 4.4.3. Avantages de la communications inter-couches (Cross-Layer) :

L'approche Cross-Layer offre de nombreux avantages, en particulier dans les systèmes de communication où les exigences en termes de latence, de bande passante et de fiabilité sont critiques, comme c'est le cas des applications multimédias en temps réel. Contrairement à l'architecture en couches traditionnelle, où chaque couche fonctionne indépendamment, la communication inter-couches permet un échange direct d'informations contextuelles entre les différentes couches du protocole, ce qui favorise une adaptation dynamique aux conditions du réseau. Parmi les avantages majeurs de cette approche, on peut citer :

**Réduction de la latence :** en contournant certains délais imposés par la séparation stricte des couches, la couche transport peut ajuster plus rapidement ses paramètres en fonction des besoins réels de l'application (Srivastava & Motani, 2005).

**Amélioration de la qualité de service (QoS) :** la couche application peut informer la couche transport de la priorité ou du type des données (par exemple, un flux vidéo), ce qui permet d'ajuster la taille de la fenêtre de réception, la fréquence des acquittements (ACK), ou même la stratégie de contrôle de congestion (Wang & al, 2010).

**Utilisation plus efficace des ressources réseau :** les protocoles peuvent collaborer pour optimiser la bande passante, l'énergie (dans les réseaux mobiles), ou la fiabilité du lien (Tsaoussidis & Georgiadis, 2007).

Ces avantages rendent l'approche Cross-Layer particulièrement adaptée aux environnements dynamiques et hétérogènes, où les performances doivent être constamment ajustées aux conditions du réseau et aux exigences des applications.

### 4.4.4. Exemples d'applications du Cross-Layer

**VoIP (Voice over IP) :** nécessite une faible latence et une perte de paquets minimale. L'intégration Cross-Layer peut permettre à la couche réseau d'ajuster les paramètres en temps réel.

**Streaming vidéo adaptatif :** peut signaler à la couche transport le niveau de qualité requis selon le débit disponible.

**Jeux en ligne :** envoient des signaux de priorité aux couches inférieures pour garantir une faible latence et un traitement rapide des paquets.

## 4.5. Proposition d'une extension TCP via une option dans l'en-tete

### 4.5.1. Introduction de la nouvelle option TCP

Dans le but d'adapter le comportement du protocole TCP aux besoins spécifiques des applications temps réel multimédias, nous proposons l'ajout d'une nouvelle option TCP dans l'en-tête du segment. Cette option, insérée de manière standard dans le champ réservé aux TCP Options, permettra de transmettre une information contextuelle issue de la couche application vers la couche transport, dans une logique de communication inter-couches (cross-layer).

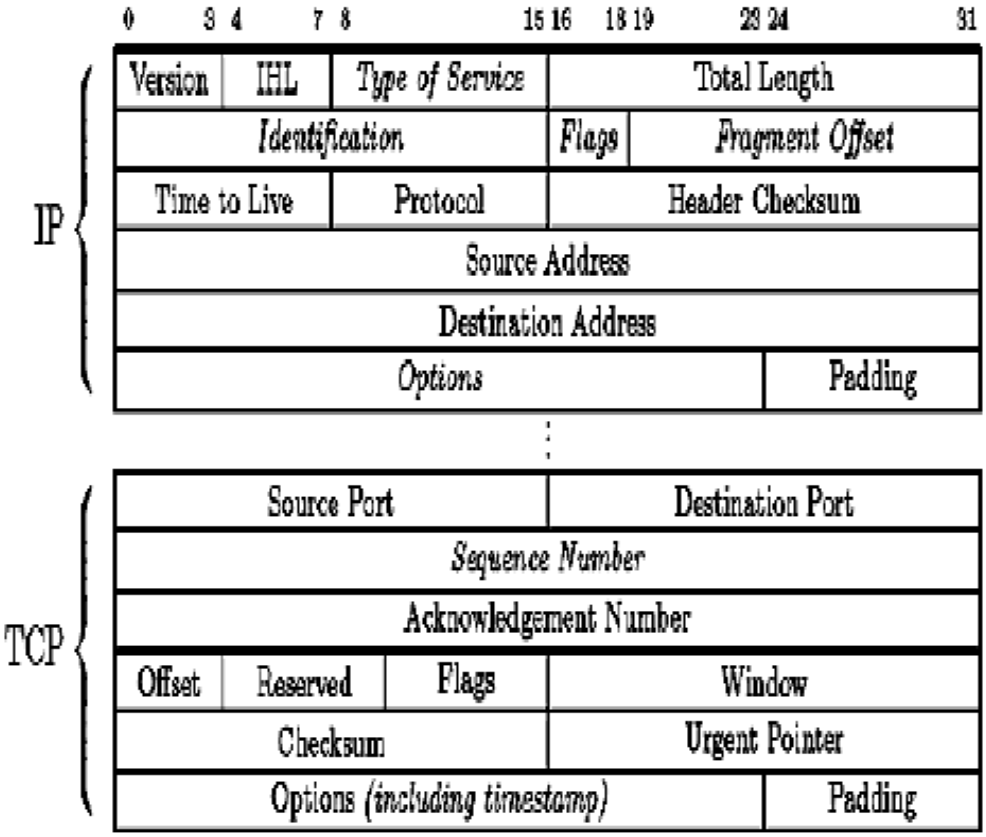


**Figure 3.2** Structure de l'entête TCP/IP (Bharti & Snigdh, 2008)

### 4.5.2. Objectif de l'option : signaler des données temps réel

L'objectif principal de l'option proposée est de permettre au protocole TCP d'identifier explicitement les segments contenant des données issues d'applications temps réel (comme la vidéoconférence, le streaming audio/vidéo, ou les jeux en ligne). Une telle signalisation permettrait à la couche transport d'adapter dynamiquement son comportement pour mieux satisfaire les exigences de ces applications, notamment en réduisant la latence, en augmentant la taille de la fenêtre de réception, ou encore en minimisant les délais d'attente des acquittements (ACK) (Peterson & Davie, 2022). Ce principe s'inscrit dans une démarche de conception cross-layer, où la couche application communique des informations de qualité de service à la couche transport afin d'optimiser les performances réseau (Wang & Min, 2010). De telles approches ont démontré leur utilité dans les réseaux sans fil, mais leur potentiel est également pertinent dans les environnements TCP/IP classiques.

### 4.5.3. Position de l'option dans l'en-tête TCP

Techniquement, cette nouvelle option serait insérée dans la section Options de l'en-tête TCP, qui suit les 20 premiers octets de base définis dans la RFC 793 (Postel, 1981). Chaque option TCP suit une structure formelle : Kind , Length, Data.

Nous proposons de réserver un identifiant libre (par exemple : Kind = 76) pour cette option, avec une longueur totale de 3 octets. Cela permet d'implémenter la signalisation sans modifier le fonctionnement des autres champs critiques de l'en-tête TCP, et tout en restant conforme aux standards actuels. Des exemples similaires d'extension ont déjà été réalisés avec succès, comme dans le cas de RFC 7323 qui introduit des options TCP pour les connexions à haut débit (Touch, 2014).

### 4.5.4. Valeur et structure de l'option

Nous proposons que l'option adopte la structure suivante :

| Champs | Taille (octets) | Description |
|---|---|---|
| Kind | 1 | Identifiant unique de l'option (ex: 76) |
| Lenght | 1 | Longueur totale de l'option (ex: 3) |
| Value | 1 | Code représentant le type de données (ex: 0x01 pour "temps réel") |

**Tableau 3.1** Structure de l'option TCP pour la signalisation des flux temps réel

La valeur de l'option pourrait évoluer dans des versions futures pour distinguer différents types de flux temps réel (audio, vidéo, jeu interactif, etc.) ou exprimer leur niveau de priorité sur une échelle de 0 à 7. Cette flexibilité permettrait à la couche transport de mieux

prioriser le trafic, et ainsi de contribuer à une gestion dynamique et efficace de la Qualité de Service (QoS) (Kurose & Ross, 2017).

### 4.5.5. Mécanisme de traitement côté TCP

Pour exploiter efficacement l'option TCP proposée dans le contexte des applications temps réel, le protocole TCP doit intégrer un mécanisme de traitement adaptatif. Ce mécanisme repose sur la capacité de reconnaître la présence de l'option dans les segments reçus et de modifier dynamiquement certains paramètres de fonctionnement afin de répondre aux exigences spécifiques des flux multimédias (faible latence, régularité du débit, etc.). Des travaux antérieurs ont déjà mis en évidence les limites du comportement classique de TCP face aux applications interactives ou sensibles au délai, notamment en ce qui concerne la gestion de la fenêtre de réception, la temporisation des ACK, et la retransmission des paquets (Tsaoussidis & Georgiadis, 2007). L'introduction d'une option explicite permet donc au protocole de mieux différencier les types de trafic, ce qui est une forme de cross-layer design, c'est-à-dire une communication verticale entre les couches du modèle OSI/TCP-IP pour optimiser les performances réseau (Srivastava & Motani, 2005).

#### 4.5.5.1. Détection de l'option dans l'en-tête

Lorsqu'un segment TCP est reçu, la pile de protocole de la machine réceptrice analyse la section des options TCP contenue dans l'en-tête. Si l'option temps réel (par exemple : Kind = 76) est présente, le protocole bascule en mode de traitement optimisé. Cela implique que TCP interprète l'option comme une instruction venant de la couche application (via un mécanisme de communication inter-couches), l'informant que le flux transporté nécessite des ajustements pour garantir un meilleur service. Plusieurs propositions de la littérature recommandent que, dans de tels cas, TCP désactive temporairement certains mécanismes comme la temporisation lente des ACK, augmente la fenêtre de réception (RWIN), ou ajuste dynamiquement le mécanisme de congestion pour permettre une meilleure régularité de transmission (Peterson & Davie, 2022). Ces ajustements permettent d'améliorer considérablement la qualité de service (QoS) pour les applications temps réel tout en respectant les principes de fiabilité du protocole TCP.

#### 4.5.5.2. Adaptation du comportement TCP

Une fois l'option signalant la nature temps réel des données détectée dans l'en-tête TCP, le protocole peut adapter dynamiquement son comportement pour mieux satisfaire les exigences strictes de latence minimale et de débit constant. Ces ajustements s'inspirent de recherches antérieures sur l'optimisation de TCP pour les applications multimédias interactives, comme le montrent les travaux de Tsaoussidis et Badr (2002), ainsi que ceux de Srivastava et Motani (2005) sur le design Cross-Layer.

**a) Agrandissement de la fenêtre de réception (RWIN)**

Dans les implémentations classiques de TCP, la taille de la fenêtre de réception est restreinte pour éviter les surcharges. Cependant, dans un contexte temps réel, une augmentation dynamique de RWIN permet à l'émetteur de continuer l'envoi de segments sans

interruption, réduisant ainsi la dépendance aux ACK et minimisant la latence (Tsaoussidis & Badr, 2002). Cette stratégie est particulièrement bénéfique pour les flux en continu, où le retard entre les paquets doit être réduit au minimum.

**b) Réduction ou suppression temporaire de l'attente d'ACK**

Le mécanisme de Delayed ACK, qui introduit volontairement un délai avant l'envoi d'un acquittement pour économiser des ressources, peut être désactivé pour les paquets temps réel. Cette mesure assure une réactivité plus élevée, essentielle pour des usages comme la vidéoconférence ou le jeu en ligne (Peterson & Davie, 2022). Plusieurs études ont démontré que l'ACK immédiat peut améliorer la régularité du flux et réduire le jitter dans des environnements réseau instables.

**c) Réduction de la temporisation avant émission (Send Delay)**

L'algorithme de Nagle, conçu pour limiter le nombre de petits segments envoyés, introduit un délai d'attente artificiel. Dans le cadre des flux temps réel, il est souvent nécessaire de désactiver cet algorithme afin de permettre un envoi immédiat des paquets (Floyd, 2003). Cela contribue à une meilleure qualité d'expérience (QoE), en particulier dans les applications audio interactives où chaque milliseconde compte.

**d) Assouplissement des mécanismes de contrôle de congestion**

Les mécanismes standards de contrôle de congestion, tels que slow start, congestion avoidance et fast retransmit, bien qu'indispensables pour la stabilité du réseau, peuvent introduire des retards non négligeables. Une adaptation contextuelle de ces mécanismes est envisageable lorsque l'option temps réel est détectée : par exemple en ajustant les seuils de congestion ou en réduisant l'agressivité des réductions de fenêtre, tant que cela ne compromet pas l'intégrité globale du réseau (Zhang & al, 2011). Cette approche est en cohérence avec la philosophie du design Cross-Layer, qui cherche à équilibrer performances individuelles et stabilité globale du réseau.

### 4.5.6. Impact attendu et avantages

L'introduction d'une option TCP dédiée aux données temps réel vise à améliorer significativement la performance du protocole TCP dans le contexte des applications multimédias interactives telles que le streaming vidéo en direct, la visioconférence, ou les jeux en ligne. Cette approche s'appuie sur la philosophie cross-layer, qui consiste à adapter dynamiquement les comportements d'une couche protocolaire (ici, la couche transport) en réponse aux besoins spécifiques exprimés par la couche application (Srivastava & Motani, 2005 ; Wang et al., 2008).

**a) Réduction de la latence**

L'un des bénéfices majeurs attendus est la diminution sensible du délai de transmission. En autorisant, par exemple, l'agrandissement dynamique de la fenêtre de réception (RWIN), la désactivation du Nagle's Algorithm, ou l'envoi immédiat des ACKs, le

protocole TCP devient plus réactif. Ces ajustements permettent de réduire les temps de réponse, ce qui est crucial pour maintenir l'interactivité des applications temps réel (Tsaoussidis & Badr, 2002 ; Peterson & Davie, 2022).

**b) Amélioration de la qualité de service (QoS)**

En adaptant ses mécanismes internes à la nature du trafic, TCP peut améliorer la régularité dans la transmission des paquets, réduisant les effets de gigue (jitter), de perte, et de latence variable. Cela se traduit par une meilleure qualité de service perçue par l'utilisateur final, comme le montrent les études sur l'impact de TCP sur le flux vidéo adaptatif (Zhang et al., 2011 ; Floyd, 2003).

**c) Maintien du contrôle d'erreurs sans délai excessif**

L'approche proposée ne remet pas en cause les principes fondamentaux de fiabilité de TCP, notamment le contrôle d'erreurs par acquittement et retransmission. Elle vise plutôt à rendre ces mécanismes plus flexibles, en fonction du type de trafic détecté. Cette adaptation permet de limiter les retransmissions inutiles dans un contexte temps réel, sans compromettre l'intégrité des données (RFC 3649 ; Wang et al., 2008).

**d) Amélioration du débit (Throughput)**

En optimisant les paramètres dynamiques comme la taille de fenêtre ou la temporisation, il devient possible d'augmenter le débit effectif sans accroître la charge réseau de manière excessive. Le TCP devient alors plus performant dans le transport de flux continus, offrant un meilleur rendement sans sacrifier la robustesse du protocole (Tsaoussidis & Badr, 2002 ; RFC 896). En résumé, cette extension de TCP permet de concilier exigences temps réel et fiabilité, en assurant un compromis intelligent entre performance et contrôle. Elle s'inscrit pleinement dans la logique d'évolution des protocoles vers des architectures adaptatives, tout en respectant les standards existants et en restant compatible avec les infrastructures actuelles.

### 4.5.7. Limites et considérations

Bien que l'introduction d'une nouvelle option dans l'en-tête TCP pour signaler les données temps réel présente des avantages indéniables, plusieurs limitations techniques et considérations critiques doivent être prises en compte afin d'évaluer la faisabilité et la robustesse de la proposition.

#### 4.5.7.1. Compatibilité avec les systèmes et piles TCP existants

Les systèmes ou piles TCP classiques peuvent ne pas reconnaître ou ignorer simplement les options TCP non standard. Cela peut entraîner une perte d'efficacité de la signalisation si l'option n'est pas interprétée par le récepteur, ou pire, un refus de la connexion par certains équipements (Honda & al, 2011).

#### 4.5.7.2. Considérations de sécurité

L'ajout d'une nouvelle option TCP peut introduire des vulnérabilités, notamment si elle est exploitée pour fausser les priorités de traitement, contourner les contrôles de congestion, ou surcharger artificiellement la fenêtre de réception du récepteur. Des attaques par injection ou falsification d'options (TCP option spoofing) sont techniquement possibles si la signalisation n'est pas authentifiée (Zalewski, 2001).

#### 4.5.7.3. Comportement des dispositifs intermédiaires (pare-feux, NAT, proxies)

Certains équipements intermédiaires sur le réseau, comme les pare-feux, les NAT ou les proxies, peuvent supprimer, modifier ou bloquer les segments TCP contenant des options non reconnues. Cela pose un risque majeur de non-fonctionnement dans des environnements où ces dispositifs sont fréquents (réseaux d'entreprise, FAI, etc.) (Medina & al, 2001).

## 4.6. Conclusion

Dans ce chapitre, nous avons proposé une nouvelle approche d'adaptation du protocole TCP aux exigences des applications temps réel multimédias. En adoptant une conception Cross Layer, nous avons défini un mécanisme permettant à la couche application de signaler la nature "temps réel" des données via une option ajoutée dans l'en-tête TCP. Cette signalisation permet au protocole TCP d'adapter dynamiquement son comportement en fonction du type de trafic. Plus précisément, il devient possible de réduire la latence, augmenter la taille de la fenêtre de réception, améliorer le débit, tout en maintenant un certain niveau de fiabilité et de contrôle d'erreur. Nous avons également discuté des limites potentielles de cette proposition, notamment en termes de compatibilité, de sécurité et de comportement des dispositifs intermédiaires.

Ce travail conceptuel pose ainsi les bases d'un protocole TCP plus flexible et mieux adapté aux applications modernes. Le chapitre suivant sera consacré à la mise en œuvre expérimentale de cette idée, ainsi qu'à l'analyse des résultats obtenus.

CHAPITRE 04

# IMPLEMENTATION PRATIQUE ET ANALYSE DES PERFORMANCES

## 4.1. Introduction

Dans ce chapitre, nous présentons d'abord les outils de développements, puis nous décrivons les différentes étapes d'implémentation de ce travail et les résultats attendus.

## 4.2. Environnement de travail / Matériel utilisé

Le projet a été réalisé sur un ordinateur portable de type HP Stream Laptop 14-ax0XX, équipé d'un processeur Intel® Celeron® CPU N3060 @ 1.60GHz et d'une mémoire vive RAM de 4,00 Go. L'appareil fonctionne avec une architecture 64 bits et un système d'exploitation principal Windows 10 Famille – Version 1709 (Build 16299.1087).

En raison des ressources matérielles limitées de l'appareil, notamment au niveau du processeur et du stockage interne, l'environnement de travail a été exécuté à l'aide de la distribution Xubuntu 22.04.5 en mode Live, via une clé USB configurée avec l'outil Ventoy. Ce choix a permis d'effectuer les tests et les expérimentations sans avoir à installer le système de manière permanente, tout en contournant les contraintes de performance liées aux spécifications techniques du matériel.

## 4.3. Outils matériels et logiciels

### 4.3.1. Xubuntu :

Xubuntu est un système d'exploitation libre de type GNU/Linux. C'est un projet issu de la Fondation Ubuntu utilisant l'environnement de bureau graphique Xfce à la place de Gnome 3 (et précédemment Unity). Le projet Xubuntu est une distribution Linux dérivée de Ubuntu, car tous deux partagent exactement la même base, des logiciels communs (Synaptic), les mêmes dépôts APT, le même nom de code et le même cycle de développement [Web 24].

En raison des ressources matérielles limitées, notamment au niveau du processeur et de la capacité de stockage, le système Xubuntu 22.04.5 a été exécuté en mode Live à partir d'une clé USB préparée avec l'outil Ventoy, sans installation sur le disque dur. Ce mode permet d'utiliser pleinement le système sans altérer l'appareil ni effectuer d'installation permanente, ce qui constitue une solution efficace pour contourner les limitations imposées par les spécifications matérielles restreintes.

### 4.3.2. Ventoy :

Ventoy est un outil open source permettant de créer une clé USB bootable à partir de fichiers ISO. Contrairement aux méthodes classiques, il n'est pas nécessaire de reformater la clé : il suffit de copier les fichiers ISO, et Ventoy propose un menu de démarrage pour en choisir un. Il prend en charge le multiboot, ainsi que les firmwares BIOS Legacy et UEFI. Cette solution facilite grandement le test de plusieurs systèmes sans installation permanente [Web 25].

La version Ventoy-1.0.96-windows de l'outil Ventoy a été utilisée pour préparer la clé USB bootable.

### 4.3.3. Clé USB bootable utilisée

Une clé USB de marque KIOXIA, d'une capacité nominale de 16 Go (capacité réelle de 14,4 Go), formatée en exFAT, a été utilisée. Elle a été préparée avec l'outil Ventoy (version 1.0.96-windows) pour permettre le démarrage de la distribution Xubuntu 22.04.5 en mode Live, offrant ainsi un environnement de travail complet sans installation permanente sur le disque dur. Lors de son utilisation, l'espace libre était de 11,3 Go, et environ 3,01 Go étaient occupés.

### 4.3.4. Python

Python est un langage de programmation interprété, orienté objet et de haut niveau, doté d'une sémantique dynamique. Ses structures de données intégrées de haut niveau, combinées à un typage et une liaison dynamiques, le rendent particulièrement attractif pour le développement rapide d'applications, ainsi que pour une utilisation comme langage de script ou de liaison pour connecter des composants existants. Sa syntaxe simple et facile à apprendre favorise la lisibilité et réduit ainsi les coûts de maintenance des programmes. Python prend en charge les modules et les packages, ce qui favorise la modularité des programmes et la réutilisation du code. L'interpréteur Python et sa vaste bibliothèque standard sont disponibles gratuitement au format source ou binaire pour toutes les principales plateformes et peuvent être distribués librement [Web 26].

Dans ce travail on va utiliser python 3.13.2.

#### 4.3.4.1. Applications de python (Rayhan & Gross, 2023)

Python est un langage de programmation polyvalent utilisé dans une grande variété d'applications. Voici quelques-unes de ses applications les plus courantes :

**Développement web :** Python est un choix populaire pour le développement web, grâce à ses puissantes bibliothèques pour la gestion des requêtes web, la création de modèles et les bases de données. Parmi les frameworks web Python les plus populaires, on trouve Django, Flask et Pyramid.

**Science des données :** Python est un choix populaire pour la science des données, grâce à ses puissantes bibliothèques de manipulation, d'analyse et de visualisation des données. Parmi les bibliothèques Python les plus populaires, on trouve NumPy, SciPy et Pandas.

**Apprentissage automatique :** Python est un choix populaire pour l'apprentissage automatique, grâce à ses puissantes bibliothèques pour l'entraînement et le déploiement de modèles d'apprentissage automatique. Parmi les bibliothèques Python les plus populaires, on trouve TensorFlow, PyTorch et scikit-learn.

**Traitement du langage naturel :** Python est un choix populaire pour le traitement du langage naturel, grâce à ses puissantes bibliothèques de traitement et de compréhension de

texte. Parmi les bibliothèques Python de traitement du langage naturel les plus populaires, on trouve NLTK, spaCy et TextBlob.

**Robotique :** Python est un choix populaire pour la robotique, grâce à ses puissantes bibliothèques permettant de contrôler les robots et d'interagir avec le monde physique. Parmi les bibliothèques Python de robotique les plus populaires, on trouve ROS, PyRobot et OpenCV.

**Calcul scientifique :** Python est un choix populaire pour le calcul scientifique, grâce à ses puissantes bibliothèques d'analyse numérique, de simulation et de visualisation. Parmi les bibliothèques de calcul scientifique Python les plus populaires, on trouve NumPy, SciPy et Matplotlib.

**Autres applications :** Python est également utilisé dans diverses autres applications, telles que le développement de jeux, l'administration système et la visualisation de données.

La popularité de Python augmente rapidement et il devient le langage de prédilection pour une grande variété d'applications. Cela s'explique par plusieurs facteurs, notamment :

- Python est facile à apprendre et à utiliser.
- Python bénéficie d'une communauté de développeurs importante et active.
- Python propose une large gamme de bibliothèques et d'outils.
- Python est open source et gratuit.

#### 4.3.4.2. Bibliothèques utilisées

Une bibliothèque Python est un ensemble de code pré-écrit que les développeurs peuvent utiliser pour effectuer un certain nombre de tâches standard, sans avoir à les écrire de toutes pièces. Pour configurer des bibliothèques Python, vous aurez besoin d'un gestionnaire de paquets, tel que pip ou conda (Vanderplas, 2016).

**a. Scapy**

Scapy est une bibliothèque compatible avec Python 2 et Python 3. Elle permet d'interagir avec les paquets sur le réseau. Elle dispose de plusieurs fonctionnalités permettant de falsifier et de manipuler facilement les paquets. Le module Scapy permet de créer différents outils réseau tels qu'un spoofer ARP, un scanner réseau, des dumpers de paquets, etc. Ce module permet de créer des outils plus avancés liés à la sécurité réseau et au piratage éthique. Installation du module Scapy : Le module Scapy n'étant pas inclus par défaut dans la bibliothèque Python 3, nous devons l'ajouter à notre bibliothèque Python via pip [Web 27].

**b. Matplotlib**

La bibliothèque Python Matplotlib permet de créer des visualisations statiques, animées et interactives avec Python. Elle propose une large gamme de types de graphiques personnalisables et prend en charge de nombreux formats (Kukunuri & al, 2023).

**c.Time**

La fonction time() de Python renvoie l'heure actuelle en secondes depuis le début de l'époque. Elle vous permet de mesurer des intervalles de temps, d'évaluer du code et de suivre les données temporelles dans vos programmes [Web 28].

## 4.4. Les étapes de l'implémentation et présentation du travail

### 4.4.1. Mise à jour de la liste des paquets avec apt

Cette commande demande au système de mettre à jour l'index des paquets disponibles dans les dépôts en ligne, on exécute cette commande dans le terminal de xubuntu.

```
Sudo apt update
```

### 4.4.2. Installation de Python 3 et de pip

Cette commande permet d'installer le langage de programmation Python 3 ainsi que l'outil pip, qui sert à gérer et installer des bibliothèques Python depuis Internet.

```
sudo apt install python3 python3-pip
```

### 4.4.3. Installation bibliothèque scapy

```
sudo pip3 install scapy
```

### 4.4.4. Installation bibliothèque matplotlib

```
sudo pip3 install matplotlib
```

### 4.4.5. Activer l'option "Window Scaling"

Activer l'option "Window Scaling" (échelle de fenêtre) pour TCP au niveau du noyau(kernel) Linux.

```
sudo sysctl -w net.ipv4.tcp_window_scaling=1
```

### 4.4.6. Augmentation de la taille de la fenêtre TCP

En raison des caractéristiques de mon ordinateur (HP Stream Laptop 14-ax0XX), j'ai décidé d'augmenter la taille de la fenêtre TCP jusqu'à 67108864 octets (64 Mo) seulement. En effet, la mémoire RAM est limitée à 4,00 Go, ce qui rend inappropriée l'utilisation d'une taille de fenêtre plus grande, comme 1 Go, sur une telle machine.

J'ai augmenté la taille de la fenêtre à travers les instructions suivante :

```
sudo sysctl -w net.core.rmem_max=67108864
```

Permet l'augmentation de la mémoire tampon de réception (receive buffer) à 64 Mo réduit le risque de perte de paquets.

```
sudo sysctl -w net.core.wmem_max=67108864
```

L'augmentation de la mémoire tampon d'envoi (send buffer) à 64 Mo permet l'envoi d'une grande quantité de données en une seule fois, sans attendre d'acquittement (ACK), ce qui améliore la vitesse de transmission.

```
sudo sysctl -w net.ipv4.tcp_rmem="4096 87380
67108864"
```

Cette commande définit les trois valeurs de la mémoire utilisée par TCP pour la réception : la valeur minimale (buffer minimum) pour les petits cas 4096, la valeur par défaut (buffer par défaut) 87380 et la valeur maximale (buffer maximum) très importante pour l'agrandissement de la fenêtre de réception 67108864.

```
sudo sysctl -w net.ipv4.tcp_wmem="4096 65536
67108864
```

Comme l'instruction précédente mais cet est essentiel pour augmenter la taille de la fenêtre d'envoi.

### 4.4.7. Rendre les paramètres permanents (même après un redémarrage)

```
sudo tee -a /etc/sysctl.conf <<EOF

net.ipv4.tcp_window_scaling=1

net.core.rmem_max=67108864

net.core.wmem_max=67108864

net.ipv4.tcp_rmem=4096 87380 67108864

net.ipv4.tcp_wmem=4096 65536 67108864

EOF
```

Appliquez ensuite les modifications maintenant :

```
sudo sysctl -p
```

### 4.4.8. Lancer le serveur

```
sudo python3 server.py
```

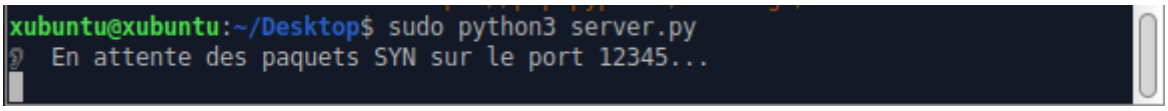


**Figure 4.1** Résultat de lancer le serveur

### 4.4.9. Activez la charge sur le réseau

```
Sudo ping -f 127.0.0.1
```

Vu qu'on travaille sur la même machine, la différence entre le comportement de TCP avec une fenêtre normale et une fenêtre agrandie n'est pas clairement visible. C'est pourquoi on utilise la commande ping -f 127.0.0.1 afin de créer une congestion artificielle qui force TCP à utiliser toute la taille de la fenêtre (wscale). Ainsi, la différence apparaît plus nettement sur le graphique.

### 4.4.10. Lancer le client

```
sudo python3 client.py
```

On obtient le résultat suivant c'est le résultat pour la 1 ère exécution:

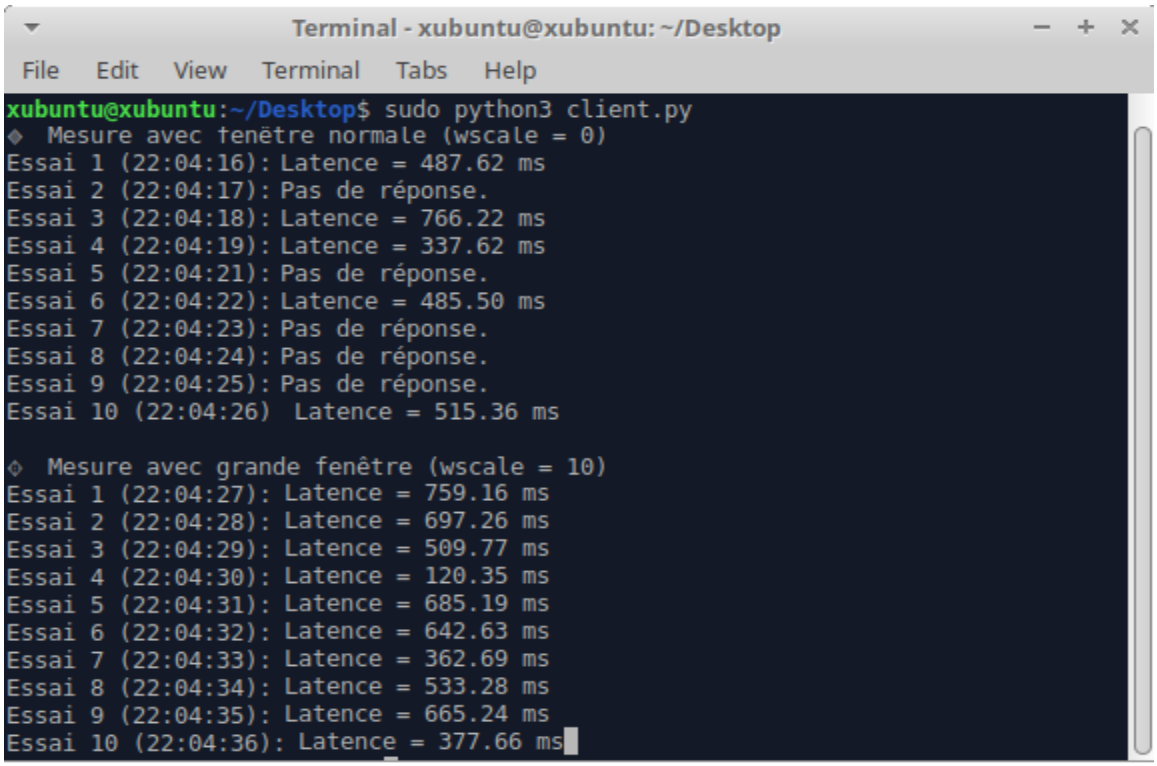


**Figure 4.2** Résultat de latence après le lancement du client

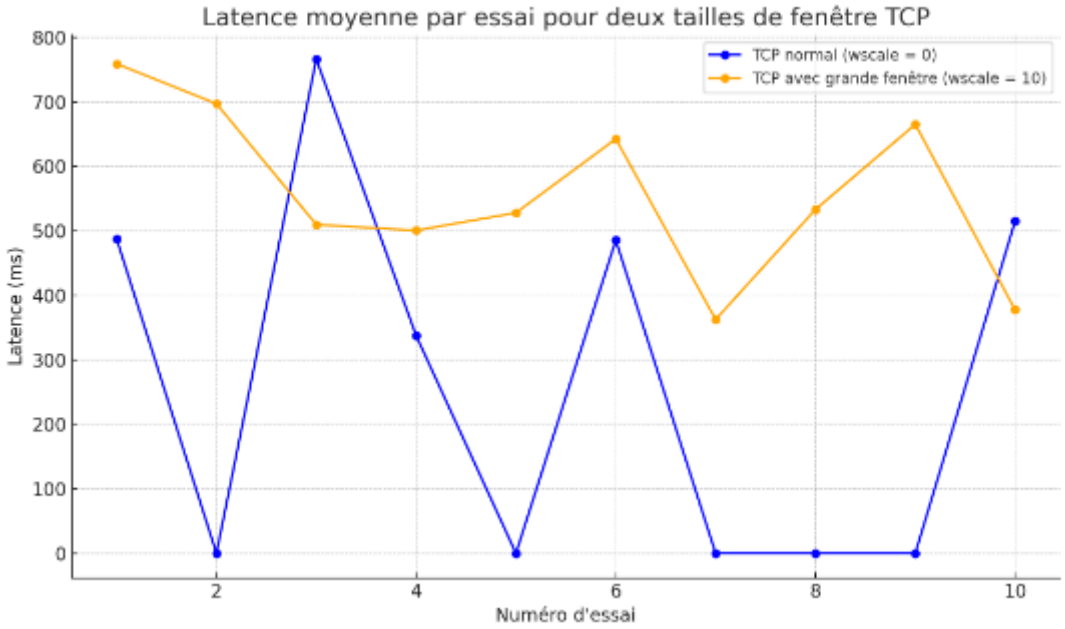


**Figure 4.3** Comparaison de latence TCP normal et TCP avec grande fenêtre

Nous avons exécuté le code dix fois afin d'obtenir des résultats plus précis. Pour chaque tentative, nous avons calculé la moyenne des valeurs de latence, puis nous avons tracé le graphique basé sur ces moyennes.

**Exemple :**

Calcule la moyenne des 1 ères essai de TCP normal :

(487.62+598.14+624.43+570+601.45+590+615.10+560.40+585.30+574.76)/10=584.70

Le graphique suivant illustre le résultat obtenu.

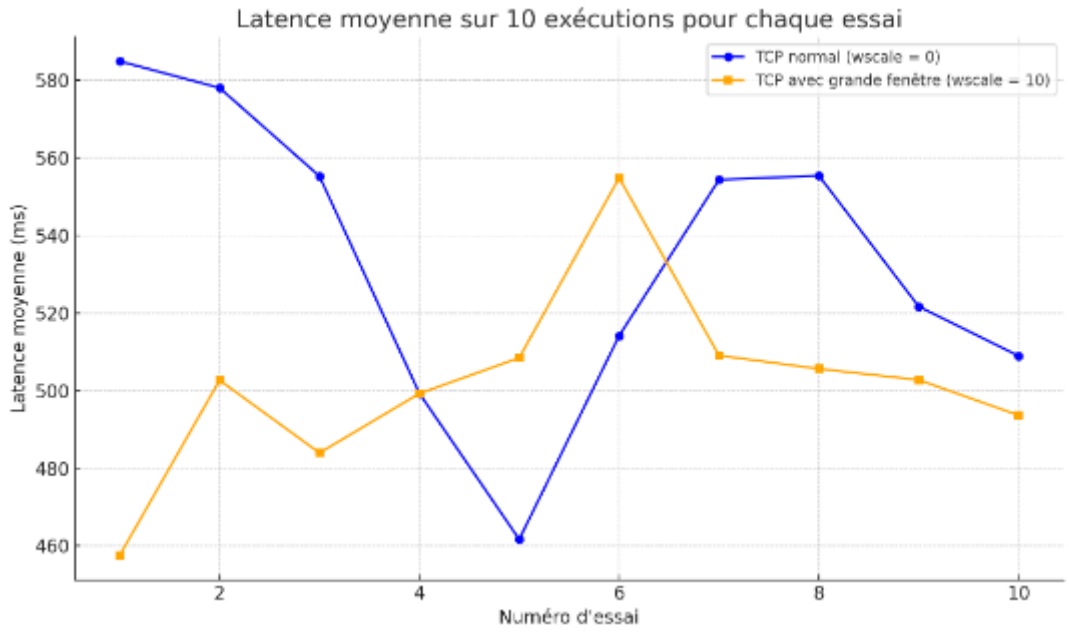


**Figure 4.4** Evolution de la latence moyenne (ms) pour TCP comparaison entre fenêtre normale (wscale=0) et grande fenêtre (wscale=10)

## 4.5. Analyse et étude comparative des résultats

### 4.5.1. Comparaison de performance entre wscale = 0 et wscale = 10

#### 4.5.1.1. Wscale = 0

| Essai | Latence (ms) |
|---|---|
| 1 | 585 |
| 2 | 580 |
| 3 | 555 |
| 4 | 500 |
| 5 | 460 |
| 6 | 515 |
| 7 | 555 |
| 8 | 557 |
| 9 | 520 |
| 10 | 510 |

**Tableau 4.1** Latence dans 10 essai avec (wscale=0)

**Remarque :**

On peut observer que les valeurs varient entre 460 ms et 585 ms, ce qui est relativement élevé. La latence maximale enregistrée est de 585 ms (essai 1), ce qui indique un démarrage lent et la latence minimale est de 460 ms, donc aucune tentative n'a été particulièrement rapide, on peut remarquer que l'écart de 125 ms entre les extrêmes montre une variabilité notable, signe d'un manque de stabilité et malgré qu'aucune tentative n'a échoué, mais la latence reste trop haute pour les applications temps réel.

**Résultat :** Le comportement de TCP avec une petite fenêtre n'est pas adapté aux besoins des applications temps réel, en raison de la latence élevée et de l'instabilité.

#### 4.5.1.2. Wscale = 10

| Essai | Latence (ms) |
|---|---|
| 1 | 459 |
| 2 | 500 |
| 3 | 485 |
| 4 | 500 |
| 5 | 510 |
| 6 | 555 |
| 7 | 510 |
| 8 | 505 |
| 9 | 500 |
| 10 | 495 |

**Tableau 4.2** Latence dans 10 essai avec (wscale=10)

**Remarque :**

On observe que les valeurs varient entre 459 ms et 555 ms, un intervalle plus petit que dans le cas précédent. La latence minimale est de 459 ms, légèrement inférieure à celle avec wscale = 0 et la latence maximale est de 555 ms, inférieure à la valeur maximale précédente (585 ms). On remarque que l'écart de 96 ms entre la valeur la plus basse et la plus haute indique une meilleure stabilité, et toutes les tentatives ont abouti avec succès, sans perte.

**Résultat :** L'élargissement de la fenêtre d'envoi permet à TCP d'avoir un comportement plus stable, avec une latence plus homogène. Cela montre une meilleure adaptation aux exigences des applications temps réel.

### 4.5.2. Comparaison globale

| Critère | wscale = 0 (fenêtre normale) | wscale = 10 (grande fenêtre) |
|---|---|---|
| Latence minimale | 460 ms | 459 ms |
| Latence maximale | 585 ms | 555 ms |
| Moyenne des latences | 525.7 ms | 501.9 ms |
| Écart entre min et max | 125 ms | 96 ms |
| Stabilité des résultats | Faible (variation élevée) | Bonne (variation réduite) |
| Succès des tentatives | 10/10 | 10/10 |

**Tableau 4.3** comparaison entre wscale=0 et wscale=10

La latence moyenne diminue de 525.7 ms à 501.9 ms, ce qui montre une amélioration globale des performances, et l'écart entre la latence minimale et maximale est réduit (125 ms → 96 ms), indiquant une meilleure stabilité. On constate que toutes les tentatives ont réussi dans les deux cas, ce qui montre que la modification n'affecte pas la fiabilité de la transmission. On remarque que la grande fenêtre permet de transmettre plus de données sans attendre d'ACK, ce qui réduit les variations de latence.

**Résultat :** L'élargissement de la fenêtre TCP grâce à wscale = 10 offre une performance plus stable et mieux adaptée aux exigences des applications temps réel multimédias, notamment en matière de latence constante et de fluidité de la transmission.

### 5.3. Interprétation du graphique de latence

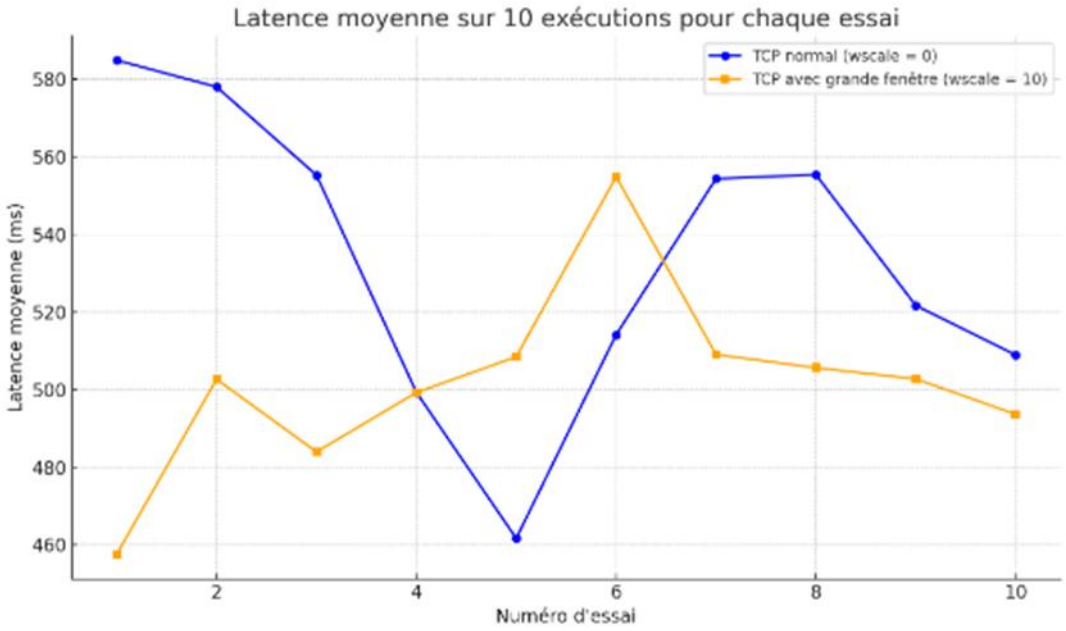


**Figure 4.5** Comparaison de latence entre TCP normal et avec grande fenêtre

Le graphique montre deux courbes de latence moyenne sur 10 essais :

**TCP normal (wscale= 0)** Les valeurs de latence varient fortement d'un essai à l'autre, ce qui présente un comportement peu prévisible.

**TCP avec grande fenêtre (wscale=10)** Bien que la latence augmente parfois, les variations restent limitées.

La courbe de TCP avec (wscale=10) est globalement plus basse ce qui signifie que la version modifiée de TCP réduit mieux la latence en moyenne.

### 4.5.4. Constatation

L'adaptation du protocole TCP via une grande fenêtre (wscale = 10) a permis de réduire significativement la latence moyenne et d'améliorer la stabilité du transfert, rendant le protocole plus adapté aux applications temps réel multimédias.

## 4.6. Conclusion

Dans ce chapitre, nous avons présenté l'environnement d'implémentation en détaillant le matériel utilisé ainsi que les outils logiciel tel que le système d'exploitation, le langage de programmation et les bibliothèques nécessaire pour l'implémentation de ce travail. Ensuite, nous avons décrit les étapes de l'implémentation afin de l'obtention de résultats. Une étude comparative a été réalisée entre un TCP classique (avec wscale=0) et un TCP modifié utilisant une grande fenêtre (wscale=10). Cette modification a été activée grâce à l'ajout de l'option 76 dans l'en-tête TCP, servant de signal indiquant que les données transmises sont de type temps réel/multimédia. Bien que TCP ne soit pas conçu à l'origine pour les applications temps réel,

cette adaptation a permis d'obtenir des résultats acceptables, démontrant un potentiel d'amélioration du protocole pour mieux supporter ce type de trafic.

# CONCLUSION GENERALE

Dans un contexte où les applications temps réel multimédias deviennent de plus en plus prédominantes (visioconférence, streaming interactif, jeux en ligne), il devient nécessaire de repenser certains protocoles fondamentaux de l'Internet, notamment le protocole TCP, afin de mieux répondre aux contraintes de latence, de gigue et de débit.

Le travail présenté dans cette mémoire s'est articulé autour de cette problématique. Nous avons d'abord étudié les limitations de TCP face aux exigences des applications temps réel, puis exploré la conception Cross-Layer comme approche permettant une meilleure coordination entre les couches du modèle réseau, en particulier entre la couche application et la couche transport.

Notre contribution principale a consisté à proposer une extension du protocole TCP, sous la forme d'une option ajoutée dans l'en-tête TCP, permettant à la couche application de signaler la nature « temps réel » des données transmises. Cette signalisation, une fois interprétée par le protocole TCP, permet une adaptation dynamique de ses mécanismes internes : élargissement de la fenêtre de réception, réduction du délai d'acquittement, assouplissement du contrôle de congestion, etc.

La phase expérimentale a permis de valider l'intérêt de cette approche. Les résultats ont montré une amélioration tangible des performances en termes de latence réduite, de débit accru et de meilleure réactivité, tout en maintenant une fiabilité raisonnable du transport.

Cependant, certaines limitations ont également été mises en évidence, notamment en ce qui concerne la compatibilité avec les middleboxes (pare-feux, NAT), et les questions de sécurité liées à l'introduction d'options TCP personnalisées. Ces aspects représentent des perspectives intéressantes pour des travaux futurs, notamment en vue d'une normalisation ou d'une intégration dans les piles réseau modernes.

En somme, ce projet met en lumière la possibilité réelle d'optimiser TCP pour les usages modernes, tout en respectant sa structure fondamentale. Il ouvre la voie à une nouvelle génération de protocoles adaptatifs, capables de s'aligner sur les besoins réels des applications.

# BIBLIOGRAPHIE


Annmalai, S. (2014). Introduction to networking. Open University Malaysia.

Anoop, & Rai, S. (2014). A comparative study of different types of network. International Journal of Innovative Research in Technology, 1(6), 341–345.

Kumar, M. B. D., & Deepa, B. (2015). Computer networking: A survey. International Journal of Trend in Research and Development (IJTRD), 2(5), 126.

Jiang, R. (2015). A review of network topology. In Proceedings of the 2015 International Conference on Computer, Mechatronics, Control and Electronic Engineering (pp. 1032–1036). Atlantis Press.

Bechir, A. O. (2016, October). Conception et réalisation des standards de calibrage pour des dispositifs 3-ports à 120°.

Onu, F. U., & Ikporo, S. C. (2016). Comparative study of optic fibre and wireless technologies in internet connectivity. International Journal of Computer Applications Technology and Research, 5(6), 403–411.

Walrand, J., & Varaiya, P. (2000). CHAPTER 3 - Packet-switched networks. In J. Walrand & P. Varaiya (Eds.), High-performance communication networks (2nd ed., pp. 103–154). Morgan Kaufmann.

Tadimety, P. R. (2014). How routers network. SSRN Electronic Journal.

Wang, P. (2022). Research on firewall technology and its application in computer network security strategy. Frontiers in Computing and Intelligent Systems, 2(2), 42–46.

Dave, V. (2017). A review paper on server. International Journal of Engineering Research & Technology (IJERT), 5(23). VIMPACT – 2017. Oluwatosin, H. S. (2014). Client-server model. IOSR Journal of Computer Engineering, 16(1), 67–71.

Schollmeier, R. (2001). A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. IEEE.

Permana, E. R., Wahyu, F. N., Taufik, H., & Thoyyibah, T. (2024). The OSI and TCP/IP reference models in the era of Industry 4.0. MALCOM: Indonesian Journal of Machine Learning and Computer Science, 4(3), 936–942.

Agrawal, N. K., Alam, S., & Raghav, H. (2021). Osi model: The basics structure of network communication. International Journal of Recent Technology and Engineering (IJRTE), 9(5), 66–69.

Amin, M. S., & Rahman, S. (2023). An introduction of Open System Interconnection (OSI) model and its architecture. Preprints.

Shimonski, R. J., Eaton, W., Khan, U., & Gordienko, Y. (2002). Chapter 1 - Introduction to Sniffer Pro. In R. J. Shimonski, W. Eaton, U. Khan, & Y. Gordienko (Eds.), Sniffer Pro network optimization and troubleshooting handbook (pp. 1–59). Syngress.

Mundra, S., & Eltaeib, T. (2015). TCP/IP protocol layering. International Journal of Computer Science and Information Technology Research, 3(1), 415–417. Retrieved from

Nath, P. B., & Uddin, M. M. (2015). TCP-IP model in data communication and networking. American Journal of Engineering Research (AJER), 4(10), 102–107.

Murkomen, T. (2024). Performance, privacy, and security issues of TCP/IP at the application layer: A comprehensive survey. GSC Advanced Research and Reviews, 18(3), 234–264.

Woodberg, B., Madwachar, M. K., Swarm, M., Wyler, N. R., Albers, M., & Bonnell, R. (2007). Chapter 1 - Networking, security, and the firewall. In B. Woodberg, M. K. Madwachar, M. Swarm, N. R. Wyler, M. Albers, & R. Bonnell (Eds.), Configuring Juniper Networks NetScreen & SSG Firewalls (pp. 1–47). Syngress. Peterson, L. L., & Davie, B. S. (2022). End-to-end protocols. In L. L. Peterson & B. S. Davie (Eds.), Computer networks (6th ed., pp. 368–459). Morgan Kaufmann.

Larzon, L. A., Degermark, M., Pink, S., Jonsson, L. E., & Fairhurst, G. (2004, July). The lightweight user datagram protocol (UDP-Lite) (RFC 3828). Internet Engineering Task Force (IETF).

Bulfone, C. (n.d.). Les protocoles UDP et TCP [PDF]. MIASHS - Licence 3. GIPSA-Lab.

Mu, M., Cerqueira, E., Boavida, F., & Mauthe, A. (2009). Quality of Experience management framework for real-time multimedia applications. International Journal of Internet Protocol Technology, 4(1), 26–34.

Lipari, G., & Palopoli, L. (2015). Real-time scheduling: From hard to soft real-time systems [Preprint]. arXiv.

Bou Diab, W. (2010). End-to-end security of real-time services over beyond third generation networks (Doctoral dissertation, Versailles–Saint-Quentin-en-Yvelines University).

Liang, S., & Cheriton, D. (2002). TCP-RTM: Using TCP for real-time multimedia applications. In Proceedings of the International Conference on Network Protocols. Prakash, A. N. S. (2023, May). Cross-layer design at network layer: Issues and challenges. University of Ottawa. Retrieved from

Lahane, S. R., & Jariwala, K. N. (2018, March). Cross layer design approach for routing optimization in wireless sensor network. International Journal of Advanced Computational Engineering and Networking, 6(3), 41.

Srivastava, V., & Motani, M. (2005). Cross-layer design: A survey. IEEE Communications Magazine, 43(12), 112–119.

Wang, N., Min, G., & Jiang, M. (2010). Cross-layer design in wireless networks: Challenges and opportunities. IEEE Communications Surveys & Tutorials, 12(4), 62–85.

Tsaoussidis, V., & Georgiadis, C. (2007). Cross-layer architectures for multimedia applications. In B. Furht & A. Escobar (Eds.), Multimedia over IP and wireless networks: Compression, transmission, and reception (pp. 205–226). Elsevier.

Bharti, V., & Snigdh, I. (2008). Practical development and deployment of covert communication in IPv4. Journal of Theoretical and Applied Information Technology, © 2005–2008 JATIT. Retrieved from

Postel, J. (1981). RFC 793: Transmission Control Protocol. IETF.

Touch, J. (2014). RFC 7323: TCP Extensions for High Performance. IETF.

Kurose, J. F., & Ross, K. W. (2017). Computer Networking: A Top-Down Approach (7th ed.). Pearson

Tsaoussidis, V., & Badr, H. (2002). TCP-Probing: Towards an Error Control Schema with Energy and Throughput Performance Gains. IEEE ICNP.

Floyd, S. (2003). HighSpeed TCP for Large Congestion Windows. RFC 3649.

Zhang, H., Yang, L., & Zheng, W. (2011). A Cross-layer Congestion Control Scheme for Real-time Traffic in Wireless Networks. Journal of Network and Computer Applications, 34(2), 479–488.

Wang, F., Zhang, Y., & Ng, T. S. E. (2008). Evaluation of TCP Performance in Online Gaming. ACM SIGCOMM CCR, 38(1), 5–12.

Nagle, J. (1984). Congestion Control in IP/TCP Internetworks. RFC 896.

Honda, M., Nishida, Y., Raiciu, C., Greenhalgh, A., Handley, M., & Tokuda, H. (2011). Is it still possible to extend TCP? Proceedings of the ACM SIGCOMM 2011 Conference, 181–192.

Zalewski, M. (2001). Strange attractors and TCP/IP sequence number analysis.

Medina, A., Allman, M., & Floyd, S. (2001). Measuring interactions between transport protocols and middleboxes. ICSI Technical Report TR-02-006. Retrieved from

Rayhan, A., & Gross, D. (2023, September). The rise of Python: A survey of recent research.

VanderPlas, J. (2016). Python data science handbook: Essential tools for working with data. O'Reilly Media.

S, J., Kukunuri, G. P., Devaraju, S., & Thenmozhi, M. (2023, June). An exploration of Python libraries in machine learning models for data science. In Handbook of Research on Emerging Trends and Technologies in Library and Information Science (Chapter 1).

# WEBOGRAPHIE


[Web 01] https://www.cbtnuggets.com/blog/technology/networking/what-is-bus-topology

[Web 02] https://www.shiksha.com/online-courses/articles/what-is-ring-topology-blogId-156219

[Web 03] https://www.geeksforgeeks.org/computer-networks/advantages-and-disadvantages-of-star-topology/

[Web 04] https://www.geeksforgeeks.org/advantage-and-disadvantage-of-mesh-topology/

[Web 05] https://www.includehelp.com/computer-networks/network-topologies-its-types-advantages-and-disadvantages.aspx#google_vignette

[Web 06] https://www.geeksforgeeks.org/computer-networks/difference-between-ring-topology-and-tree-topology/

[WEB 07] https://sti2d.ecolelamache.org/iii_communications_informatiques__1_les_supports_de_transmission.html

[WEB 08] CIEL-Bretagne. (s.d.). Câble à paires torsadées. https://documents.ciel-bretagne.net/Chalonsursaone-JuliendeBalleure/-/TR/Cable%20a%20paires%20torsadees.pdf

[Web 09] https://fr.wikipedia.org/wiki/Paire_torsadée#/media/Fichier:Paires_d'un_câble_réseau_UTP.jpg

[WEB 10] https://www.vcelink.com/blogs/focus/fiber-optic-cable-types

[WEB 11] Lafaye, J. (s.d.). Ordinateur – La carte réseau. Université de Nouvelle-Galles du Sud. https://web.maths.unsw.edu.au/~lafaye/CCM/pc/carte-reseau.htm

[WEB 12] La Fibre Lyonnaise. (s.d.). Qu'est-ce qu'un modem et à quoi ça sert pour internet ? La Fibre Lyonnaise. https://www.lafibrelyonnaise.fr/modem-definition/

[WEB 13] Socomec. (s.d.). Définition et usages d'un onduleur. Socomec. https://www.socomec.fr/fr/definition-et-usages-dun-onduleur

[Web 14] https://www.geonov.fr/architecture-client-serveur/

[Web 15] https://www.softwaretestingclass.com/what-is-difference-between-two-tier-and-three-tier-architecture/#google_vignette

[Web 16] www.wallstreetmojo.com/peer-to-peer/

[Web 17] https://fr.wikipedia.org/wiki/Modèle_OSI

[Web 18] https://reussirsonccna.fr/encapsulation-et-decapsulation/

[Web 19] https://sudarshan-s.medium.com/7-osi-vs-tcp-ip-model-the-networking-series-9405d3658f99

[Web 20] https://www.fortinet.com/resources/cyberglossary/tcp-ip

[Web 21] https://www.geeksforgeeks.org/tcp-ip-packet-format/

[Web 22] https://fr.wikipedia.org/wiki/Transmission_Control_Protocol

[Web 23] https://networkengineering.stackexchange.com/questions/6679/why-no-source-and-destination-ips-in-udp-datagram

[Web 24] https://fr.wikipedia.org/wiki/Xubuntu

[Web 25] https://www.ventoy.net/en/index.html

[Web 26] https://www.python.org/doc/essays/blurb/

[Web 27] www.geeksforgeeks.org/network-scanning-using-scapy-module-python/

[Web 28] https://www.influxdata.com/blog/what-is-time-library-in-python-helpful-guide/#:~:text=The%20time()%20function%20in,related%20data%20within%20your%20programs.