N° d'ordre :
N° de série :

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Superieur et de la Recherche Scientifique

UNIVERSITÉ ECHAHID HAMMA LAKHDAR D'EL OUED

FACULTÉ DESSCIENCES EXACTES

DEPARTEMENT D'INFORMATIQUE

Mémoire de fin d'étude

MASTER ACADEMIQUE

Domaine: Mathématiques et Informatique
Filière: Informatique
Spécialité: Systèmes Distribués et Intelligence Artificiel (SDIA)

Thème

Conception et réalisation d'un simulateur des couches réseau (Modèle OSI)

Présenté par: MANSOURI Zineb
TERCHA Amira

Soutenu devant le jury composé de

M.	MEFTAH Mohemed Charef Eddine	Rapporteur	Univ. d'El Oued
M.	MAA KERTIOU Ismail	Présidente	Univ. d'El Oued
M.	MAB NEGOUDI Motaze Belleh	Examinatrice	Univ. d'El Oued

Année universitaire 2015/2016

Résumé

L'objectif de notre projet est la conception et le développement d'un simulateur du modèle OSI (**O**pen **S**ystems **I**nterconnections) , son intérêt est la disposition aux enseignants et étudiants un outil pédagogique, pratique et clair, Et leur permettre d'étudier le modèle , cet simulateur facilite la compréhension des rôles des sept couches de la couche d'application (texte) à la couche physique (bit) où le message même devient un signal électrique prêt à passer à travers le support physique.

Les mots clé : Réseau , modèle OSI , couche , PDU .

Abstract

The aim of our project is the design and the development of an OSI simulator (Open Systems Interconnection), its interest is to provide an educational, practical and clear tool to teachers and students ,and allow them to study a model; this simulator makes easy the understanding of the roles of the seven layers from the application layer (text) to the physical layer (bit) where the message becomes an electrical signal to pass through the physical device.

Keywords : Network , OSI model , layer , PDU.

ملخص

الهدف من مشروعنا هو تصميم و تنفيذ محاكاة لنموذج **OSI** ، تكمن مصلحته في توفير أداة تعليمية وعملية واضحة للطلاب وتمكينهم من التعرف على هذا النموذج ، و تسهيل فهم مختلف طبقاته السبعة والعمليات التي تمر بها الرسالة انطلاقا من طبقة التطبيق (نص) إلى غاية طبقة الفيزيائية (بيت) حيث تصبح الرسالة عبارة على إشارة كهربائية جاهزة للمرور عبر الناقل الفيزيائي .

الكلمات المفتاحية : شبكة ، نموذج **OSI** ، طبقة , **PDU**.

Remerciements

Avant tous, nous remercions dieu le tout puissant de nous avoir donné le courage et la patience pour réaliser ce travail malgré toutes les difficultés rencontrées

*nous remercions infiniment tous qui nous a aidé de prés et de loin d'avoir compléter ce travail et dépasser tous les obstacle surtout notre enseignant " **MEFTAH CHAREF EDDINE**" qui n'a pas cessé de nous donner les conseils et les bonnes orientations et nous prive pas de son temps*

*Nous remercions **M. NANI Azeddine** a nous aidez*
Et en fin tous qui nous ont aidés de prés et de loin pour

La réalisation de notre mémoire.

Sommaire

Chapitre 02 :conception

Liste des figures

Liste de tableau

Introduction générale

Le premier objectif de la norme OSI a été de définir un modèle de toute architecture de réseau basé sur un découpage en sept couches, chacune de ces couches correspondant à une fonctionnalité particulière d'un réseau. Les couches 1, 2, 3 et 4 sont dites basses et les couches 5, 6 et 7 sont dites hautes.

Chaque couche est constituée d'éléments matériels et logiciels et offre un service à la couche située immédiatement au-dessus d'elle en lui épargnant les détails d'implémentation nécessaires.

Chaque couche n d'une machine gère la communication avec la couche n d'une autre machine en suivant un protocole de niveau n qui est un ensemble de règles de communication pour le service de niveau n.

L'objectif de notre travail est de réaliser un simulateur graphique de modèle OSI pour faciliter la compréhension et l'assimilation de mécanismes souvent compliqués .

Notre projet est découpé en trois chapitres :

- Le premier chapitre consiste à l'étude de l'existant qui donne une vue globale de l'architectures de réseau modèle OSI, en suit explique en détail a les 07 couches .
- Le deuxième chapitre concerne la conception de notre simulation par des algorithmes et des données (Tableaux) de chaque couche.
- Le dernier chapitre présent les outils utilisé lors de la réalisation du simulateur aussi bien quelques interfaces représentent notre application.

Enfin, nous conclurons ce projet par une conclusion générale.

CHAPITRE 01

Généralité sur le modèle OSI

1. Introduction

Le modèle OSI s'intéresse aux réseaux à commutation de paquets. Il s'agit d'un modèle à 7 couches qui décrit les réseaux. Chaque couche possède une problématique qui lui est propre. Alors, nous avons présentés dans ce chapitre un vis globale pour le modéle OSI , et nous avons expliqué les sept couches et son fonction .

2 Architectures des réseaux

Pour que les données transmises de l'émetteur vers le récepteur arrivent correctement avec la qualité de service exigée, il faut une architecture logicielle.

2.1: Le modèle TCP / IP

TCP/IP (Transmission Control Protocol/Internet Protocol) est actuellement le protocole de communication le plus utilisé dans les réseaux locaux. C'est aussi le protocole de transport utilisé par le réseau Interne Le modèle TCP/IP est constitué de 4 couches contrairement au modèle OSI qui comporte 7.

Le modèle TCP/IP propose de regrouper Les couches Application, Présentation , Session, sous une couche nommée Application tout simplement. Et de regrouper les couches liaison des données et physique sous le nom Hôte-réseau. Ce modèle renomme également la couche réseau en couche Internet. [09]

2.2 : Le modèle OSI :

OSI signifie **O**pen **S**ystems **I**nterconnection , ce qui se traduit par Interconnexion de systèmes ouverts. Ce modèle a été mis en place par l'ISO afin de mettre en place un standard de communications entre les ordinateurs d'un réseau, c'est-à-dire les règles qui gèrent les communications entre des ordinateurs. En effet, aux origines des réseaux chaque constructeur avait un système propre (on parle de système propriétaire). Ainsi de nombreux réseaux incompatibles coexistaient. C'est la raison pour laquelle l'établissement d'une norme a été nécessaire .[16]

Le rôle du modèle OSI

Le rôle du modèle OSI consiste à standardiser la communication entre les machines afin que différents constructeurs puissent mettre au point des produits (logiciels ou matériels) compatibles (pour peu qu'ils respectent scrupuleusement le modèle OSI).[16]

3. Système de couche

Le modèle OSI est composé de sept couches . Chaque couche peut interagir uniquement avec les deux couches adjacentes . Une couche N est constituée d'un ensemble d'entités formant un sous-système de niveau N . Elle ne peut dialoguer qu'avec une couche de même niveau N sur une autre machine. Les communications se font donc entre entités homologues. La communication entre deux entités homologues de niveau N obéit à un ensemble de règles et formats, syntaxiques et sémantiques, prédéfinis pour les entités de niveau N. Ces règles et formats définissent le protocole de niveau N.

Une couche de niveau N fournit des services pour la couche de niveau N + 1. La couche de niveau N + 1 communique à la couche N les caractéristiques du service attendu. Les services fournis par une couche N sont identifiés par des SAP (Service Access Point) ou ports . La figure suivante illustre cette.[01]

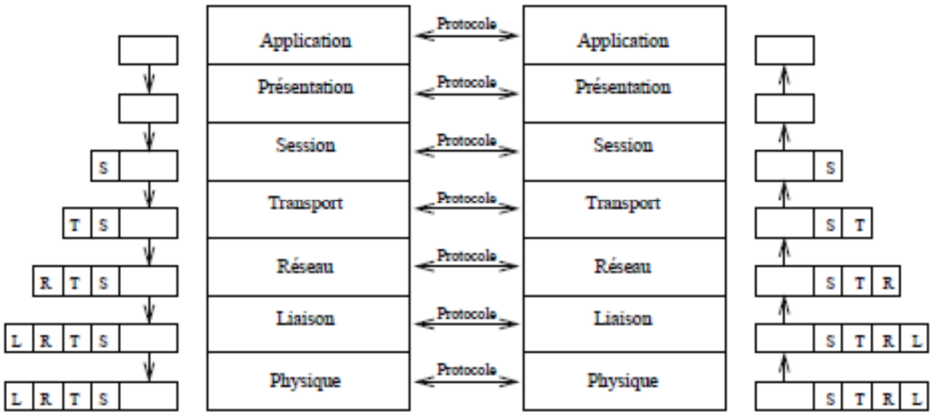


figure 01 : Modèle OSI en 7 couches.[10]

3.1: L'intérêt d'un système en couches

Le but d'un système en couches est de séparer le problème en différentes parties (les couches) selon leur niveau d'abstraction.

Chaque couche du modèle communique avec une couche adjacente (celle du dessus ou celle du dessous). Chaque couche utilise ainsi les services des couches inférieures et en fournit à celle de niveau supérieur .[17]

3.2: Interactions entre couches

Un protocole est un ensemble de règles et formats, syntaxiques et sémantiques prédéfinis pour les entités d'un même niveau N de deux machines différentes. La figure 02 montre les protocoles des couche .

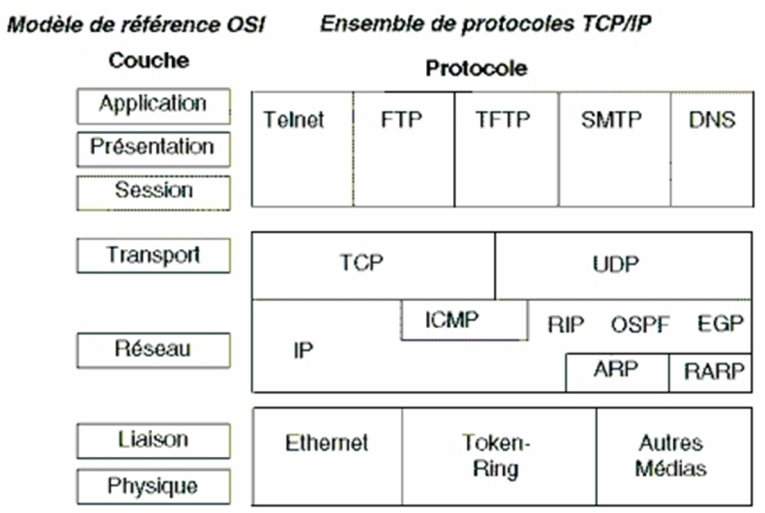


figure02 *:* Les protocoles de modèle OSI

Un service est fourni par une couche de niveau N à la couche de niveau N + 1 d'une même machine.

La figure 03 décrit la communication entre les 7 niveaux de couches de deux entités communicantes A et B .[10]

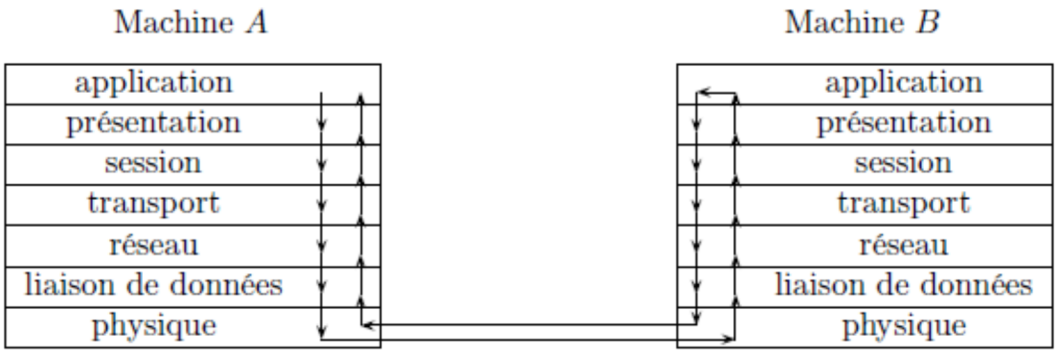


figure03 : Communication entre couches [10]

3.3 : Fonctionnement du modèle

Encapsulation

Lorsqu'une application envoie des données à travers le modèle OSI, les données traversent de haut en bas chaque couche jusqu'à aboutir au support physique où elles sont alors émises sous forme de suite de bits. Chaque couche ajoute aux données reçues de la couche supérieure des informations appelées "Entête" avant de les transférer à la couche inférieure. Cette entête permet à cette couche d'accomplir son rôle. A la réception, chaque couche retire l'entête, ajouté par la couche du même niveau, des données avant de les transmettre à la couche supérieure.[11]

Protocol Data Unit

PDU : est l'unité de mesure des informations échangées dans un réseau informatique. [18]

✓ Les procédures du niveau N+1 s'échangent des unités d'informations appelées (N+1)PDU en accord avec le protocole du niveau N+1.

✓ Chaque échange est réalisé grâce aux services fournis par la couche N (sauf si N+1 = 1 bien sûr!). Les procédures de niveaux N et N+1 d'un même site échangent des SDU.

✓ Plus précisément, entre les couches N+1 et N circulent des IDU (Interface Data Unit). Une IDU est composée de la manière suivante : IDU = ICI+ SDU

✓ N-SDU = information passée par N au niveau N-1 ou N+1.

✓ ICI (Information Control Interface) est l'information propre à l'interface N+1/N pour aider les services du niveau N. Elle n'est bien sûr pas transmise sur le réseau.

✓ De même, dans un protocole, l'entité d'information communiquée devient PCI + PDU où PCI (Protocol Control Interface) est l'information propre au protocole entre deux entités de même niveau.

La figure suivante illustre cette .

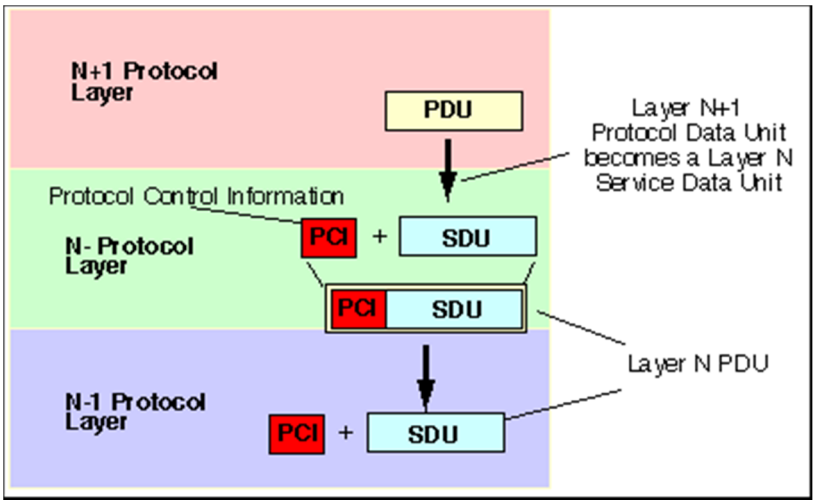


figure 04 : Protocol Data Unit [18]

4 : Les couches OSI

4.1 La couche application :

Cette couche est le point de contact entre l'utilisateur et le réseau. C'est donc elle qui va apporter à l'utilisateur les services de base offerts par le réseau, comme par exemple le transfert de fichier, la messagerie.[02]

4.1.1 Simple Mail Transfert Protocol

SMTP est le protocole applicatif qui permet de transporter les messages sur l'Internet. Il sait acheminer un message jusqu'à une boîte aux lettres, mais ne va pas plus loin.

Pour y arriver, il analyse dans un premier temps la partie de l'adresse située à droite du @ , pour trouver le domaine du destinataire. Si ce domaine le concerne, il cherche alors la boîte aux lettres du destinataire en regardant la partie de l'adresse située à gauche du @. Si le domaine du destinataire ne le concerne pas, il va chercher le serveur SMTP qui gère ce domaine, au moyen des champs MX du DNS du domaine destinataire et transmet le message à ce serveur. [03]

Utilité de l'entête

L'en-tête contient donc toutes les informations nécessaires pour [03] :

- ✓ Identifier l'auteur du message
- ✓ Identifier le destinataire
- ✓ Savoir à qui il faut répondre
- ✓ Retrouver le chemin suivi par le message
- ✓ Savoir comment a été codé ce message.

4.1.2 POP3 (Post Office Protocol 3)

Ce protocole est exclusivement utilisé pour le dialogue entre le client de messagerie et la boîte aux lettres. Il ne fait pas de transport sur l'Internet, il permet juste à l'utilisateur de gérer son courrier. [03]

4.2 La couche présentation

Cette couche s'intéresse à la syntaxe et à la sémantique des données transmises : c'est elle qui traite l'information de manière à la rendre compatible entre tâches communicantes. Elle va assurer l'indépendance entre l'utilisateur et le transport de l'information . Typiquement, cette couche peut convertir les données, les reformater, les crypter et les compresser.[02]

4.3 La couche session

C'est au niveau de la couche session que se négocie l'établissement d'une connexion avec un processus sur un autre système. Dès que la connexion est établie, c'est la couche session qui gère le dialogue d'une manière ordonnée. Si les utilisateurs, aux deux extrémités, veulent choisir parmi une variété d'options, par exemple une **communication semi-duplex** ou **une communication duplex** intégral, c'est à la couche session que revient la tâche de gérer ces options. Par ailleurs, la couche session marque le début et la fin des messages. Elle ajoute enfin un nouvel en-tête indiquant les ententes prises, entre autres sur le mode de communication.[04]

4.4 La couche transport

La couche de transport garantit que les messages sont remis sans erreur, dans l'ordre et sans pertes ou duplications. Cela soulage les protocoles des couches supérieures des problèmes liés au transfert de données.

La taille et la complexité d'un protocole de transport dépend du type de service auquel il peut accéder à partir de la couche réseau. Pour une couche réseau fiable avec capacité de circuit virtuel, une couche de transport minimale est requise. Si la couche réseau est peu fiable ou prend uniquement en charge des datagrammes, le protocole de transport doit inclure de récupération et détection d'erreur étendue. .[19]

Parmi les activités de cette couche, on peut citer [20]:

- **La réorganisation des segments** : lorsque de grands messages sont découpés en segments avant d'être envoyés sur le réseau, la couche transport doit **réordonner les segments à leur arrivée** du côté récepteur avant de pouvoir recomposer le message d'origine.
- **Le contrôle d'erreurs** : si des segments sont **perdus** lors de leur transmission ou s'il y a **duplication** de certains d'entre eux, la couche transport doit lancer le processus de correction d'erreur. Cette couche détecte aussi les segments altérés en assurant un contrôle de bout en bout à l'aide de techniques telles que les sommes de contrôle (checksum).
- **Le contrôle de flux de bout en bout** : la couche transport utilise des **acquittements** pour gérer le contrôle de flux de bout en bout entre deux entités connectées . En plus de l'acquittement négatif, certains protocoles de ce niveau peuvent demander la retransmission des segments les plus récents , comme la montre la figure suivante :

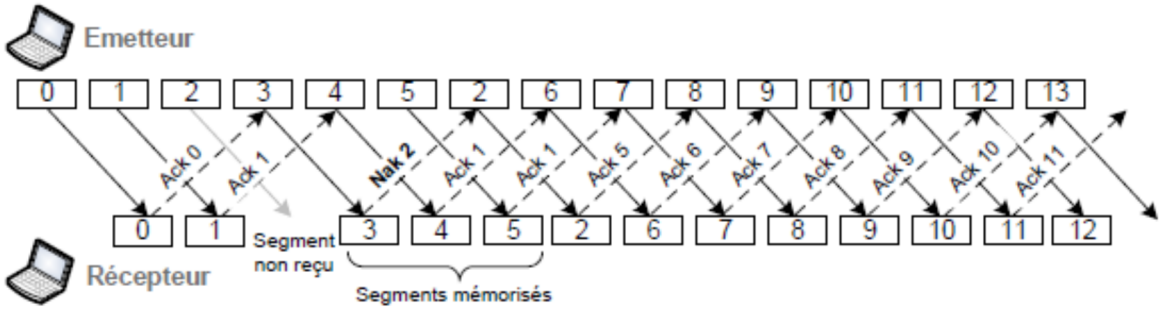


figure 05 : L'envoi des segments[20]

le domaine des communications de bout en bout indépendantes de l'état du sous-réseau. Les paquets de la couche réseau peuvent être acheminés à destination par des chemins différents et dans le désordre. Par nature, le protocole IP n'offre pas de garantie. Les routeurs peuvent se débarrasser des paquets suivant plusieurs critères tels que des erreurs sur les sommes de contrôle, des congestions de trafic sur les interfaces réseau ou des adresses ne correspondant à aucune route connue. Tous les programmes et services de la couche application ne peuvent se contenter d'un mode de fonctionnement aussi «fragile». C'est une des raisons pour laquelle deux protocoles distinct sont été développés pour la couche transport.

4.4.1 Protocole TCP

le premier protocole de transport développé pour l'Internet. Les premières spécifications ARPANET prévoyaient un transport de l'information très fiable indépendant du type et de l'état du réseau. Le fonctionnement du protocole TCP a été décrit dans le document RFC793 : *Transmission Control Protocol* .[21]

✓ **Protocole de bout en bout.**

Les processus pairs des couches transport de deux équipements connectés dialoguent l'un avec l'autre sans rien connaître du réseau. C'est au niveau IP que l'on se préoccupe de la fragmentation et du réassemblage des paquets.

✓ **Protocole orienté connexion.**

La fiabilité du transport TCP dépend de l'établissement d'une connexion entre les processus pairs qui veulent dialoguer. L'établissement d'une connexion est réalisé par l'échange d'informations telles que le numéro de port, le numéro de séquence et la taille de fenêtre.

✓ **Établissement d'une connexion TCP**

TCP utilise les segments pour déterminer si le système récepteur est prêt à recevoir les données. Lorsque le protocole TCP de l'hôte émetteur souhaite établir les connexions, il envoie un segment appelé **SYN** au protocole TCP de l'hôte récepteur. Le TCP récepteur renvoie un segment appelé **ACK** afin d'accuser la réception du segment. Le TCP émetteur envoie un autre segment ACK, puis initialise l'envoi des données. Cet échange d'informations de contrôle est appelé **négociation en trois étapes**.

4.4.2 Description de l'entête TCP

✓ **Port source (16 bits) :** le numéro de port de la source.

✓ **Port Destinataire (16 bits)** : le numéro de port du destinataire.

✓ Numéro de séquence : indique le numéro du premier octet transmis dans le message. À l'ouverture de connexion un numéro de séquence initial (ISN) est attribué d'une manière aléatoire

✓ Numéro d'accusé de réception : contient le numéro de la séquence du prochain octet attendu.

✓ Longueur de l'en-tête : indique la longueur de l'en-tête TCP en mot de 4 octets.

✓ Six bits de drapeaux :

— URG : si positionné, indique que la valeur du champ message urgent est significative,

— ACK : si positionné, indique que la valeur du champ acquittement est significative,

— PSH : Les données reçues doivent être transmises immédiatement à la couche supérieure,

— RST : fermeture de la connexion suite à une erreur,

— SYN : ouverture de la connexion,

— FIN : fin de la connexion.

✓ Fenêtre : indique le nombre d'octets que le récepteur peut accepter. Ce champ permet de gérer le contrôle de flux. Ainsi la valeur de la fenêtre indique à l'émetteur le nombre d'octets qu'il peut envoyer sans être acquittés

✓ Pointeur message urgent : indique des octets qui doivent être traités en priorité.

✓ Checksum: 16 bits Somme de contrôle sur 16 bits de l'en-tête et des données.[22]

4.4.3 Le protocole UDP

Le protocole UDP (User Datagram Protocol) est apparu avec le développement des réseaux locaux dont la fiabilité permet de s'affranchir des fonctions de contrôle. C'est un protocole minimum sans garantie de délivrance des messages et sans séquencèrent .[20]

Format de l'en-tête UDP:

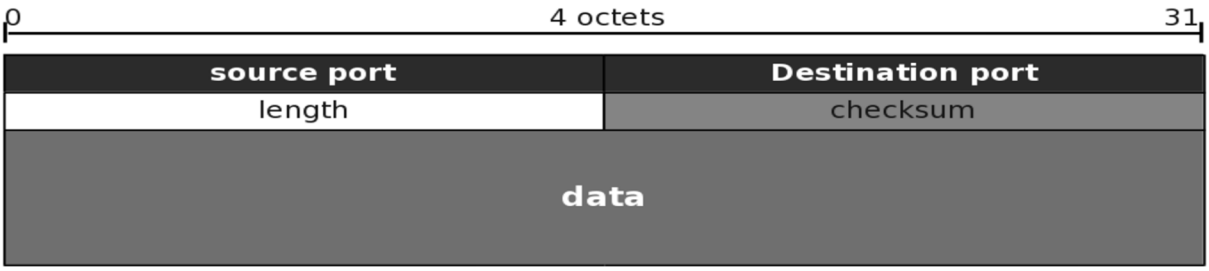


figure 06: Format de l'en-tête UDP.[20]

4.4.4 Description de l'en-tête

- ***Source Port***: 16 bits Numéro du port source. Ce champ est optionnel.
- ***Destination Port***: 16 bits Numéro du port destination.
- ***Length***: 16 bits Longueur en octets du datagramme UDP incluant l'en-tête et les données.
- ***Checksum*** : 16 bits Somme de contrôle sur 16 bits de l'en-tête et des données.[05]

4.5 La couche réseau

Construit une voie de communication de bout en bout à partir de voies de communication avec ses voisins directs. Ses apports fonctionnels principaux sont donc:

- le routage : détermination d'un chemin permettant de relier les 2 machines distantes .
- le relayage : retransmission d'un PDU dont la destination n'est pas locale pour le rapprocher de sa destination finale.

Cette couche est donc la seule à être directement concernée par la topologie du réseau. C'est aussi la dernière couche supportée par toutes les machines du réseau pour le transport des données utilisateur: Les couches supérieures sont réalisées uniquement dans les machines d'extrémité. [23]

4.5.1 Description de l'en-tête

- **VER**: (4 bits) Version du protocole IP qui doit interpréter ce datagramme. La version actuelle la plus "déployée" est 4 (soit 0100 en binaire)
- **HLEN**: (ou IHL) Internet Header Length(4 bits)

Cette valeur est à multiplier par 4 pour connaître le nombre d'octets constituant l'en-tête. L'en-tête correspond au début du datagramme jusqu'au données. Permet notamment de savoir si il y a des options et où commencent les données. Peut varier de 5 (soit 20 octets d'entête) à 15 (60 octets d'entête). Cette variation s'explique par la présence ou non d'options.

✓ **Type of service** est un champ codé sur 8 bits qui attribue une qualité de service liée au paquet. C'est à dire sa priorité, le délai, le débit et la fiabilité du paquet.

Total Length est un champ codée sur 16 bits qui indique la longueur du paquet en octets, en-tête inclus. Si dans la théorie le paquet peut faire au maximum 65535 octets, dans la pratique, la taille est d'environ 576 octets au plus.

✓ **Identificati**on est une valeur donnée par la station émettrice en vue de reconstruire les fragments d'un message. Le détail du rôle de ce champ est expliqué dans la RFC 815.

✓ **Flags** est un champ codé sur 3 bits qui gère également la fragmentation des messages, suivant la valeur de chaque bits du champ :

– Bit 0, valeur 0,

– Bit 1, 0 = may fragment, 1 = do not fragment. Celui-ci indique si le paquet peut-être fragmenté ou non,

– Bit 2, 0 = last fragment, 1 = more fragments. Celui-ci indique si des fragments doivent encore venirou non.

✓ **Fragment** offset (sur 13 bits) indique la place du fragment dans le paquet. Le premier fragment d'un paquet a ainsi un offset de '0',

✓ **Protocol** est un identifiant codé sur 8 bits permettant de déterminer le protocole qui est encapsulé dans le datagramme IP

✓ **Time to Live** est aussi appelé en abrégé TTL. Codé sur 8 bits, ce champ indique la durée de vie du paquet. Cette dernière est exprimée en nombre de sauts de réseaux à réseaux. Chaque fois que le paquets change de réseau, le champ est décrémenté. Quand ce dernier tombe à 0, le paquet est alors détruit. Ceci permet de détruire des paquets qui ne peuvent pas être acheminé à leur destination.

✓ **Padding** est un champ de bourrage. Celui-ci est constitué de '0' pour compléter l'en-tête jusqu'à ce que sa longueur soit un multiple de 4 octets.[06]

- **Longueur Totale**: (16 bits)Donne la taille totale en octets du datagramme (ou du fragment). Ainsi, un datagramme ne peut pas excéder65535 octets . La norme impose à toute implémentation de pouvoir traiter des datagrammes d'au moins 576 octets.

- **checksum (16 bits)** : ce champ contient une valeur codée sur 16 bits qui permet de contrôler l'intégrité de l'en-tête afin de déterminer si celui-ci n'a pas été altéré pendant la transmission. La somme de contrôle est le complément à un de tous les mots de 16 bits de l'en-tête (champ somme de contrôle exclu). Celle-ci est en fait telle que lorsque l'on fait la somme des champs de l'en-tête (somme de contrôle incluse), on obtient un nombre avec tous les bits positionnés à 1 .

- **Adresse IP source** (32 bits) : Ce champ représente l'adresse IP de la machine émettrice, il permet au destinataire de répondre

- **Adresse IP destination** (32 bits) : adresse IP du destinataire du message . [07]

4.5.2 Protocole IP

La fonction ou rôle du Protocole Internet est d'acheminer les datagrammes à travers un ensemble de réseaux interconnectés. Ceci est réalisé en transférant les datagrammes d'un module Internet à l'autre jusqu'à atteindre la destination. Les modules Internet sont des programmes exécutés dans des hôtes des routeurs du réseau Internet. Les datagrammes sont transférés d'un module Internet à l'autre sur un segment particulier de réseau selon l'interprétation d'une adresse Internet. De ce fait, un des plus importants mécanismes du

protocole Internet est la gestion de cette adresse Internet . Lors de l'acheminement d'un datagramme d'un module Internet vers un autre, les datagrammes peuvent avoir éventuellement à traverser une section de réseau qui admet une taille maximale de paquet inférieure à celle du datagramme. Pour surmonter ce problème, un mécanisme de fragmentation est géré par le protocole Internet.[07]

4.5.3Format des adresses :

Les adresses IP sont composées de 4 octets. Par convention, on note ces adresses sous forme de 4 nombres décimaux (0-255) séparés par des points. Même s'il est techniquement possible d'associer plusieurs interfaces à une adresse IP, la définition originale de l'adressage impose une affectation unique d'adresse par interface. [07]

Les 4 octets de la définition des adresses IP rassemblent une partie réseau et une partie hôte.

• La partie réseau est commune à l'ensemble des hôtes d'un même réseau,

• La partie hôte est unique et désigne une seule interface physique.

Prenons un exemple d'adresse IP pour identifier les différentes parties :

Tableau 1. Exemple : adresse 192.168.100.1

Adresse complète 192.168.100.1:
Masque de réseau 255.255.255.0
Partie réseau 192.168.100.
Partie hôte .1
Adresse Réseau 192.168.100.0
Adresse de diffusion 192.168.100.255

4.6 La Couche liaison

Elle transforme les flots de bits en lignes de données sans erreur. Pour cela, elle fractionne les données d'entrée de l'émetteur en trames de données. C'est donc à elle de reconnaître les frontières des trames. Cette fonction entraîne la résolution des problèmes de trames endommagées, perdues ou dupliquées.

On retrouve ici la fonction de contrôle d'erreur. C'est aussi à ce niveau que l'on peut trouver des mécanismes de régulation pour éviter la saturation du canal de communication par un émetteur unique. C'est la fonction de contrôle de flux. Une technique très simple, employée dans les réseaux Ethernet, interdit les émissions continues à tous les hôtes du réseau. Une émission ne peut avoir lieu que si tous les hôtes ont eu le temps matériel de détecter que le média partagé est libre .Si le service le requiert, le récepteur confirme la réception de chaque trame en émettant une trame d'acquittement.

Les réseaux à diffusion utilisent un service ou une sous-couche spécifique pour contrôler que l'accès au média est libre. Dans le cas des réseaux sans-fil de types IEEE 802.11 ou Wifi, des trames de gestion indépendantes des informations utilisateur sont échangées entre les équipements pour contrôler l'accès aux canaux hertziens.[20]

4.5.1 *Une trame:*

- Une suite de bits (d'une longueur variable mais bornée)
- Le début et la fin de trame sont souvent identifiés par des délimiteurs
- Composée d'un certain nombre de champs ayant chacun une signification précise.
- On distingue souvent 3 ensembles de champs : l'entête (header), le champ de données, la terminaison (trailer) [13]

4.6.2 PROTOCOLE HDLC

- est un protocole de couche liaison de données (couche 2 du modèle OSI)
- est un ensemble de classes, de procédures et de fonctionnalités optionnelles (normalisée par l'ISO en 1976)

La figure suivant montre le format de trame :

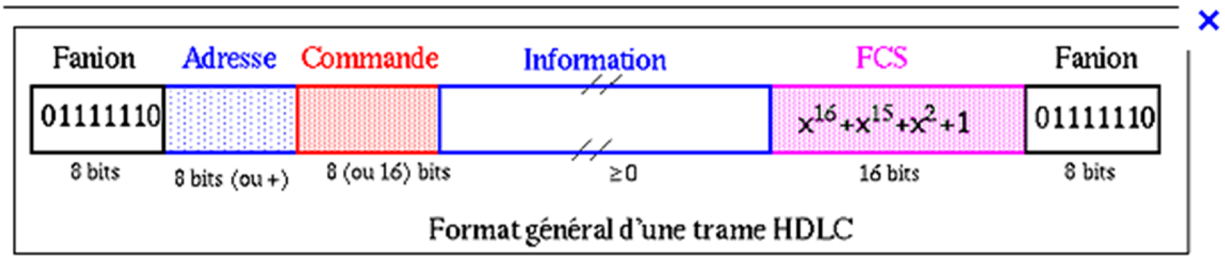


figure 07: format trame HDLC [13]

Caractéristiques

– Transmission synchrone

– Orienté bit

– Liaisons point-à-points ou multi-points

4.6.3 *Détection et correction des erreurs*

les bruits dans les supports de communication la perte de synchronisation entre émetteur et récepteur produisent des perturbations entrainant des erreurs dans les transmissions de données Plusieurs mécanismes de détection d'erreur existent :

- ✓ détection par écho
- ✓ détection par répétition
- ✓ détection d'erreur par clé calculée

- **Un code détecteur/correcteur** d'erreurs est un groupe de bits que l'émetteur ajoute au message à transmettre. Ce groupe de bits, appelé FCS (Frame Check Sequence), dépend du message et introduit de la redondance.[08]

La figure 08 explique la détection d'erreur :

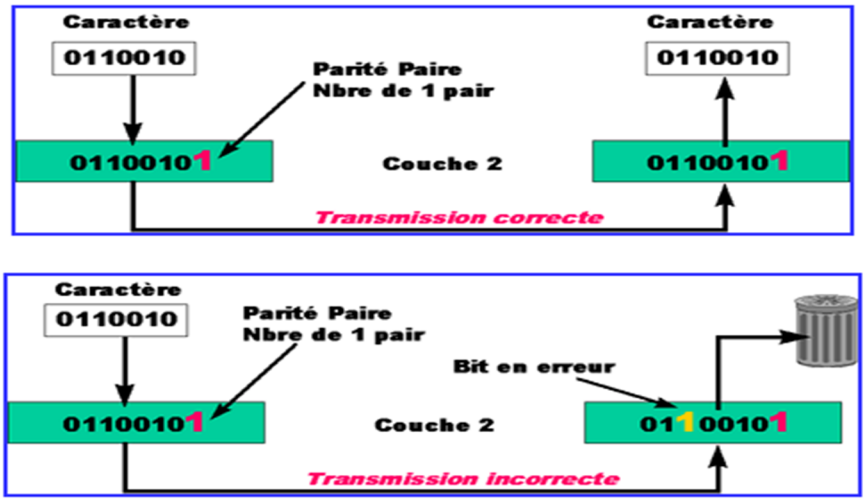


figure 08: détection d' erreur

- **Codes à contrôle de parité**

Les codes à contrôle de parité sont de parité soit paire, soit impaire. Dans le premier cas, on va protéger une séquence de bits en ajoutant un nouveau bit de telle sorte que le nombre de bits

ayant la valeur 1 (dans la séquence protégée plus le bit introduit) soit pair. Dans le second cas, ce nombre doit être impair.

- **VRC (Vertical Redundancy Check)**

C'est la technique la plus simple. Un code ASCII étant défini sur 7 bits, on utilise le 8ème bit de l'octet pour introduire le code vérificateur.

Exemple : Pour transmettre la chaîne de caractères IUT, on code chaque lettre en ASCII, puis on ajoute le code de parité.

Lettre	ASCII	VRC pair	VRC impair
I	1001001	**1**1001001	**0**1001001
U	1010101	**0**1010101	**1**1010101
T	1010100	**1**1010100	**0**1010100

Pour envoyer le message avec un code de parité pair, on transmet (avec l'ordre d'envoi des bits de gauche à droite) :

$$\underbrace{\mathbf{1}1001001}_{I}\ \underbrace{\mathbf{0}1010101}_{U}\ \underbrace{\mathbf{1}1010100}_{T}$$

Ce code permet de détecter les erreurs en nombre impair sans pouvoir corriger. Il est peu efficace.

- **LRC (Longitudinal Redundancy Check)**

Le principe est similaire à celui du VRC, mais au lieu de protéger les caractères un par un, on protège l'ensemble des bits de même rang de tous les caractères. On obtient alors un code de protection sur 7 bits. .[12]

Exemple : Pour protéger IUT, on calcule le code :

I	1001001
U	1010101
T	1010100
LRC pair	1001000
LRC impair	0110111

Pour envoyer le message avec un code de parité pair, on transmet :

$$\underbrace{1001001}_{I}\ \underbrace{1010101}_{U}\ \underbrace{1010100}_{T}\ \underbrace{1001000}_{LRC}$$

L'efficacité du code LRC dépend fortement du message transmis.

- **LRC et VRC**

On peut également combiner les deux techniques précédentes. On protège alors chaque caractère par un code VRC et l'ensemble des bits par un code LRC. On obtient donc un LRC sur 8 bits. La parité des LRC et VRC utilisés est la même (tous les deux pairs ou tous les deux impairs).

Exemple : Pour transmettre la chaîne de caractères IUT, on code chaque lettre en VRC puis en LRC .[20]

		VRC pair	VRC impair
I	1001001	**1**1001001	**0**1001001
U	1010101	**0**1010101	**1**1010101
T	1010100	**1**1010100	**0**1010100
LRC		01001000	00110111

Pour envoyer le message avec un code de parité pair, on transmet :[03]

$$\underbrace{11001001}_{I}\ \underbrace{01010101}_{U}\ \underbrace{11010100}_{T}\ \underbrace{01001000}_{LRC}$$

4.6.4 fanion

- Le champ « fanion » indique les bordures de la trame (début et fin)
- Il est représenté par un 0 suivi de 111111 suivi de 0.
- si la données contient la même séquence de bits (donnée= ...01111110...) . ajouter un 0 après chaque 11111 (5 un consécutifs au niveau de l'émetteur). [08]

Bits de transparence

Pour éviter qu' un fanion se retrouve parmi les données a envoyer, on ajoute des bits de transparence . plus précisément , on insère un bit 0 après chaque séquence de cinq 1 consécutifs trouvés dans les donnes à transmettre , comme le montre la figure 09. [08]

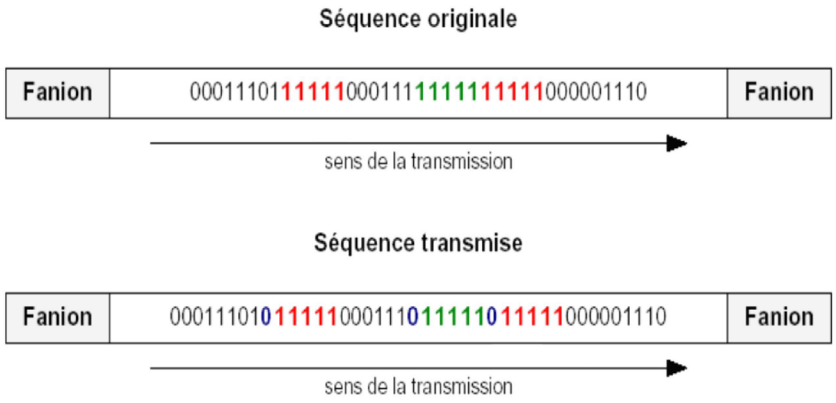


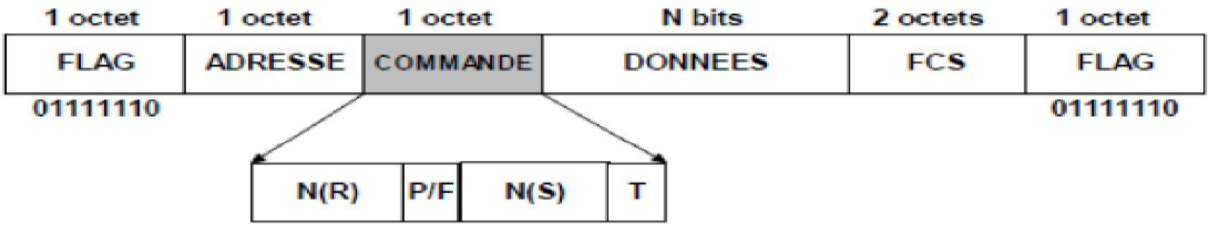
figure 09 : Bits de transparence [08]

4.6.5 *Adresse*

- L'adresse est celle du destinataire à qui est envoyée la trame. Cette adresse était utilisée lorsque la communication était de type maître-esclave, l'adresse étant celle de l'esclave.
- Le champ adresse identifie la station secondaire dans le cas d'une liaison multipoint
- Dans une commande il représente la station destinataire.
- Dans une réponse il représente la station émetteur.
- Dans le cas de liaison point-à-point il n'est pas pris en compte .

4.6.6 *Commande (control)*

Ce champ permet de distinguer 3 types de trames :



- T (1 bit) : Indique le type de trame.
  0 : information, 1 : supervision.
- N(S) et N(R) (6 bits) : Indique le numéro des trames émises et reçues.
- P/F (1 bit) : Demande de réponse immédiate à la suite de l'envoi d'une trame de Commande .

Numérotation des trames

– V(S) : numéro de la prochaine trame à envoyer (0 à 7) .

– V(R) : numéro de la prochaine trame attendue en réception (0 à 7) .

– N(S) : numéro de la trame .

– N(R) : acquittement des trames reçues de numéro strictement inférieur à N(S) .

types de trames circulent sur la liaison

les trames d'information (I Information) : trames de données (+Ack) ;trame I d'information contenant essentiellement des données et des indications sur l'état de la transmission ;

Transportent les données utilisateurs

les trames de supervision (S Supervisory) : trames de supervision (+Ack) .

- **Acquittement RR – RNR**

RR ("Received & Ready") - 00 : acquittement

- confirme la réception des trames de données de nº < N(R) .

- **Retransmission REJ – SREJ**

RNR ("Received & Not Ready") - 10 :contrôle de flux .

- confirme la réception des trames de données de nº < N(R) .
- interdit la transmission des trames .

- **Contrôle de flux RR – RNR**

SREJ (" Selective Reject ") - 11 : protection contre les erreurs.

- confirme la réception des trames de données de nº < N(R).
- demande la retransmission de la trame de nº = N(R) .[09]

4.7 La couche physique

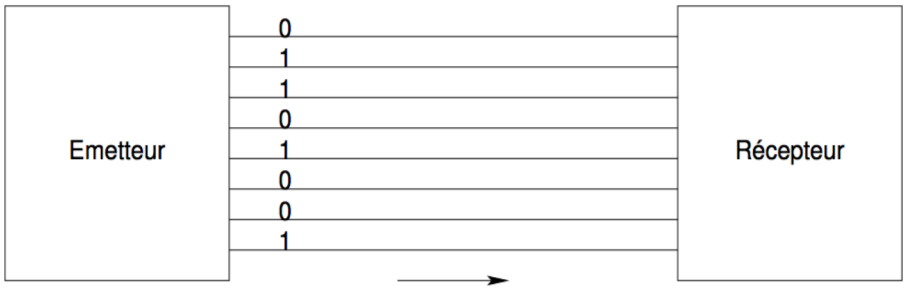
La rôle de la couche physique est de transformer une suite de bits en signaux (et inversement) pour les adapter au canal de communication et les transmettre d'une machine à une autre. Les bits transformés représentent des informations numérisées (codées) tel que le

code ASCII pour les textes . La couche physique détermine la façon selon laquelle les bits sont transportés sur le support physique. Elle permet d'introduire les bits 0 et 1 sur le support sous une forme spécifique, reconnaissable par le récepteur. Plusieurs composants sont utilisés dans cette couche, comme les modems, multiplexeurs, concentrateurs, etc. Ce chapitre étudie les supports de transmission et leurs caractéristiques ainsi que les méthodes utilisées pour transmettre l'information sur ces supports.[14]

4.7.1 *Modes de transmission*

Les blocs d'informations transmis sur des fils peuvent l'être en parallèle ou en série.

Transmission en parallèle: Les ordinateurs manipulent non pas des bits isolés, mais des mots de plusieurs bits aussi bien pour le calcul que pour le stockage. On est donc conduit à imaginer un système de transport dans lequel les différents bits d'un mot sont véhiculés en parallèle. Cela implique que pour des mots de N bits il faut N lignes de transmission.



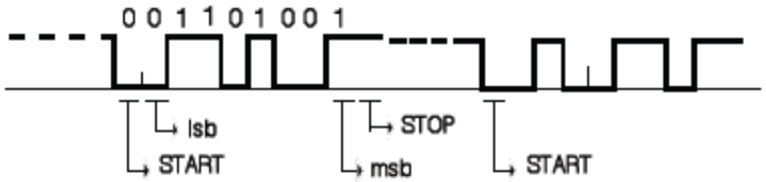
Transmission en parallèle

Transmission en série **:** Dans ce mode, les bits sont transmis les uns derrière les autres, ce qui nécessite une "sérialisation" effectuée par une logique de transmission dont la pièce maîtresse n'est autre qu'un registre à décalage dont le fonctionnement est rythmé par une horloge.

Une difficulté majeure de ce mode de transmission est liée à l'horloge ; en effet, il est nécessaire d'employer une horloge d'émission et une horloge de réception qui doivent fonctionner en synchronisme parfait. [14]

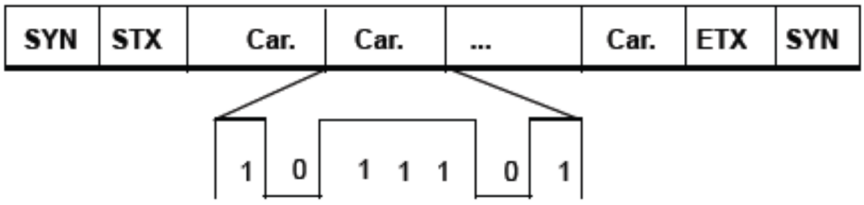
1. **Transmission asynchrone:** Ce mode est asynchrone au niveau caractère.

- Synchronisation au niveau des bits du caractère.
- Un bit(s) START et un bit(s) STOP.



2. **Transmission synchrone** : Échange sous forme de blocs. Chaque bloc est précédé par un caractère(s) de synchronisation (SYN = 00010110).

- SYN sert à reconnaître le début du bloc de transmission et à synchroniser les horloges des systèmes communicants.
- Un bloc débute par STX (00000010) et se termine par ETX..



4.7.2 Supports de transmission (en bande de base)

– *Code tout ou rien:*

• 0 ⇒0 (volt)

• 1 ⇒+V

– *Code NRZ:*

• 0 ⇒-V

• 1 ⇒+V

–*Code bipolaire:*

• 0 ⇒0

• 1 ⇒alternativement +V, -V

– *Code RZ:*

• 0⇒ 0

• 1 ⇒+V durant la 1ère moitié de l'intervalle et 0 durant la2ème moitié

– *Code Miller:*

• 0 ⇒pas de transition si le bit suivant est 1, transition à la fin de l'intervalle si le bit suivant est 0

• 1 ⇒transition au milieu de l'intervalle 0

Code Manchester:

Le code Manchester, également appelé code biphase, transforme chaque bit en deux bits : un bit 1 est transformé en deux bits 10, et un bit 0 est transformé en 01. Puis le codage NRZ est employé. Le code Manchester permet de corriger les deux défauts du NRZ : il n'y a pas de composante continue, et les transitions à chaque temps-bit permettent au récepteur de récupérer l'horloge. Mais il y a un prix à payer : tout se passe comme si le débit était deux fois plus important, d'où une bande occupée deux fois plus importante que pour un codage NRZ. Ce code était utilisé dans la version 10BT d'Ethernet.

Exemple : Le codage Manchester de la séquence de bits 1110 0001 1011 est représenté dans la figure suivant .[15]

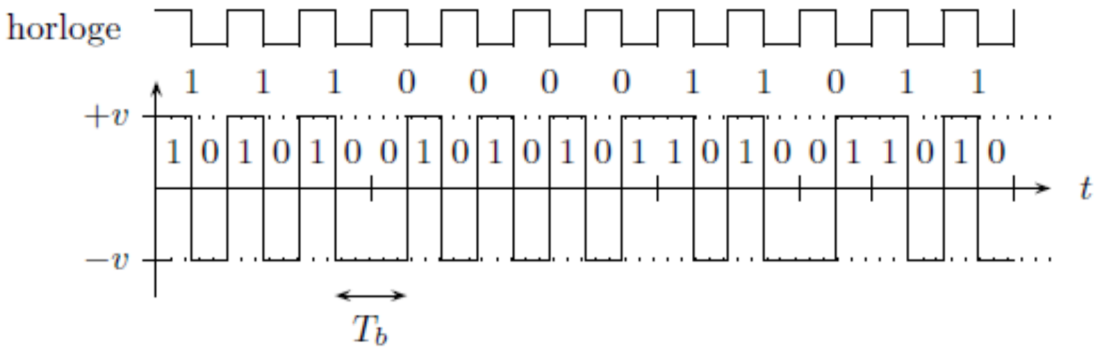


figure10– Un codage Manchester [15]

5 Conclusion

Le modèle OSI est une des structures réseau les plus étudiées car il permet de bien comprendre les principes. OSI a été conçu pour convenir à tous les types d'utilisation que l'on pouvait souhaiter en faire. Par conséquent certaines couches sont inutiles pour certaines utilisations. Le modèle OSI n'est que très peu utilisé dans la réalité. Malgré une mise à jour du modèle en 1994, OSI a clairement perdu la guerre face à TCP/IP. Même si c'est maintenant TCP/IP qui est le plus utilisé, les gens continuent à utiliser OSI come modèle de référence. Certains ont tendance à voir TCP/IP comme l'implémentation réelle de OSI.

Le modèle Internet a été créé afin de répondre à un problème pratique, alors que le modèle OSI correspond à une approche plus théorique, et a été développé plus tôt dans l'histoire des réseaux. Le modèle OSI est donc plus facile à comprendre, mais le modèle TCP/IP est le plus utilisé en pratique. Il est préférable d'avoir une connaissance du modèle OSI avant d'aborder

TCP/IP, car les mêmes principes s'appliquent, mais sont plus simples à comprendre avec le modèle OSI .

CHAPITRE 02

Conception

1. Introduction

Chaque couche de modèle OSI est constituée d'éléments matériels et logiciels et offre un service à la couche située immédiatement au-dessus d'elle en lui épargnant les détails nécessaires d'implémentation.

Chaque couche n d'une machine gère la communication avec la couche n d'une autre machine en suivant un protocole de niveau n qui est un ensemble de règles de communication pour le service de niveau n.

En fait, aucune donnée n'est transférée directement d'une couche n vers une autre couche n, mais elle l'est par étapes successives. Supposons un message à transmettre de l'émetteur A vers le récepteur B. Ce message, généré par une application de la machine A va franchir les couches successives de A via les interfaces qui existent entre chaque couche pour finalement atteindre le support physique. Là, il va transiter via différents nœuds du réseau, chacun de ces nœuds traitant le message via ses couches basses. Puis, quand il arrive à destination, le message remonte les couches du récepteur B via les différentes interfaces et atteint l'application chargée de traiter le message reçu.

Une couche correspond à un ensemble de fonctions ou de processus cohérents entre eux et assurant une fonction précise globale. Deux couches adjacentes sont indépendantes dans leurs fonctions mais elles s'interconnectent par ce qu'on appelle une interface. C'est au niveau des interfaces que les couches s'échangent des données : une couche utilise les fonctionnalités de la couche inférieure et propose ses fonctionnalités à la couche supérieure. Chaque couche est caractérisée par un ensemble de fonctions et de messages de contrôle associés. Les messages de contrôle sont appelés Unités de Protocolede Données PDU. Ils sont échangés par des entités dans une couche donnée.

Dans ce chapitre, nous allons présenter chaque couche du modèle OSI sous la forme d'un ensemble des données (Tableaux) et des opérations (Algorithmes) qui les affectent.

Donc le principe général proposé est:

Tableau (Donnée Couche X)+Algorithme (Opérations Couche X) => Tableau(Donnée Couche X- 1)

Nous tenons à souligner que les données transitant sur le réseau est très nombreux, soit en termes de:quantité , types ou formes ; et nous ne pouvons pas les étudiés pleinement dans un tel projet, On selimiterons à l'exemple des données de texte (lettres et chiffres) comme exemple les données de texte d'un e-mail.

Chaque caractère de texte réserve dans la mémoire 8 bit (1 octet), comme nous le verrons dans la suite :

2. Les définitions

Dans ces définitions, on a déclaré à toutes les en têtes (l'en tête SMTP , l'en tête TCP l'en tête IP et l'entête du protocole HDLC) et le UDP (segment , trame , paquet) pour chaque couche.

L'en tête SMTP

Cette définitions pour déclaré l'en tête SMTP

Type def struct tete smtp ={

Return-Path[8] , FromTo[16] , Received [16] ,Subject[32] , MIME-Version[8]

Content-Type[8] , charset[4] , Transfer [8] :string ;

Message-ID[8] : entier ;

Date[16] : date ;

}

L'en tête TCP

Cette définitions pour déclaré l'en tête TCP

Type def struct tetet cp = {

d'urgence[8] , OpitionTCP[44] , Remplissage[16] , Réservé[6] , URG[1] , ACK[1]

PSH[1] , RST[1] , SYN[1] , FIN[1], Checksum[16] , Pointeurd'ur[16] : bool;

Portsour[16] , Portdes[16] ,NuméroS[32],Numérod'ac[32] longueur[32] : entier ;

}

L'en tête IP

Cette définitions pour déclaré l'en tête IP

Type def struct tete ip={

Protocol[8] :stringe ;

Header checksum[16] , Source address[32] , Destination address[32] , IHL[4] , Type of service[8] , Total length[16] ,Identification[16] , Flags[3] , Fragment offset[13] , Time to live[8] ,Padding[4] , Version[4] , Longueur Totale[16] :entier;

}

L'en tête HDLC

Cette définitions pour déclaré case contrôle d'en tête HDLC

- ✓ **Case control :**

Type def struct tete Ctrl={

Control [8] : entier ; }

- ✓ **L'adresse MAC :**

Cette définitions pour déclaré case adresse MAC d'en tête HDLC

Type def struct tete Mac={

MacSend [4] :entier;

MacRece[4] : entier ;

}

- ✓ **Début d'en tête HDLC(control, adresse MAC , Fanion) :**

Cette définitions pour déclaré début d'en tête HDLC

Type def struct tete debut={

Control :teteCtrl;

MacSR: teteMac;

Fanion : entier;

}

- ✓ **L'afin d'en tête HDLC (FCS , Fanion)**

Cette définitions pour déclaré L'afin d'en tête HDLC

Type def struct tete fin = {

Fanion[8] : entier ;

Fcs [16] : entier ;

}

- **Le element :**

Cette définitions pour déclaré le element **.**

L'élément contenant case des types char pour stocké le caractère et 8 cases pour stocké représentation en binaire

Type def struct element ={

Ch : char;

Ascii[8] : bool; }

- **Le Segment :**

Cette définitions pour déclaré le segment **.**

Le segment contient les cases de élément (donnée)et l'entête SMTP .

Type def struct segment ={

SMTP :tetesmtp;

cases[472] : element;

}

- **Le Paquet :**

Cette définitions pour déclaré le paquet .

Le paquet contient les cases d' élément (données) en tête SMTP et en tête TCP .

Type def struct packet = {

SMTP :tete smtp;

Tcp:tete tcp;

cases[472] : élément;

}

- **Le Trame :**

Cette définitions pour déclaré le trame .

Le trame contient les cases de élément (donnée) et l' en tête SMTP et l' en tête TCP et IP .

Type def struct trame = {

SMTP :tete smtp;

Tcp :tete tcp;

ipSR: tete ip;

cases[472] : élément;

}

- **Déclaration général :**

Cette définitions pour déclaré toute les information de la fin couche

Couche physique contient les cases d'élément (donnée) l'entête SMTP et l'entête TCP et l'entête IP et l'entête de couche Liaison de données .

Type def struct final = {

SMTP :tete smtp;

Tcp :tete tcp;

ipSR :tete Ip;

debut :tete debut;

fin :tete fin;

cases[472] : element ;

}

3. Les algorithms des 07 couches

3.1 Couche application

	Les entres	Les opérations	Les sorties
Couche 7: application	Données: Contient le message (D) (texte e-mail)	-Lecturedu message -stockagedu message dansun tableau -application du protocole SMTP (Algorithme «3.1.a») -ajout l'en tête SMTP (Tableau)	nouvelle Données (D1)

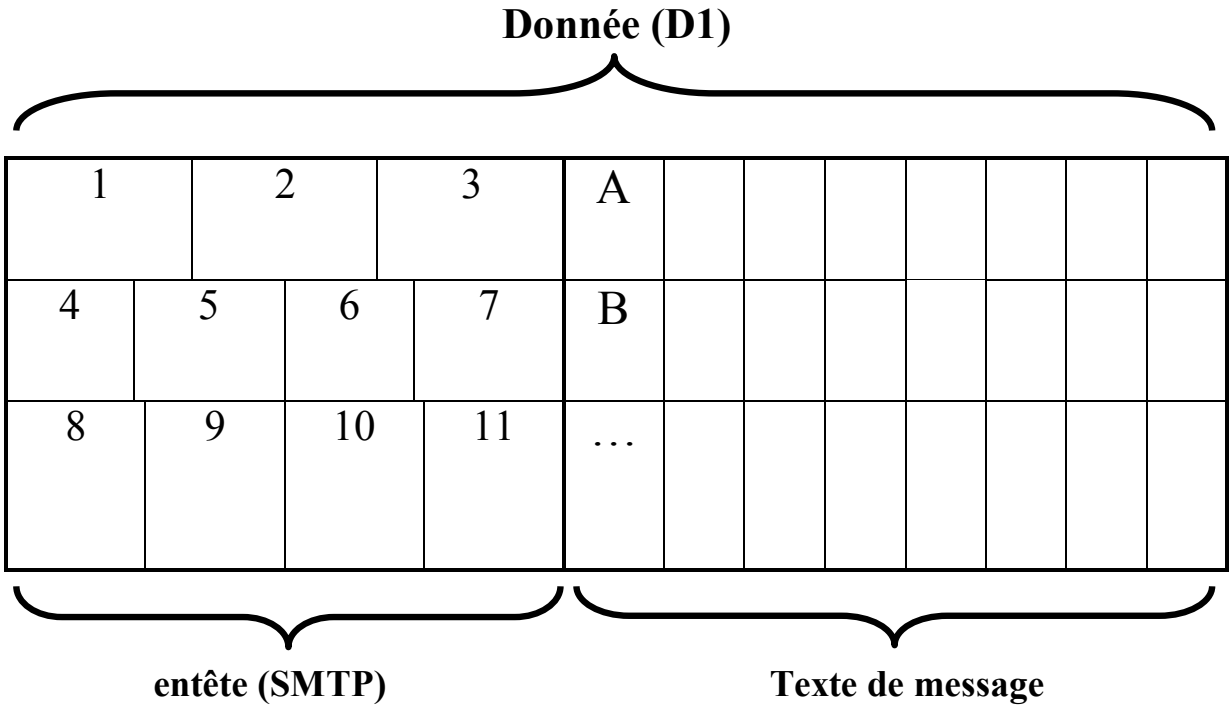
Tableau 1 :Opérations couche application

3.1.a L'algorithme

Cette procédure pour stocker le message dans tableau et ajouter la donnée a l'entête SMTP.

```
Procedure  Ch1Application(* Tab[N] : Char)
Déclaration  :
Variable
i  :  entier ;
c  :   char ;
Début
|         i ← 1 ;
|         Tanque(c != '/') faire
|         |         lire c ;
|         |         Tab[i]← c;
|         FinTQ
Fin
```

3.1.b Représentation



1_Return-Path2_Received 3_ Message-ID 4_ From 5_To 6_Subject 7_Date

8_ MIME-Version 9_Content-Type 10_ charset 11_Transfer.

3.2 Couche presentation

	Les entres	Les opérations	Les sorties
Couche 6: Présentation	Données (D2)	-Convertir le texte en système binaire : utilisation la code ASCII (Algorithme « 3.2.a ») -convertir le message au système binaire .	nouvelle données (D2)

Tableau 2 : Opérations Couche présentation

3.2.a L'algorithme

Cette fonction pour trouver le nombre d'utilisation du code ASCII à partir de nombre de caractères.

```
FunctionCharAscii (a : charactaire ) : entier
Declaration :
      Debut
      | Rs <- int (a) ;
     fin
```

Cette procédure pour convertir les nombresdécimauxau système binaire

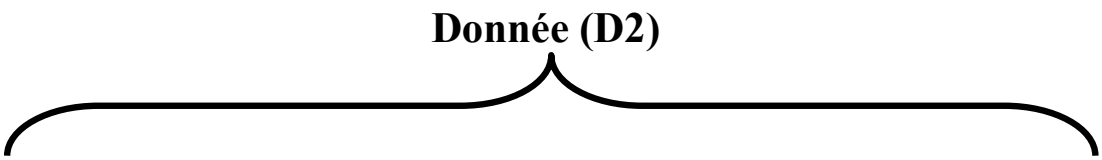
```
Procédure int2bin (cd : entier, t1[8]:element )
Déclaration :
Variable :
        i,r : entire ;
Debut
|     i←8;
|   Tanque (i>1)
|   |     r ← cd mod 2 ;
|   |   t1 [i] .ascii ← r ;
|   |   i ← i-1 ;
|   |   cd ← ( cd dive 2 )
| finTQ
Fin
```

L' algorithme suivant pour convertir le caractère au système binaire :

```
ProcedureCh2Present( Tab[N] : Char , tab2[N] : element)
Déclaration :
Variable
        i : entier ;
```

```
Début
|
|       pour (i=1 ;i<=N ;i++) faire
|       |       Tab2[i].ch← tab[I ];
|       |       Tab2[i].ascii = int2bin (tab[i]) ;
|       FinPour
Fin
```

3.2.b Représentation



En tête (SMTP)	A	0	0	0	1	0	0	1
	B	0	0	0	1	1	0	0
	…							

3.3. Couche session

	Les entres	Les opérations	Les sortie
Couche 5: Session	Données (D2)		Donnée (D2)

Tableau 3 :Opérations Couche session

La couche sessionNe pas ajouter d'informations sur les données (D3) .

3.3 Couche transport

	Les entres	Les opérations	Les sorties
Couche 4: Transport	données (D3)	- La division des données en segments, la longueur de chacun segment : 512 bit -application du protocole TCP - ajout d'entête dans chaque segment composé en : Numéro de séquence, Pointeur Numéro d'acquittement , d'urgence Option TCP(éventuelles), Remplissage longueur d'en tête, Réservé , URG, ACK , PSH,RST,SYN,FIN et Checksum (Algorithme « 3.4.a ») -deviser la donne en segments et ajouter en tête TCP	Segment

Tableau 4 :Opérations Couche transport

3.4.a L'algorithme

Cette algorithme pour deviser la donnée en segments et ajouter en tête TCP

Procedure Ch4transport (*Tab1[N] : element , Tab2[472] : element , pac : packet , seg : segment ,
ttcp : tete tcp)
Déclaration :
Variable
R , i, j : entier ;
Début
| si(N mod 472 = 0)

```
|       | R ←( N div 472)  ;
|       sinon
|       | R ← (N div 472)+1 ;
|       finsi
|
|       pour (i=1 ;i<=R ;i++) faire
|       |
|       |       pour (j=1 ; j<=472 ; j++) faire
|       |       |       Tab2[i].cases [j]← Tab1[i*j] ;
|       |       FinPour
|       FinPour
|
|      pac .tcp←tcp ;
|       pac.smtp←seg .smtp ;
|       pac .case[]←seg.cases[] ;
Fin
```

3.4.b Représentation

entête(TCP) **segments**

Port source					Port destination								
Numéro de séquence													
Numéro d'acquittement													
1	2	3	4	5	6	7	8	9					
Checksum					Pointeur d'urgence								
OpitionTCP(éventuelles)					Remplissage								

En tête (SMTP)

B	1	1	0	0	1	0	
O	1	0	1	0	0	1	
O							
N							
J							
…							

1: longueur d'en tête 2: Réservé 3: URG 4: ACK 5: PSH 6: RST

7: SYN 8: FIN 9: Fenêtre .

3.5 Couche Réseau

	Les entres	Les opérations	Les sorties
Couche 3: Réseau	Segment	-ajout d'une entête composée de: Adresse IP de L'envoyeur et Adresse IP Récepteur (Algorithme « 3.5.a »)- ajouter en tête IP	Paquet

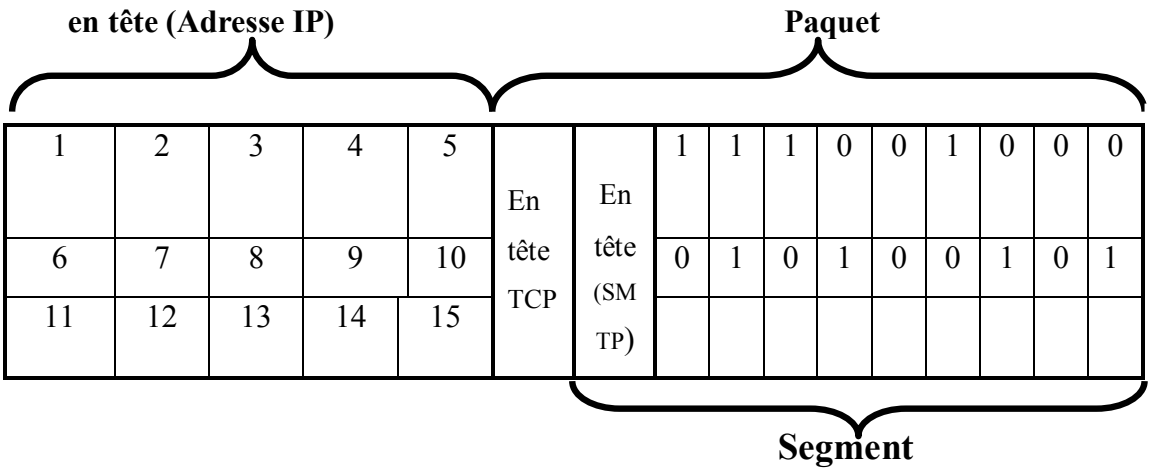
Tableau 5:Opérations Couche réseau

3.5.a L'algorithme

Cette algorithme pour ajouter l'entête IP

Procedure Ch5Resau (tra :trame , pac : packet , ip1_2 : teteip)
Déclaration :
Variable
Début
| tra.ipSR←ip1_2;
| tra .tcp←pac .tcp ;
| tra .smtp←pac.smtp;
| tra .cases[]←pac .cases[];
Fin

3.5.b Représentation



1_ Version 2_IHL 3_Type of service 4_Total length 5_Identification 6_Flags 7_Fragment offset 8_Time to live 9_Protocol 10_Header checksum 11_Source address 12_Destination address 13_options 14_Padding 15_ Longueur Total .

3.6. Couche Liaison de données

	Les entres	Les opérations	Les sorties
Couche 2 : Liaison de données	Paquet	Ajout d'une entête composéede : Fanion (Flag) Adresse MAC de l'envoyeur Adresse MAC de récepteur , FCS et Control -FCS : appliqué les opérations: LRC et CRC -Control : en appliqué le Protocole HDLC - HDLC: ajouter en tête composer en: N(S) , N(R), F|P (Algorithme «3.6.a»)- ajouter en tête couche liaison et appliquer Protocol LRC/VRC	Trame

Tableau 6 : Opérations Couche liaison de données

3.6.a L'algorithme

Cette procédure pour appliquer protocol LRC

```
Procedure lrc (tab[472]  : lement , lrc : element )
Déclaration  :
Variable
  K =0  : entier
Début
|    lrc.ch←null ;
|    pour ( i←1  a 8)
|    |          pour ( j←1 a 472)
|    |          |          si tab[j].ascii[i]←1
|    |          |                    K←k+1 ;
|    |          |finpour
|    |
|    |          si( k mod 2 )=1
|    |          |          lrc.ascii[i]= 1;
|    |          sinon
|    |          |          lrc.ascii[i]= 0;
|    |          finsi
|    |finpour
Fin
```

Cette procédure pour appliquer protocol VRC

```
Procedure vrc (tab[472]  :   element , vrc[472] : bool )
Déclaration  :
Variable
K =0  : entier
```

```
Début
|
|   pour ( i=1  a 472)
|   |
|   |       pour ( j=1 a 8)
|   |       |       si tab[i].ascii[j]=1
|   |       |               l = l+1 ;
|   |       |finpour
|   |       si( k mod 2 )=1
|   |       |       vrc [i]= 1;
|   |       sinon
|   |       |       vrc[i]= 0;
|   |       finsi
|   |finpour
Fin
```

Cette fonction Trouver l'intersection VRC et LRC

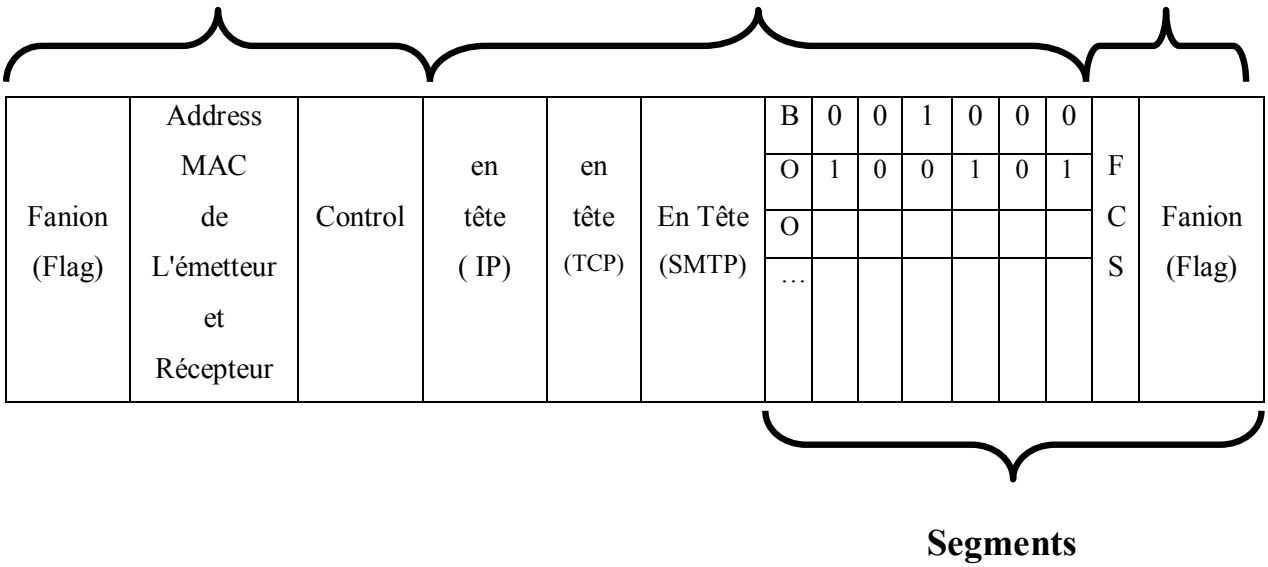
```
Functionintersect () : bool
Déclaration :
Variable
Slrc= 0 : entier ;
Svrc =0  : entier
Debut
|   pour ( i←1 a 8)
|   |       slrc←slrc + lrc.ascii[i];
|   |finpour
|   pour ( i←1 a 472)
|   |       svrc←svrc+vrc[i];
|   |finpour
|   intersect←((svrc+ src) mod  2) ;
Fin
```

Cette algorithme pour ajouter en tête couche liaison

Procedure Ch6liaison (don : final , tra : trame , f_start : tetedebut , f_end : tetefin)
Déclaration :
Variable
Début
| don.ipSR←tra.ipSR;
| don .tcp←pac .tcp ;
| don .smtp←pac.smtp;
| don .cases[]←pac .cases[];
| don.debut←f_start;
| don.fin←f_end
Fin

3.6.b Répresentation

en tête (debut) **Trames** **en tête (la fin)**

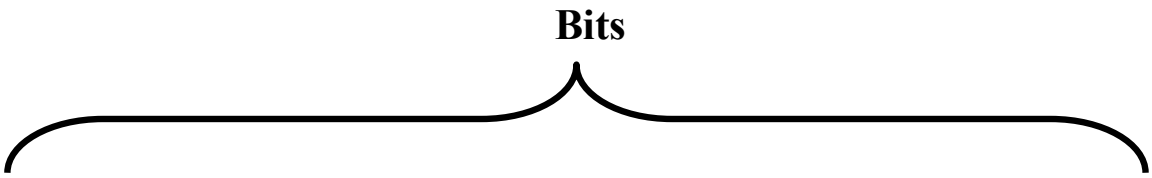


3.7 Couche Physique

	Les entres	Les opérations	Les sorties
Couche 1: Physique	trame	-Convertir toute les informations en système binaire (les bits) -conversion les bits en signaux en utilisant (code MANCHESTER)	signaux

Tableau 7 :Opérations Couche physique

3.7.b Répresentation



0	1	1	1	0	1	0	0	0	1
1	1	0	0	1	0	1	0	1	1
1	0	1	……						

3.8 Les signaux

Nous avons choisi code MANCHESTER

3.8.a L'algorithme

- Il faut 2 bauds pour coder 1 bit.

Le signal doit changer de polarité au milieu du temps bit. (Transition au milieu de chaque bit.

Les 0 sont codés par un front montant, les 1 par un front descendant)

- Transmission d'un 1 : le signal part de V1 pour finir en V0
- Transmission d'un 0 : le signal part de V0 pour finir en V1

3.8.b Représentation

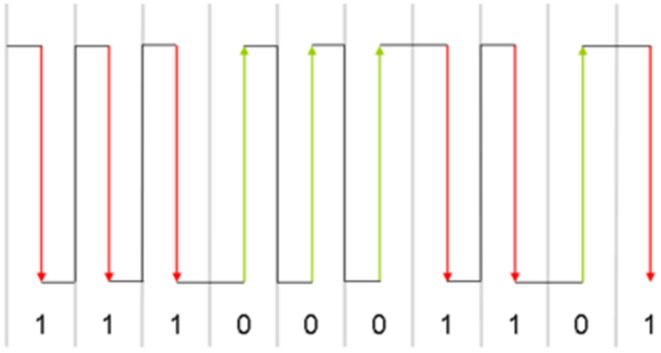


figure 11 : Un codage Manchester

4. Conclusion

A la fin de ce chapitre, nous concluons qu' il existe une encapsulation des données dans les couches du modèle OSI; et chaque couche de niveau N fourni à la couche de niveau N + 1 un service; alors chaque couche utilise ainsi les services de la couche inférieure et en fournit des service à celle de niveau supérieur.

CHAPITRE 03

Implémentation de la simulation et résultats obtenus

1. Introduction

Dans ce chapitre, consacré à la réalisation et la mise en œuvre de notre application de simulation de modèle OSI, nous allons présenter les outils de développement adoptés; soit langage utilisé qui est JAVA et l'environnement NetBeans, enfin nous montrons les principales interfaces et fenêtres de l'application

2. Environnement de travail

le langage java

On a utilisé le langage java.

Java est un langage de programmation orienté objet qui possède une grande bibliothèque qui ont des fonctions très diverses permettant de crée un code simple compréhensible et bien organisé [9]

L'EDI NetBeans

Est un environnement de développement intégré, gratuit à l'usage, se concentrant principalement sur simplifier le développement d'applications Java. Il fournit du support pour tous les types d'applications Java, depuis le client riche jusqu'aux applications d'entreprises multi-couches , en passant par les applications pour les mobiles supportant Java.

L'EDI NetBeans a une architecture modulaire qui permet les 'plug-ins'. Cependant, l'étendue des fonctionnalités dans l'installation de base est tellement riche que vous pouvez probablement utiliser l'EDI pour votre travail sans du tout vous souciez des modules externes [24].

3. *Résultats de la simulation*

Dans cette partie, nous allons montrer l'interface de la simulation et expliquer son déroulement:

3.1 L'onglet du simulateur : Lorsque on exécute le logiciel , la fenêtre suivant s'affiche

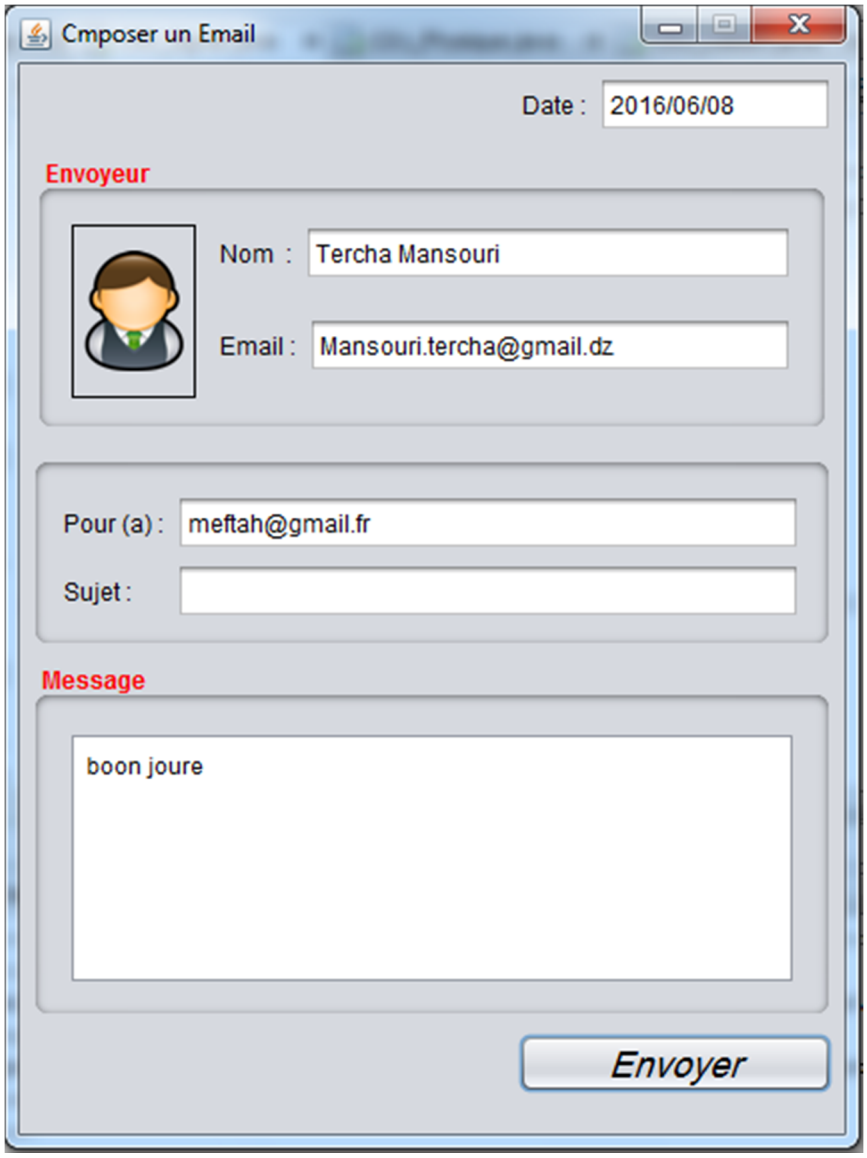


Figure -12- L'onglet du simulateur

Pour envoyé un message entre l'émetteur et le récepteur L'utilisateur entre tout d'abord insérer les paramètres de

- **L' émetteur**

✓ Nom

✓ Email

- **Le récepteur :** Email
- **Message** :

✓ le sujet

✓ le contenu

puis en cliquant sur "**Envoyer**".

Après cela, classement des bouton des 07 couches le modèle OSI , En plus **l'adresse ip** et **l'adresse mac** et **port** d' émetteur et de récepteur

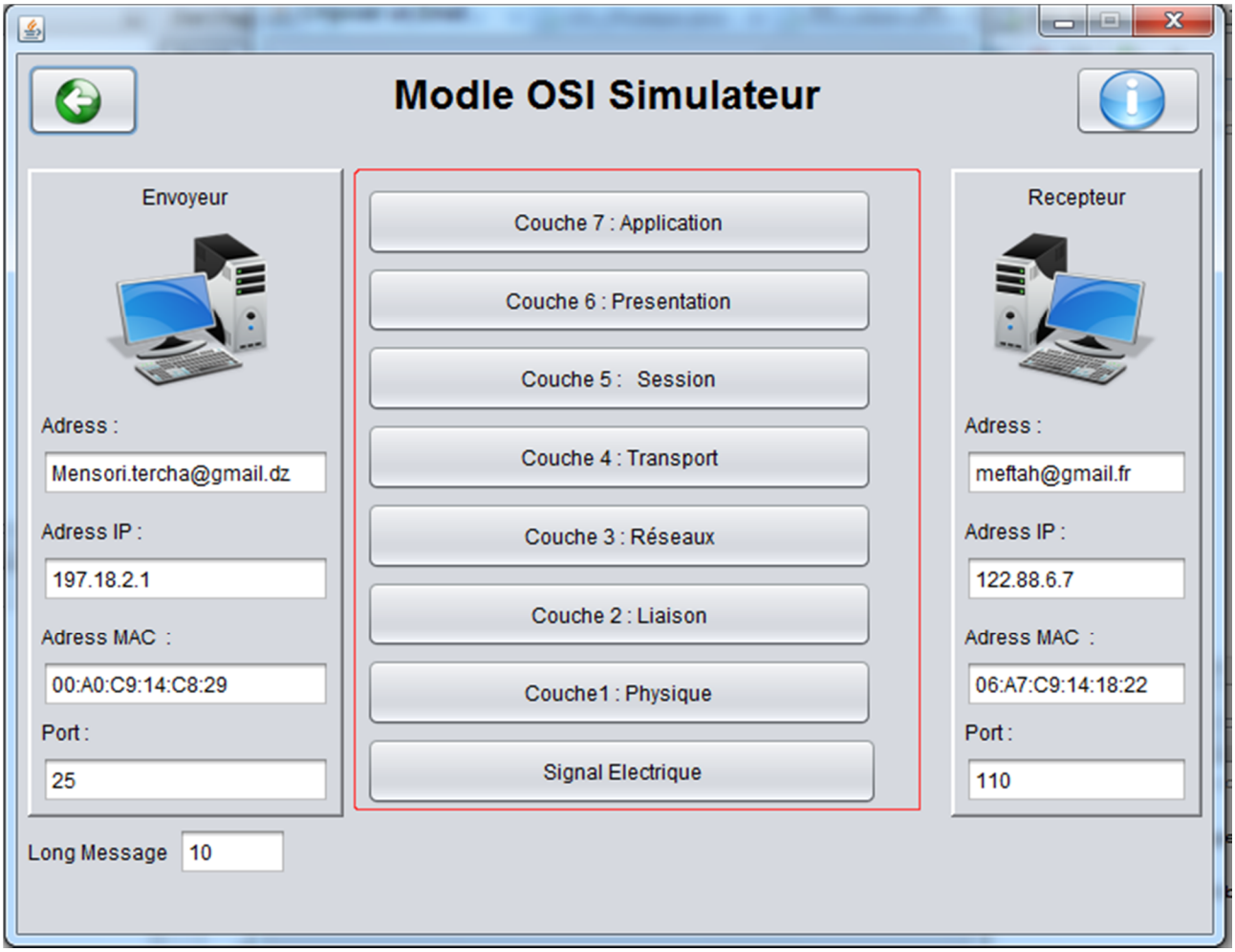


Figure -13- interface de 07 couches

Pour informer les opérations au message sur chaque couche on clique sur le bouton de chaque couche, alors pour premier couche :

3.2 Le Couche application

- ✓ Stocker le message dans un tableau et applique le protocole SMTP , ajouter l'entéte de smtp : Return-Path , FromTo , Received ,Subject , MIME-Version ;Content-Type , charset , Transfer ; Message-ID; Date
- ✓ UDP = Donné
- ✓

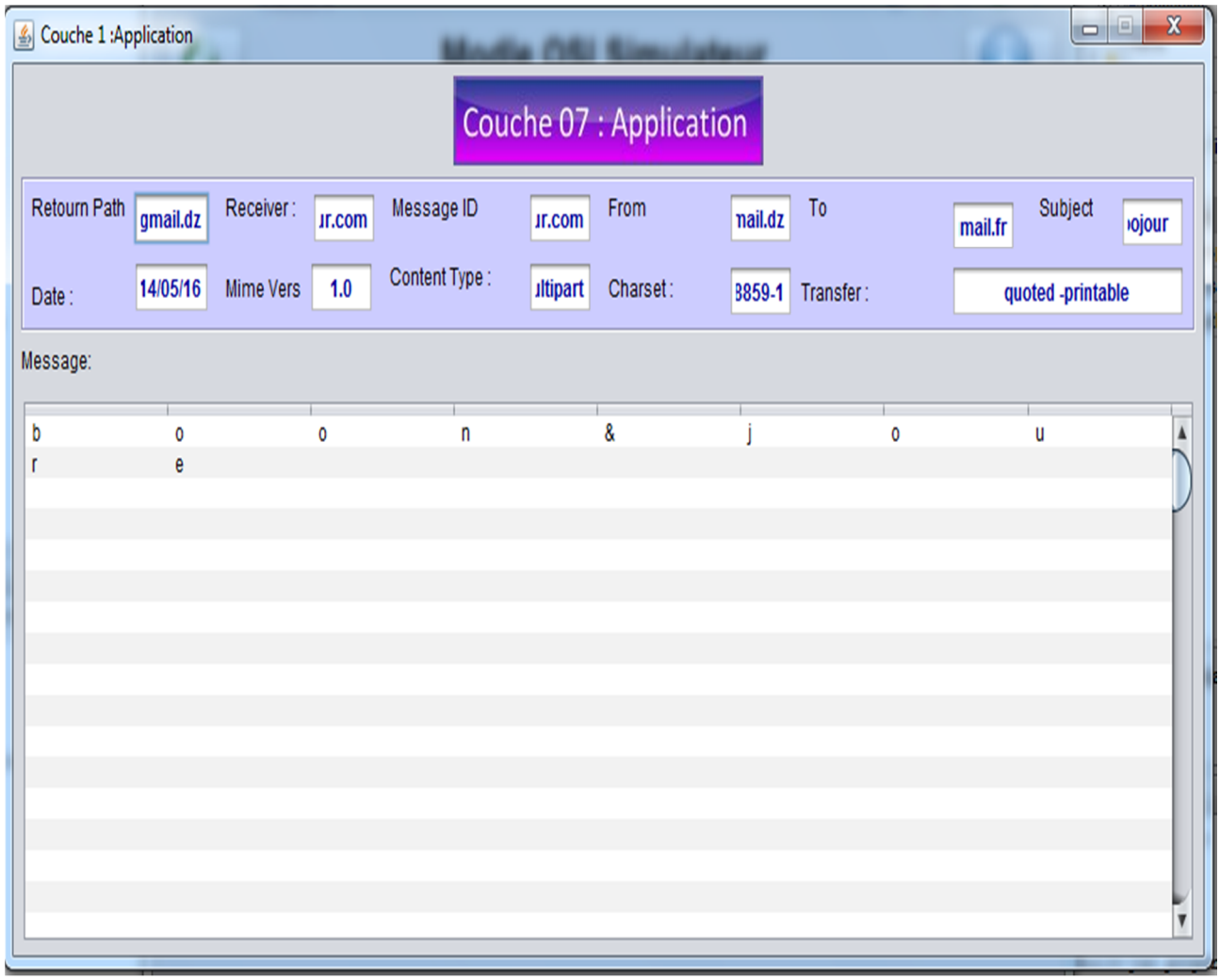


Figure -14- interface de couche application

3.1. Le Couche présentation

- ✓ encoder le texte en code ASCII puis en binaire , alors dans notre simulation chaque lettre offre son code binaire , comme suivant:
- ✓ UDP = Donné

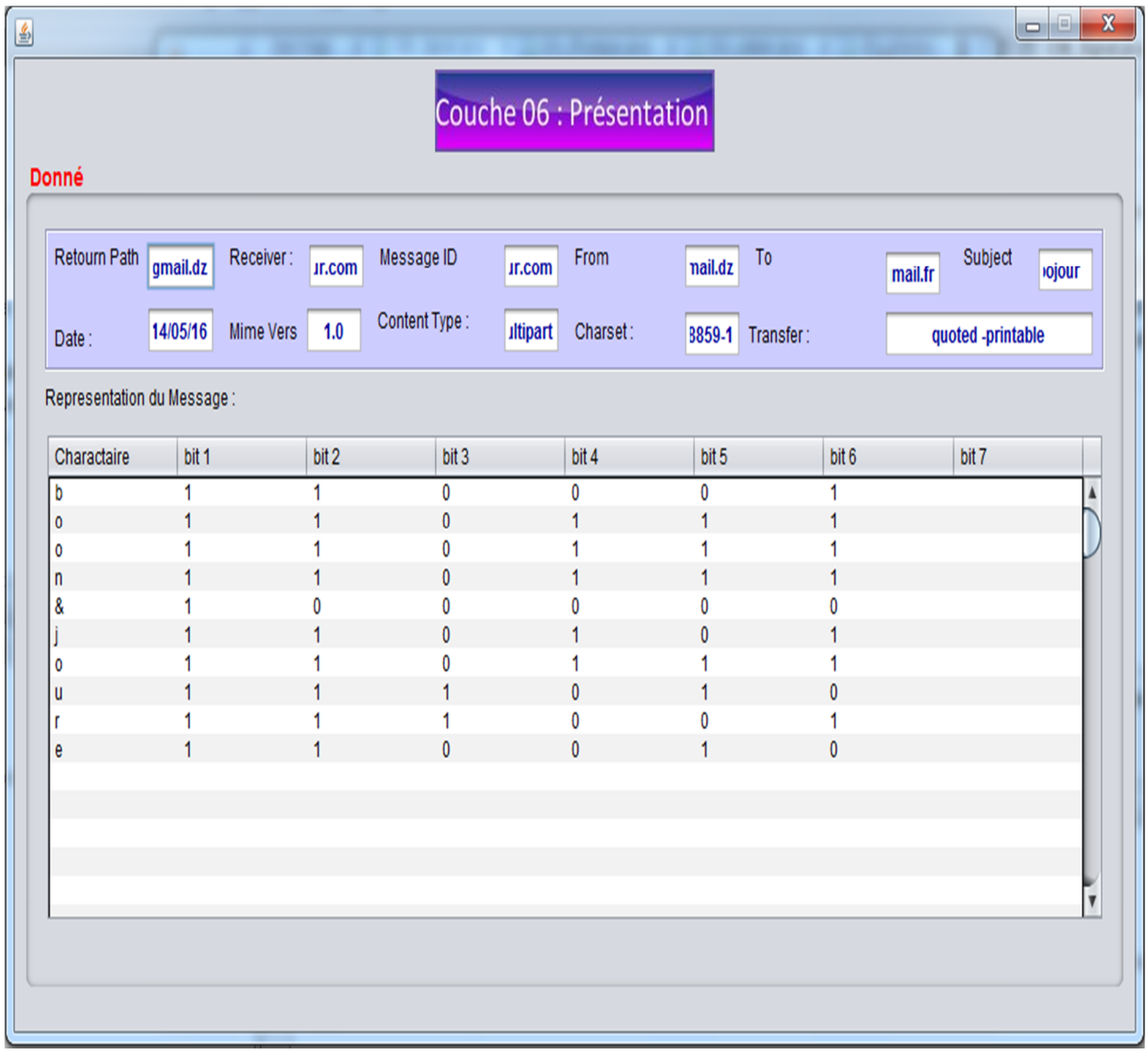


Figure-15- interface de couche présentation

3.2. Le Couche transport

Dans la couche transport : la protocole TCP

- ✓ diviser le donnée en segments , chaque segment a la taille de 512 octet
- ✓ ajouter l' en-tête TCP : d'urgence , OpitionTCP , Remplissage , Réservé , ACK PSH , RST, SYN[1] , FIN , Checksum , Pointeur d'ur , Portsour, Portdes, NuméroS , Numérod'ac , longueur . comme suivant :
- ✓ UDP = segement

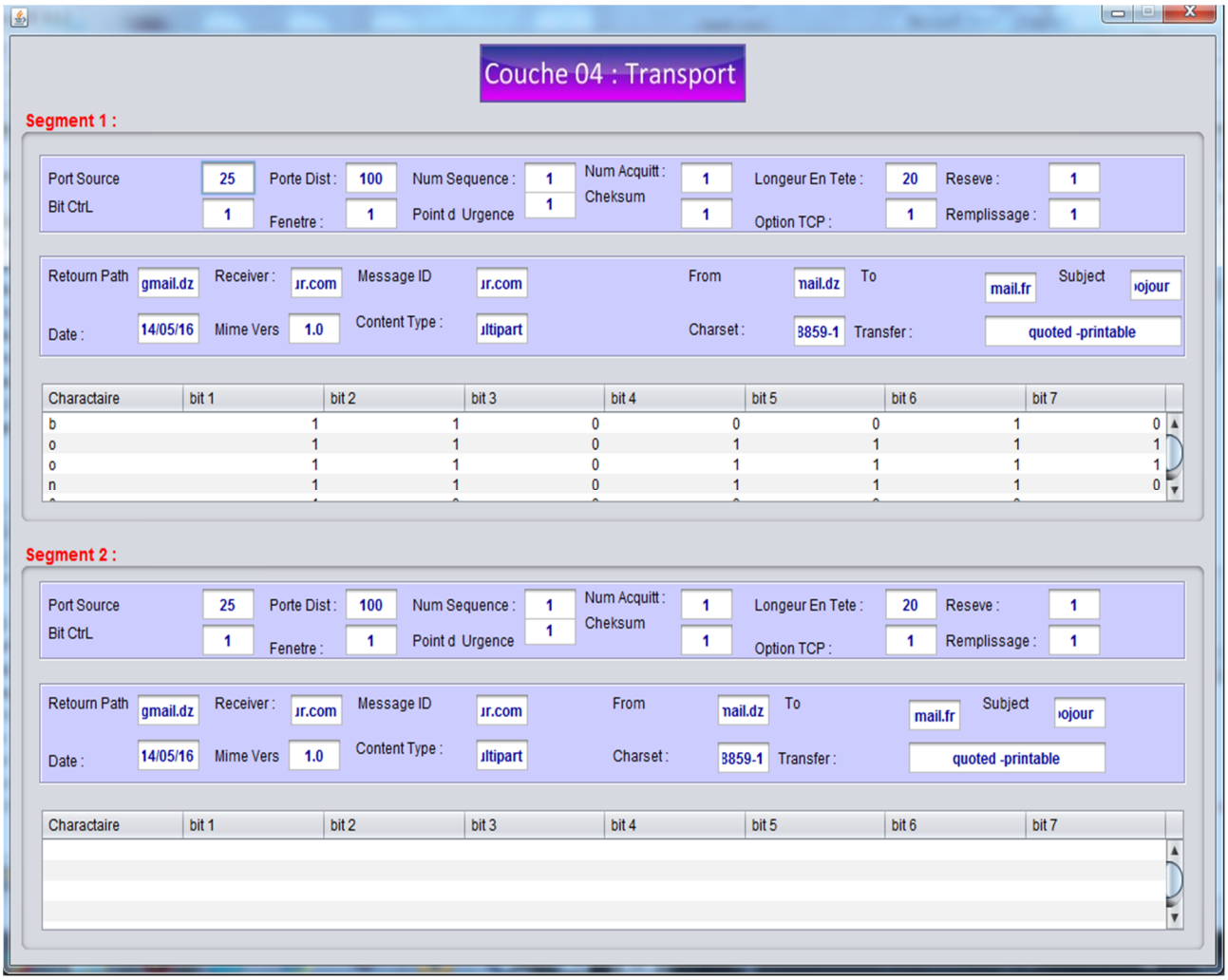


Figure-16- interface de couche transport

3.3. Couche réseau

- ✓ Protocol IP ajouter à le segement l' en-tête suivant:

  Header checksum, Source address, Destination , address , IHL , Type of service, Total length , Identification , Flags , Fragment offset , Time to live Padding , Version ,Longueur Totale;

- ✓ UDP = Packet

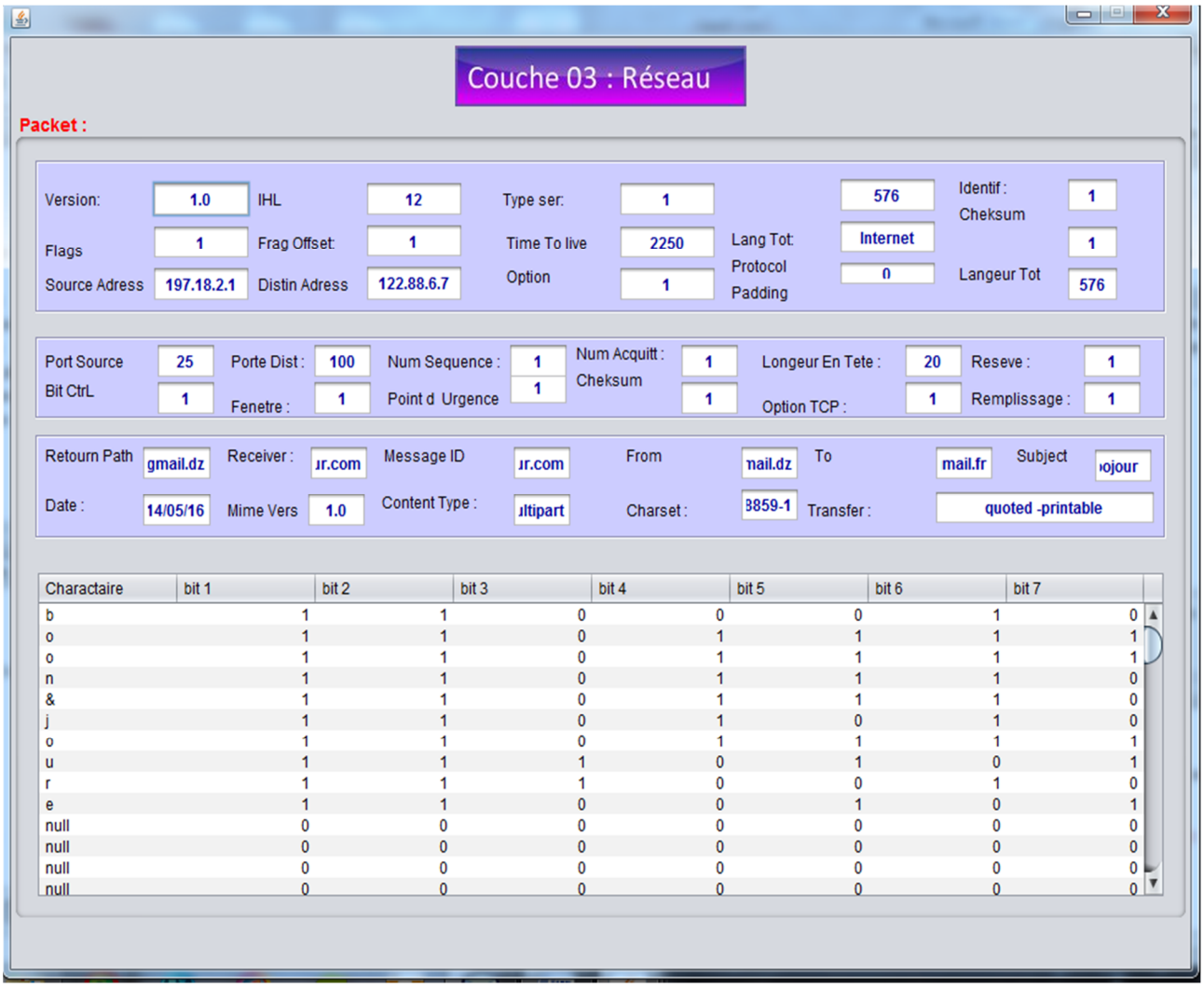


Figure-17- interface de couche réseau

3.4. Couche liaison de donnée

- ✓ La protocole HDLC ajoute au packet l' entête suivant : Control , Mac adresse d' émetteur et récepteur ; Fanion , fcs .
- ✓ UDP = trame

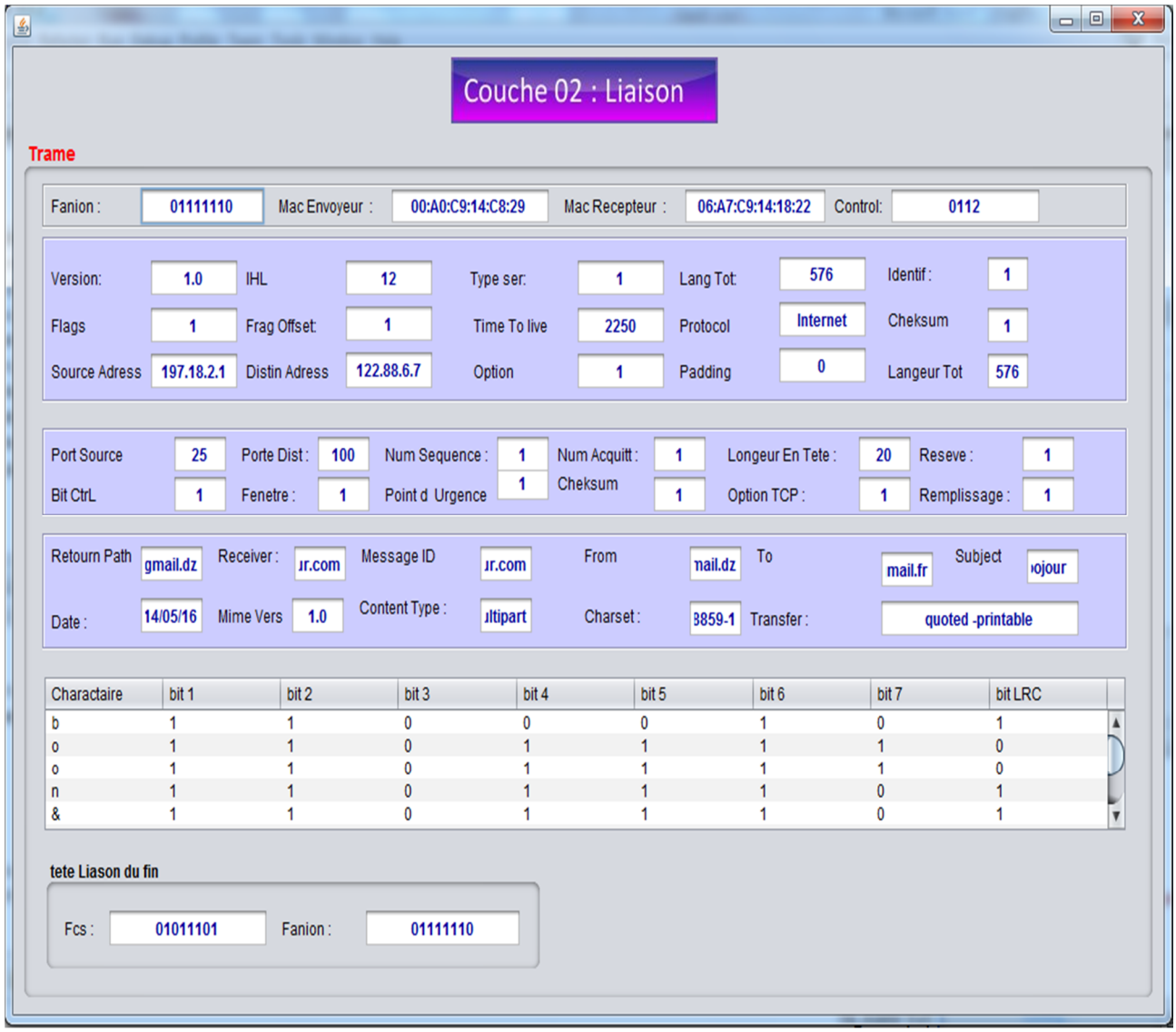


Figure-18- interface de couche liaison

3.5. Couche physique

- ✓ Dans la dernier couche on donne le transforme à un signale pour envoyer par les support de transmission , ce signale traduit comme un codage selon le 0 et 1, nous avons utilisé le coda Manchester en exemple
- ✓ UDP = bit

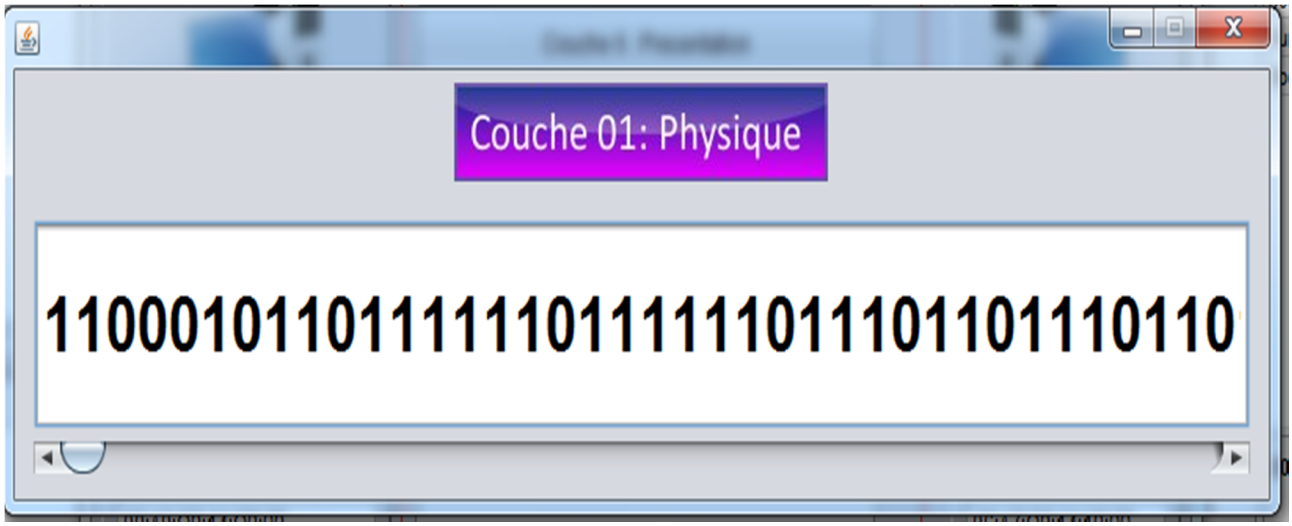


Figure-19- interface de couche physique

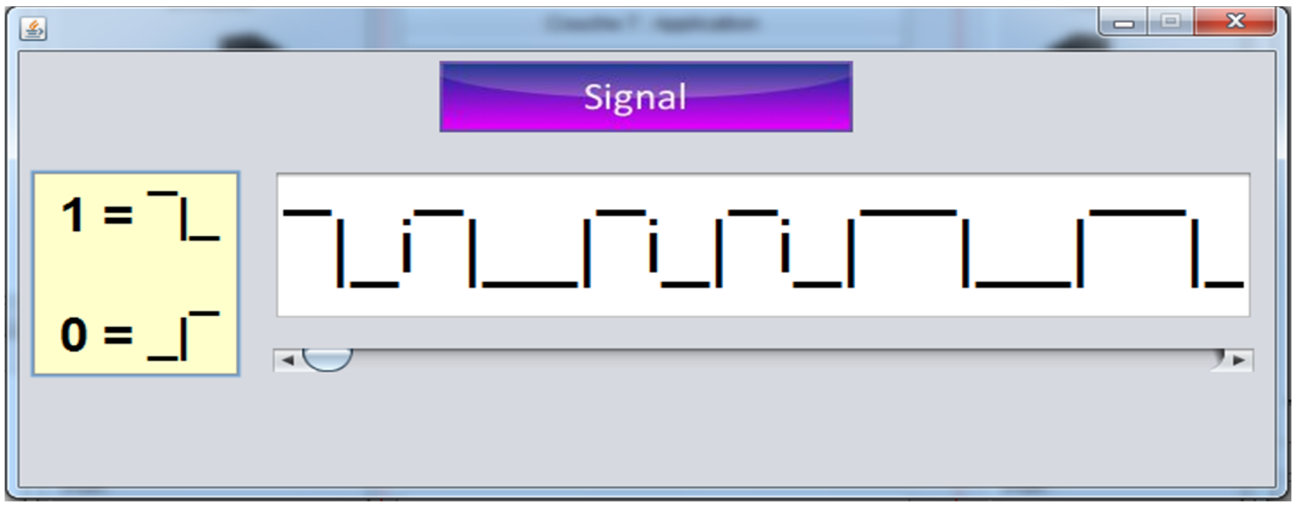


Figure-20- interface de signale

4. Conclusion

Dans ce chapitre , on a présenté les différentes fonctionnalités de notre application, le simulateur graphique de modèle OSI , son intérêt réside dans la mise à disposition des étudiants en réseaux, un outil pédagogique, pratique et clair, qui leur permettre de se familiariser avec ce modèle, de faciliter la compréhension et l'assimilation de mécanismes souvent compliqués , et aussi de rendre les cours et les travaux pratiques concernant ces 07 couches de modèle OSI plus pratiques et plus agréables .

Conclusion générale

L'objectif de notre travail était la construction d'une application d'un simulateur des couches réseau (Modèle OSI)

Nous avons commencé notre travail par une étude générale sur le modèle OSI pour avoir une vue globale a les 07 couches . Par la suite, nous avons présenté la conception de notre application par les algorithme de chaque couche . Le dernier chapitre présent l'application du système étudié ou nous avons traduit notre modélisation conceptuelle en une implémentation(simulateur).

On a implémenté un simulateur graphique qui facilite l'apprentissage et l'enseignement pour les apprenants et pour les enseignants. On a utilisé langage JAVA et l'environnement netbeans .

le thème de notre projet est un sujet très grand , Pour cette raison dans notre application nous avons études que le contenu du message est un texte , et nous avons étudié les processus qui se produisent sur le message exactement , alors peut être dans les années à venir pour les étudiants de compléter ce sujet tournant vers d'autres types de messages, tels que la voix ou de l'image, la vidéo . et étude les processus des en-têtes de chaque couche .

Bibliographie

Livres électroniques

[01] Emmanuel Vien , Réseaux Module , 2012-2013 .

[02]Sylvain , RESEAUX le modèle, Le 02 mai 2003.

[03] Laurent BAYSSE , Le protocole SMTP, du 6 mars 2005 .

[04] Philippe Latu , Modélisations réseau.

[05] Philippe Latu , Transmission de l'information &protocoles Internet.

[06] Jean-Yves Didier , Introduction aux réseaux.

[07] Degouet Faien et Desgeorge Guillaume et corbel Steve, synthése de protocoles courants , 1998/1999.

[08]Bouabid El Ouahidi , Transmission , 2009 .

[09] Far Hadjar et Ghilani Sabrina; Un simulateur graphique de protocole de liaisons de donnée dédié a l'apprentissage et l'enseignement ; /06/2013; Mémoire MASTER ACADEMIQUE, Ouargla .

[10] Fran¸cois Laissus , Cours d'introduction `a TCP/IP , Version du 25 f´evrier 2009

[11] Dr. Abdelhamid DJEFFA , cour Réseaux Informatiques , 2015/2016

[12] Laure Petrucci , Cours de Réseaux ,7 septembre 2012.

[13] Omar Cheikhrouhou , le protocole hdlc, Cours d'Olivier Glück .

[14] Abdelhamid-djeffal , cours_rc_physique .

[15] Rziza Mohammed , Cours des réseaux Informatique (2010-2011).

Web électroniques

[16] TCP/IP, URL : http://www.commentcamarche.net/contents/539-tcp-ip. consulté le 22/02/2016

[17] Transmission de données - Les modes de transmission ,URL : http://www.commentcamarche.net/contents/1131-transmission-de-donnees-les-modes de transmission] consulté le 09/05/2016 .

[18] http://www.rfai.li.univ-tours.fr/PagesPerso/jyramel/fr/1_osi consulté le 22/03/2016.

[19] Définition des sept couches du modèle OSI et explication de leurs fonctions

[20] PROTOCOLES ET ARCHITECTURES ;URL : http://www.teluq.uquebec.ca/expl_inf5004/consulté le 02 /03/2016.

[21] https://docs.oracle.com/cd/E19957-01/820-2982/ipov-32/index.html/consulté 05/03/2016.

[22] https://lipn.univ-paris13.fr/~manzonetto/~M1104/M1104_Chap3 consulté le 12/03/2016.

[23]http://dirac.epucfe.eu/projets/wakka.php?wiki=OsI&show_comments=1/consulté 23/02/2016.

[24] http://www.netbeans.org/community/kb/index.html consulté le 01/04/2016.