2/04

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Electronique

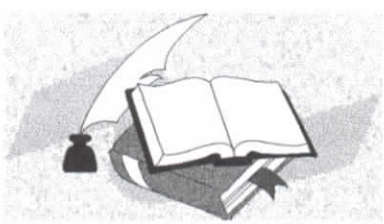
Projet de fin d'étude

Thème :

Interface Ethernet pour le 68HC11 : Application au contrôle d'accès

Proposé et dirigé par :
M^r^ R.SADOUN

Étudié par :
M^r^ K.BELDJATIT



Promotion 2004

E.N.P. 10, Avenue Hassen-Badi, El Harrach, ALGER

ملخص

هذا العمل يتمثل في انجاز نظام مراقبة المرور باستعمال نموذج Client / serveur . هذا الاخير يقوم بالتحقق من هوية الاشخاص .الهدف المقصود هو بساطة المخطط،اخترنا انجاز واجهة ترتكز على microcontrôleur 68HC11 مقترن مع الدارة ethernet CS8900

كلمات المفاتيح: Client/Serveur، مراقبة المرور، 68HC11، CS8900.

Résumé

Notre projet rentre dans le cadre de la conception d'un système de contrôle d'accès distribué implémentant le modèle client serveur. Ce dernier sera dédié à la vérification de l'identifiant d'une personne donné. Le but recherché étant la simplicité et la souplesse de l'identificateur, nous avons opté pour la conception d'une interface basée sur un microcontrôleur 68HC11 couplé à circuit ethernet CS8900.

Mot clés : Architecture Client/Serveur , Contrôle d'accès, 68HC11,CS8900.

Abstract

This project enters in the framework of the design of a distributed access security system implementing the client / server model. This latter is bounded to a person's identity verification. Our aim is to achieve an identifier simplicity and flexibility, the design interface was based on a 68HC11 micro controller coupled to a CS8900 ethernet circuit.

Key words : Client/ Server model, Access security, 68HC11, CS8900.

Dédicace

Je dédie ce travail :

♣ A mes chers parents et grand parent.

♣ A mes frères et sœurs.

♣ A tous mes amis et collègues d'étude.

Remerciement :

Tous d'abord, je remercie mes parents, qui m'ont donné le courage et la foie, et qui étaient toujours là pour moi.

Mes remerciements vont à Monsieur R. SADOUN, qui m'a proposé le sujet et m'a accompagné avec une grande patience.

Je remercie tous les enseignants de Polytechnique qui m'ont aidé, et encouragé.

En fin, je tiens à remercier tous mes amis de L'ENP, ceux de la cité BURAOUI Amar, surtout Rachid .

SOMMAIRE

IV. ETHERNET

V. DEVELOPPEMENT

INDEX DES FIGURES:

Introduction générale

La sécurité dans les organismes modernes est devenue une nécessité et une préoccupation majeure. Beaucoup de solutions ont été proposées; chacune ayant des inconvénients et des avantages. L'approche que nous avons abordée va dans le sens d'un système distribué d'un contrôle d'accès ; le point d'accès à piloter devenant autonome et surtout communicant à travers un réseau local.

Afin d'accorder les droits nécessaires à chaque personne d'accéder à un lieu donné, il doit être menu d'un identificateur permettant de le reconnaître. Chaque identificateur transmet l'identifiant vers un serveur chargé de la vérification de l'identité et de donner la réponse.

L'architecture nécessaire pour ce genre de protocole est une architecture client-serveur appliquée à un réseau Ethernet. Pour pouvoir transmettre les données entre client et le serveur, il nous faut une adaptation au support de transmission. Cette étape passe par l'encapsulation des données et construction des trames. Cette dernière se fait à l'aide d'un circuit Ethernet géré par un micro-contrôleur; le tout est appelé une carte d'interface.

Nous avons scindé notre développement en quatre étapes :

- la première traite du contrôle d'accès
- la seconde aborde le modèle client serveur adapté dans notre cas au modèle que nous avons choisi à savoir un système de contrôle d'accès distribué
- le support technologique choisi étant ethernet, nous avons consacré la troisième partie à sa présentation
- les derniers chapitres ont été dédiés aux développements que nous avons effectués

I. CONTRÔLE D'ACCES

1. Introduction

Le contrôle d'accès a pour fonction de demander à toute personne, entrant dans un lieu sous contrôle, de s'identifier. Ce lieu peut être un immeuble, une propriété, une salle, un sous-sol... délimité et mis sous surveillance.
Les techniques utilisées pour le contrôle d'accès permettent d'identifier la personne voulant accéder à ce lieu, en comparant les données de son laissez-passer à celles de référence. Ce laissez-passer peut être un badge, une carte, un support perforé, etc., dont les données sont identifiées par un lecteur ou un capteur, autonome ou centralisé(exemple : serveur d'un réseau).
Les systèmes de contrôle d'accès peuvent autoriser les accès sur la base de quatre principes permettant normalement une sécurité croissante :

- présentation d'un objet que l'on a, et qui est reconnu par le système (par exemple : un badge).
- expression de quelque chose que l'on sait (par exemple : entrer un code confidentiel).
- réalisation de quelque chose que l'on sait faire (par exemple : signer son nom avec une succession de gestes précisément définis, dire une phrase avec une intonation et une voix précise...).
- présentation d'une partie du corps, qui sera reconnue par le système.

Les systèmes de contrôle d'accès des deux dernières catégories sont dits biométriques. Les systèmes de contrôle d'accès combineront éventuellement plusieurs de ces principes. Par exemple un contrôle d'accès biométrique pourra reposer sur la présentation d'un badge, puis de la main.
D'autres services sont de plus en plus souvent associés au contrôle d'accès : gestion du temps de travail, de services communs...
Longtemps, pour contrôler les accès, pour n'importe quel site, l'homme a prévalu sur la technique comme : garde, vigie, sentinelle... Aujourd'hui, c'est l'inverse : la technologie, de plus en plus élaborée au fil des ans, s'impose à l'individu. L'image du gardien, filtrant les visiteurs d'un coup d'oeil sur leur badge ou leur carte d'accès devenant de plus en plus révolu.

2. Le contrôle d'accès

Il participe totalement à l'organisation générale de la sécurité-accès, au sens large du terme. Il a deux fonctions majeures :

- filtrer les personnes et véhicules qui passent.
- gérer ces passages selon une hiérarchie (en fonction des pièces et étages auxquels ces personnes peuvent accéder) pour en garder une trace afin d'établir des statistiques ou, après une intrusion, pour en analyser et comprendre les causes.

Sans trace, en effet, on ne contrôle pas; on se contente de filtrer les passages.

3. Usages de contrôle d'accès

Il ne se limite pas à autoriser ou non une personne ou un véhicule à pénétrer dans une enceinte ou un lieu déterminé. Ses fonctions s'étendent :

- au contrôle d'horaires variables.
- à des systèmes de paiement interne (cantine, restaurant d'entreprise, par exemple).

- à la gestion d'un parc automobile ou encore d'autres services communs à des utilisateurs donnés.

Aussi, il n'y a pas un secteur d'activité dans lequel le contrôle d'accès ne soit utilisé : industries, entreprises, immeubles d'habitation, communes, administrations, hôpitaux, aéroports, etc.

4. Mise en place

Pour se préserver de nombreux actes de malveillance, le contrôle d'accès est tout indiqué : avec l'anti-intrusion, la vidéosurveillance, la télésurveillance, il représente l'un des systèmes constitutifs de la protection et de la sécurité des personnes.
Pour mettre en place un système de contrôle d'accès, il faut :

- de la méthode.
- mais aussi beaucoup de psychologie.

On peut distinguer quatre grandes étapes :

- la prise de décision.
- la définition du concept retenu.
- le choix de la technologie la plus adaptée.
- la rédaction du cahier des charges.

5. Les différentes technologies de l'identificateur

5.1. Le magnétisme

Trois technologies sont fondées sur le magnétisme et peuvent prétendre à l'appellation de cartes magnétiques :

5.1.1. La carte à piste magnétique

Doit son nom à sa bande du type bande de magnétophone collé sur une carte plastique au format ISO. Au lieu d'un enregistrement musical ou vocal, l'information numérique est reconnue par une tête magnétique qui sait la déchiffrer .Cette technologie a pour elle son faible coût relatif à sa compatibilité de lecture avec beaucoup de systèmes existants, sa normalisation et sa grande diffusion. Elle est en revanche, sensible aux champs magnétiques, à la poussière. Sa durée de vie est donc relativement limitée dans le temps car les lecteurs, en général, usent les pistes de technologie standard **[-5-]**.

5.1.2. La carte à induction baryum-ferrite

Une technologie de moins en moins employée. Cette carte contient un certain nombre d'aimants qui, suivant leur orientation, représentent la valeur binaire 0 ou 1. Par association de ces valeurs en quartets, on code les chiffres 1 à 9 pour former un nombre qui peut être reconnu par une matrice de bobines. Peu utilisée pour des grandes configurations, cette technologie, par sa simplicité, représente pas mal d'avantages dont le coût et l'insensibilité aux chocs et aux rayures.

5.1.3. La carte magnétique Watermark

Il s'agit d'une carte à 2 pistes magnétiques superposées encodables exclusivement par le fabricant. Non reproductible et infalsifiable.

5.2. La puce

C'est l'avenir. La carte à puce contient un circuit intégré qui est soit une simple mémoire de type REPROM ou PROM, soit un microprocesseur avec mémoire. Dans tous les cas on peut stocker une information avec un niveau de sécurité très élevé puisqu' on peut imaginer des algorithmes d'encryptage très poussés grâce au microprocesseur. On peut également, sur ce type de carte,

modifier la valeur du contenu. Il s'agit là d'une technologie très évoluée, multifonction et qui offre une capacité de mémoire importante. En outre, la carte à puce permet le transport d'informations et l'accès aux services liés à la monétique. En contrepartie, son coût est en rapport avec sa sophistication et le support reste sensible aux chocs, au pliage et à la torsion.

5.3. Le code barres

Peu utilisé en contrôle d'accès .La technologie du code barres est aujourd'hui très répandue, ne serait-ce que dans la grande distribution. Le principe est simple : il s'agit de juxtaposer des barres claires et sombres, de largeur variable [-6-]. Un procédé optique mesure l'épaisseur de chaque « trait » ainsi que leur nombre et leur répartition linéaire. Il existe, dans ce type de technologie, plusieurs méthodes de codification : l'EAN 13, le code 39, le 2 parmi 5, etc. Cette technologie a pour elle son coût, son insensibilité aux champs magnétiques et aux chocs et rayures. En revanche, à défaut de masquage, sa copie est possible. C'est également une technologie figée ce qui explique qu'elle soit peu utilisée en contrôle d'accès.

5.4. La radio

L'étiquette radio est très peu utilisée pour les personnes mais comme antivol d'objets (vêtements, etc.). Placée dans un champ d'émission, celle-ci émet ses caractéristiques. Si l'identifiant possède sa propre source d'énergie, on parle de système actif et sa durée de vie est en rapport avec celle de sa pile. Dans le cas contraire, il est passif. Cette technologie a l'avantage d'être réfractaire à l'usure et de présenter un fort potentiel de sécurité. Son coût et sa mise en oeuvre sont en rapport.

5.5. La perforation

Dans l'hôtellerie et les laveries automatiques principalement, et dans certaines entreprises, pour les visiteurs. Mais elle offre une très faible sécurité. Le badge possède une grille de perforations 5x6 codée correspondant à son numéro. Un contrôle de parité augmente le degré de sécurité de lecture. Lors de l'introduction du badge entre une série de diodes émettrices et réceptrices, le numéro du badge est identifié selon le nombre et la position des perforations dans la grille 5x6. La codification est facile et insensible à la saleté, à l'environnement et aux influences électromagnétiques. Cette technologie est peu coûteuse et pratique, mais copiable.

5.6. Biométrique

La biométrie est la technologie de capture et de stockage des caractéristiques uniques des êtres humains afin de les identifier et de vérifier leur identité.
Pour les systèmes de contrôle d'accès utilisés dans la biométrique on trouve:

1. Reconnaissance faciale
2. Emprunte digitale
3. Morphologie de la main
4. Œil : Iris et Rétine

Les caractéristiques biométriques, sont de plus en plus déployées pour l'authentification de l'utilisateur dans beaucoup d'applications vitales.
A l'inverse de l'authentification basée sur la connaissance de l'utilisateur, comme un code PIN (Personal identification number), mot de passe, cartes à puce ; les systèmes biométriques se basent sur des caractéristiques physiques uniques et propres à la personne.
Avec l'authentification les codes PIN, mots de passe et clefs peuvent être oubliés, perdus ou volés, avec la biométrie toutes les données de l'utilisateur sont conservées.
L'anatomie de l'utilisateur devient la signification de l'identification, le mot de passe biologique.
L'authentification biométrique de l'utilisateur améliore le rendement des systèmes de sécurité et

facilite son utilisation, dans la mesure où les utilisateurs n'ont plus besoin de mémoriser des PIN et mots de passe.

6. Infrastructures nécessaires

Pour permettre la gestion des lecteurs dans les portes, un réseau informatique est nécessaire, ainsi qu'un ensemble de serveurs.
Le système créé utilise l'informatique dans toute sa dimension: de l'identificateur, en passant par le lecteur, comprenant un système microprocesseur(microcontrôleur) relié par un réseau local (Ethernet)à des concentrateurs constitués de PC, eux-mêmes interconnectés aux serveurs des bases de données et aux postes des gestionnaires.

Au niveau e l'identificateur, l'information obtenue par celui-ci, sera traitée par une application stockée dans une mémoire reliée avec le micro-contrôleur qui permet la transmission de cette information sous la forme de paquet vers le serveur qui va lui donner la décision du faire l'ouverture ou pas.
Au niveau de serveur, notre application aura la mission de faire la vérification d'authentification en faisant une comparaison par le contenue de la base de donnée qui contient un ensemble de fichier spécifique à chaque personne.

7. Conclusion

Nous avons vu que le système de contrôle d'accès est un aspect nécessaire dans des endroits à doter de moyens de sécurité.
On peut rencontrer dans ce domaine plusieurs technologies permettant une diversité de choix d'identificateurs. Pour faire la conception d'un tel système il nous faut une architecture contenant les deux éléments principaux suivant :
- Un identificateur de la personne.
- Un serveur de base de données enregistrant et autorisant éventuellement l'accès.

Cette architecture rassemble beaucoup plus aux modèles client/serveur qui sera abordée dans le chapitre suivant.

I I. ARCHITECTURE CLIENT/SERVEUR

1. Introduction

L'architecture client/serveur sera le sujet de ce chapitre ; une présentation des concepts de base, ainsi qu'une idée générale sera donnée sur cette architecture.

2. Présentation de l'architecture d'un système client/serveur

Les serveurs permettent la mise à disposition, et la gestion, des ressources communes aux stations de travail. Chaque serveur dispose en principe d'une mémoire de masse importante, qui contient tout ou partie des fichiers et programmes communs.
Chaque demande d'utilisation d'une des ressources communes que gère le serveur doit être reçue, traitée, puis une réponse doit être donnée à la station demandeuse (client). Une architecture client-serveur peut être définie comme une architecture logicielle fournissant des services distants (bases de données, messagerie, impression, ...) à des clients qui peuvent utiliser, de manière transparente, l'ensemble des ressources informatiques mises à leur disposition.
Selon la répartition des données et des traitements entre la station cliente et le serveur, on considère que l'on évolue d'un client-serveur propriétaire (cas de la présentation distribuée où le serveur gère quasiment tout) à un client-serveur hétérogène où données et applications sont réparties.
Un serveur peut n'être affecté qu'à son strict rôle de serveur, on dit alors qu'il s'agit d'un serveur dédié, ou bien il peut être à la fois serveur et station de travail.

3. Avantages de l'architecture client/serveur

Le modèle client/serveur à les avantages principaux sont:

- des ressources centralisées: afin d'éviter les problèmes de redondance et de contradiction, le serveur est au centre du réseau ,il peut gérer des ressources communes à tous les utilisateurs, comme par exemple une base de données centralisée.
- la sécurité: car le nombre de points d'entrée permettant l'accès aux données est moins important
- un réseau évolutif: grâce à cette architecture il est possible de supprimer ou rajouter des clients sans perturber le fonctionnement du réseau

4. Inconvénient de l'architecture client/serveur

L'architecture client/serveur a tout de même quelques lacunes parmi lesquelles:

- un coût élevé : dû à la technicité du serveur
- un maillon faible: grâce à la grande probabilité que le serveur être en panne, le serveur est le seul maillon faible du réseau client/serveur.

5. Fonctionnement d'un système client/serveur

Le modèle client/serveur repose sur une communication d'égal à égal entre les applications, cette communication réalisée par dialogue entre le client et le serveur forme un système coopératif.
Le résultat de cette coopération se traduit par un échange de données, le client réceptionne les résultats finaux délivrés par le serveur.

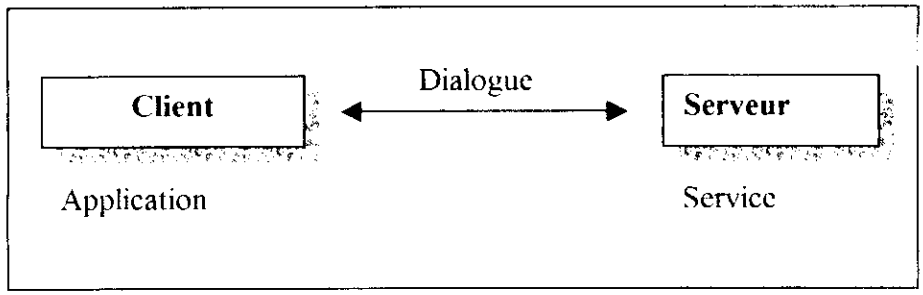


Figure 2.1: Système client/serveur

- Le client émet une requête vers le serveur grâce à son adresse et le port, qui désigne un service particulier du serveur.
- Le serveur reçoit la demande et répond à l'aide de l'adresse de la machine client et son port.

6. La notion de Port

De nombreux programmes TCP/IP peuvent être exécutés simultanément sur Internet (vous pouvez par exemple ouvrir plusieurs navigateurs simultanément). Chacun de ces programmes travaille avec un protocole, toutefois l'ordinateur doit pouvoir distinguer les différentes sources de données.
Ainsi, pour faciliter ce processus, chacune de ces applications se voit attribuer une adresse unique sur la machine, codée sur 16 bits: un port (la combinaison adresse IP + port est alors une adresse unique au monde, elle est appelée socket).
Le numéro de port indique l'application à laquelle les données sont destinées .S'il s'agit d'une requête à destination de l'application, l'application est appelée application serveur. S'il s'agit d'une réponse, on parle alors d'application cliente.
Il existe des milliers de ports (ceux-ci sont codés sur 16 bits, il y a donc 65536 possibilités), c'est pourquoi une assignation standard a été mise au point, afin d'aider à la configuration des réseaux.
Voici certaines de ces assignations par défaut:

Port	Service ou Application
21	FTP
23	Telnet
25	SMTP
53	Domain Name Server
63	Whois
70	Gopher
79	Finger
80	HTTP
110	POP3
119	NNTP

Figure 2.2 : Quelques numéros des ports usuels

- Les ports 0 à 1023 sont les ports reconnus ou réservés (Well Known Ports). Ils sont assignés par le IANA (Internet Assigned Numbers Authority).
- Les ports 1024 à 49151 sont appelés ports enregistrés (Registered Ports).
- Les ports 49152 à 65535 sont les ports dynamiques ou privés.

Ainsi, un serveur possède des numéros de port fixes auxquels l'administrateur réseau a associé des services.
Du côté du client, le port est choisi aléatoirement parmi ceux disponibles par le système d'exploitation.

7. Les Sockets [-1-]

Il s'agit d'un modèle permettant la communication inter processus (IPC - Inter Processus Communication) afin de permettre à divers processus de communiquer aussi bien sur une même machine qu'à travers un réseau TCP/IP.
Les sockets sont généralement implémentées en langage C.Il se situent juste au-dessus de la couche transport du modèle OSI (protocoles UDP ou TCP), elle-même utilisant les services de la couche réseau (protocole IP / ARP).

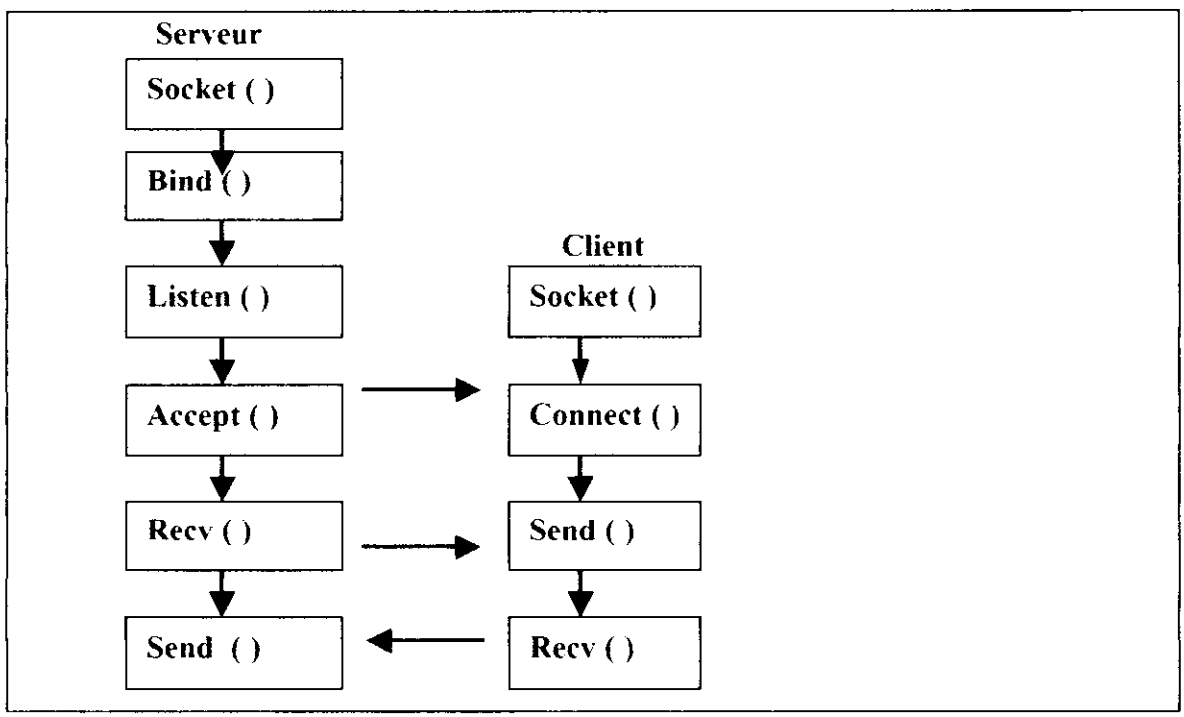


Figure 2.3 : La communication par Socket mode connecté

socket():création d'un socket.
bind ():Spécification de type de communication associé au socket.
Listen (): l'écoute des messages.
accept():permet la connexion en acceptant un appel.
connect():permet d'établir une connexion avec un serveur.
recv():permet de lire dans un socket en mode connecté (TCP).
send():permet d'écrire dans un socket.
close():permet la fermeture d'un socket.

8. Architecture client-serveur et le contrôle d'accès

Une application est bâtie selon une architecture client-serveur lorsqu'elle est composée de deux programmes, cooperant l'un avec l'autre à la réalisation d'un même traitement Dans le cas de contrôle d'accès on à deux partie, un identificateur (exemple:badge), qui contient l'authentification (mot de passe), cette authentification doit être analyser et comparer avec les informations situé dans une base de donnée(deuxième partie).
La première partie, appelée module client, est installée sur la porte de l'endroit contrôler alors que la seconde, appelée module serveur, est implantée sur l'ordinateur chargé de rendre le service (ouverture de porte,ou, déclanchement d'alarme). L'architecture client-serveur répond aux objectifs précédemment cités. On peut essayer de préciser ce que l'on entend par modèle client-serveur et contrôle d'accès par le schéma suivant :

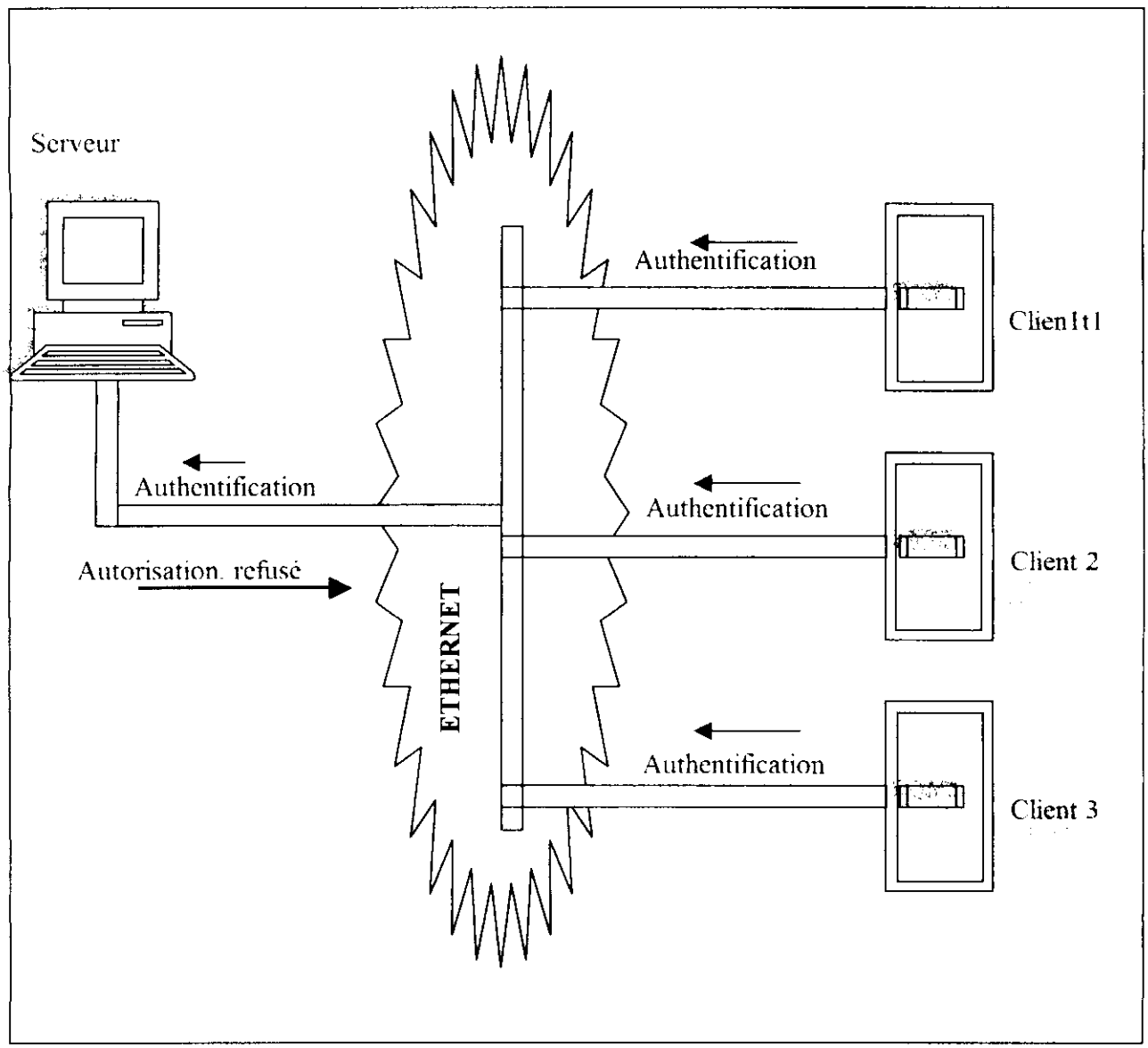


Figure 2.4 : Contrôle d'accès modèle client-serveur

9. Conclusion

Le modèle client/serveur est un système efficace dans notre cas, parce qu'il permet de contrôler plusieurs endroits, à partir d'un ordinateur centralisé. Comme on peut aussi ajouter et enlever un identificateur sans influencer les autres
Mais le modèle demande un réseau informatique fiable comme le cas du réseau Ethernet (TCP/IP).

III. ETUDE DU PROTOCOLE TCP/IP

1. Introduction

Dans ce chapitre, nous exposons les différentes couches d'une pile TCP/IP, ainsi que les différents protocoles associés.

2. Classification OSI

TCP/IP (Transmission Control Protocol / Internet Protocol) a été développé à partir de 1969.
A l'origine, les protocoles TCP/IP font partie de la hiérarchie des protocoles ARPA (Advanced Research Project Agency), sous l'égide du DOD (Department Of Defense) aux États-Unis. Ils sont présents dans toutes les implantations du système d'exploitation UNIX et constituent des protocoles de référence pour l'interconnexion des réseaux locaux et des réseaux longue distance.
Les protocoles TCP et IP servent de base à une famille de protocoles de niveau supérieur définis dans les RFC (Requests For Comments, demandes de commentaires), documents publiés par des organismes spécialisés.
TCP/IP n'est pas un unique protocole mais une suite de protocoles ou pile TCP/IP , travaillant sur un modèle en couches particulier DOD , qui recouvre les différentes couches du modèle ISO - notamment au niveau des couches basses- représentées comme suit:

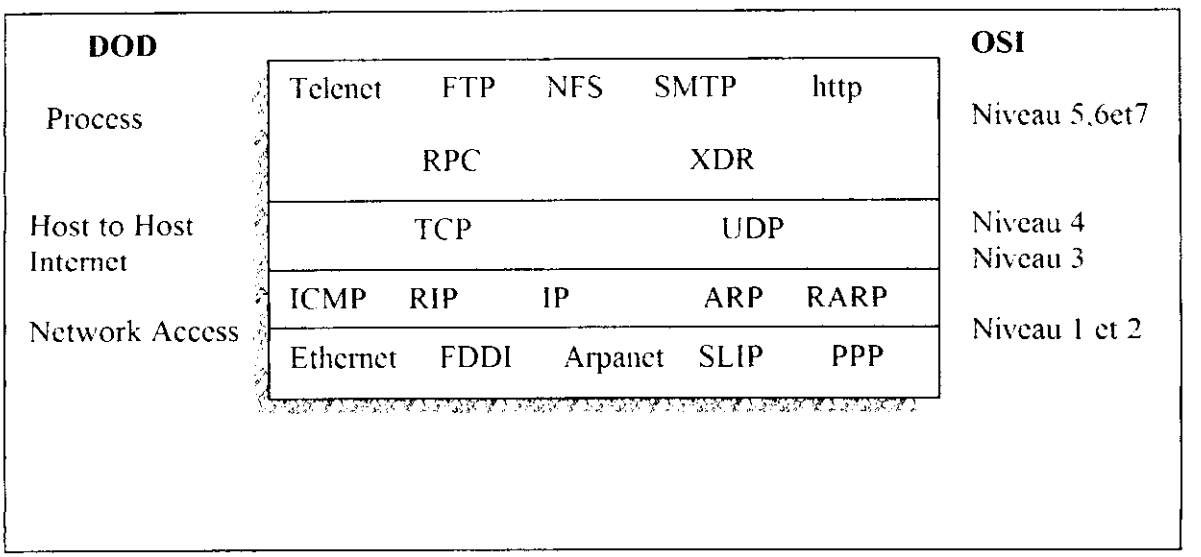


Figure 3.1 : Architecture DOD et OSI des protocoles TCP/IP

Niveaux 1 et 2
On trouve les protocoles liés aux architectures Ethernet. Les identifiants des sous-couches MAC et LLC peuvent prendre deux valeurs distinctes suivant l'architecture utilisée.
Pour une architecture type 802.3, les champs DSAP et SSAP de la sous-couche LLC prennent la valeur 06h pour indiquer le protocole IP au niveau supérieur.
Pour une architecture type Ethernet II, la sous-couche LLC n'existe pas, le protocole IP est indiqué directement dans le champ longueur de la sous-couche MAC par la valeur 0800h.
Les procédures SLIP (Serial Line Internet Protocol) et PPP (Point to Point Protocol) sont des cas particuliers permettant d'adapter le réseau ou le poste de travail à une communication série asynchrone par l'intermédiaire d'un modem avec un serveur distant (cas du réseau Internet).

Niveau 3
On trouve l'implantation du protocole IP (Internet protocol). Ce protocole, en mode datagramme, va offrir les fonctions de routage. L'interconnexion entre deux machines situées n'importe où sur le réseau est possible. Le protocole IP gère également la fragmentation des données.
La couche 3 contient quatre autres protocoles :
- ARP (Address Resolution Protocol) permet de faire la correspondance entre les adresses logiques (Internet) et les adresses physiques (MAC). Ce protocole permet de masquer les adresses nécessaires à l'acheminement des trames de niveau MAC.
- RARP (Reverse Address Resolution Protocol) permet d'établir la correspondance entre les adresses physiques (MAC) et les adresses logiques (Internet).
- ICMP (Internet Control Message Protocol) n'est pas à proprement parlé un protocole de niveau 3,puisqu'il utilise l'encapsulation IP. Mais il sert à la gestion du protocole IP.
- RIP (Routing Information Protocol) est un protocole de routage utilisant le principe de la multidiffusion. Les routeurs utilisant RIP diffusent périodiquement leurs tables de routage aux autre routeurs du réseau.

Niveau 4
On trouve le protocole TCP (Transmission Control Protocol) qui offre aux utilisateurs un transfert fiable sur connexion et le protocole UDP (User Datagramme Protocol) qui offre un transfert en mode datagramme.

Niveau 5
On trouve les routines de base des RPC (Remote Procedure Call) qui permettent de cacher aux couches supérieures les accès au réseau en utilisant la sémantique des appels de fonctions. Ces routines se trouvent dans des bibliothèques liées aux programmes d'application au moment de la compilation.

Niveau 6
Les procédures XDR (eXternal Data Representation) de la couche 6 permettent de rendre universelle la représentation des données et de s'affranchir des codages et de la structuration des données proposée par les différents constructeurs.

Niveau 7
Regroupe les différentes applications :
- Telnet (Terminal Emulation Protocol) pour la connexion et l'émulation de terminal.
- FTP (File transfert Protocol) pour le transfert de fichiers.
- NFS (Network File Server) pour la gestion de fichiers.
- SNMP (Simple Network Management Protocol) pour l'administration et la gestion des machines du réseau .
- SMTP (Simple Mail Transfert Protocol) pour les services de courrier électronique.
- HTTP (HyperText Transmission Protocol) pour des recherches d'informations en mode hypertexte.

3. Architecture des protocoles TCP/IP

Etant donné qu'il n'existe pas d'accord général concernant la description de TCP/IP à l'aide modèle en couche, celui-ci est généralement représenté sous la forme d'un modèle dont le nombre de couche est inférieur à celui du modèle OSI qui en contient 7couches.La plupart des description de TCP/IP définissent de trois a cinq niveaux fonctionnels dans son architecture .Le modèle à quatre niveaux illustré en figure suivante se base sur les trois couches (application , hôte-à-hôte et Accès Réseau)exposées par le DOD Protocol Model dans le DDN PROTOCOL Handbook , volume 1.

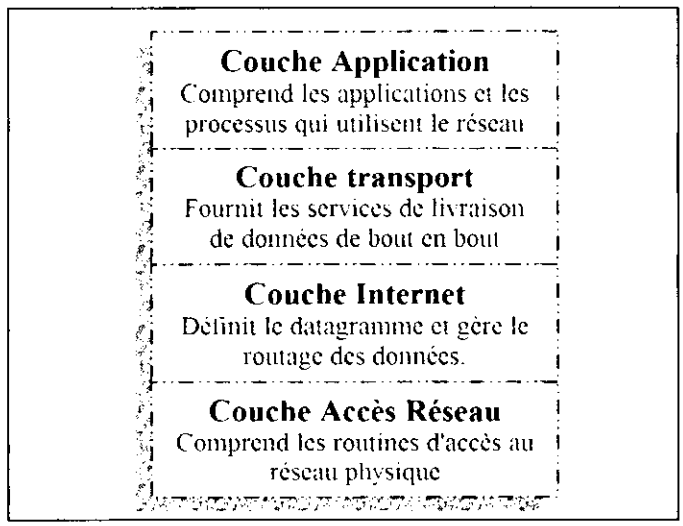


Figure 3.2 : Les couches de l'architecture des protocoles TCP/IP

Chacune des couches de la pile ajoute des informations de contrôle afin d'assurer une livraison correcte. Ces informations de contrôle sont appelées un en-tête parce qu'elles sont placées devant les données à transmettre. Chaque couche traite toutes les informations qu'elle reçoit des couches supérieures en tant que données et place son propre en-tête avant elles. L'addition d'informations de distribution à chaque couche est appelée encapsulation. Quand les données reçues, c'est l'inverse qui se produit.

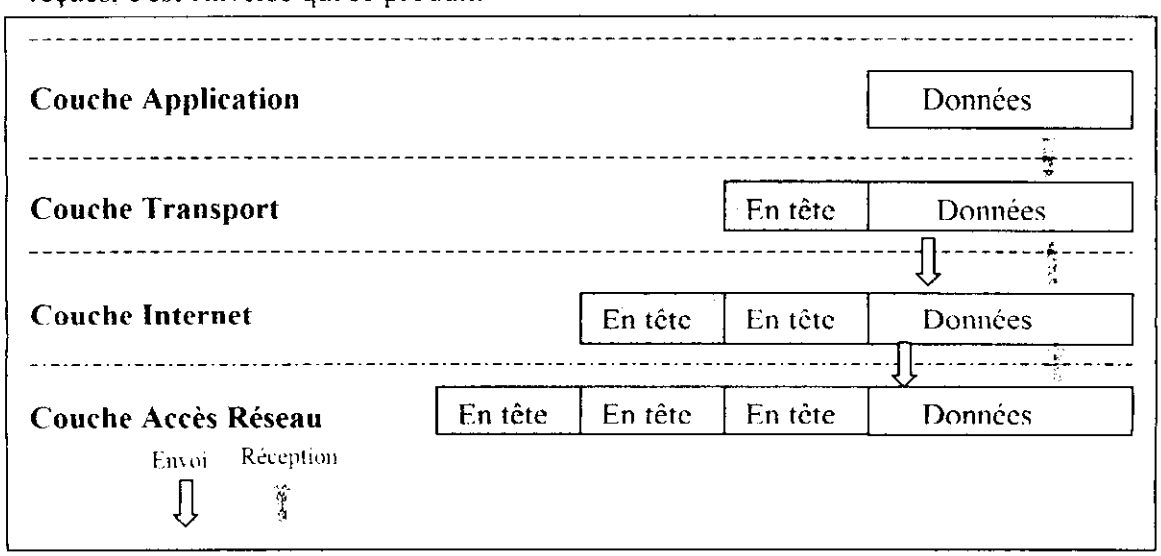


Figure 3.3 : Encapsulation de données

3.1. Couche Accès Réseau

La Couche d'Accès Réseau est le plus bas niveau de la hiérarchie des protocoles TCP/IP. Les protocoles de cette couche fournissent le moyen de délivrer des données aux autres systèmes directement rattachés au réseau. Il définit la façon d'utiliser le réseau pour transmettre un datagramme IP.
A l'inverse des protocoles de plus haut niveau, ceux de la Couche d'Accès Réseau doivent connaître les détails du réseau sous-jacent (sa structure de paquets, son adressage, etc.) pour formater correctement les données transmises afin de se conformer aux contraintes de réseau. La Couche d'Accès Réseau TCP/IP peut regrouper les fonctions des trois couches les plus basses du modèle de référence OSI (réseau, liaison et physique).

3.2. La couche Internet

La couche Internet, c'est elle qui définit les datagrammes (paquets de données), et qui gère les notions d'adressage IP.
Elle permet l'acheminement des datagrammes vers des machines distantes ainsi que de la gestion de leur fragmentation et de leur assemblage à réception. La couche Internet contient 5 protocoles:
- Le protocole IP.
- Le protocole ARP.
- Le protocole ICMP.
- Le protocole RARP.
- Le protocole IGMP.

3.2.1. Le protocole IP (Internet Protocol)

Le protocole Internet est un protocole de niveau réseau. Il est responsable de :
- la transmission des données en mode sans connexion.
- l'adressage et le routage des paquets entre stations par l'intermédiaire de routeurs.
- la fragmentation des données.

Lors de l'émission, les fonctionnalités assurées sont :
- identification du paquet.
- détermination de la route à suivre (routage).
- vérification du type d'adressage (groupe ou diffusion).

À la réception, les fonctionnalités sont :
- vérification de la longueur du paquet.
- contrôle des erreurs.
- réassemblage en cas de fragmentation.
- transmission du paquet réassemblé au niveau supérieur.

Format du paquet

Le paquet IP, ou datagramme IP, est organisé en champs de 32 bits.
Les fonctionnalités IP se retrouvent dans chaque groupement de bits de l'en-tête.
- Version : numéro de version du protocole IP (actuellement on utilise la version 4 IPv4).
- Longueur : ou IHL pour Internet Header Length , longueur de l'en-tête codée sur 4 bits et représentant le nombre de mots de 32 bits (généralement 5).
- Type de service (TOS) : il indique la façon selon laquelle le datagramme doit être traité.

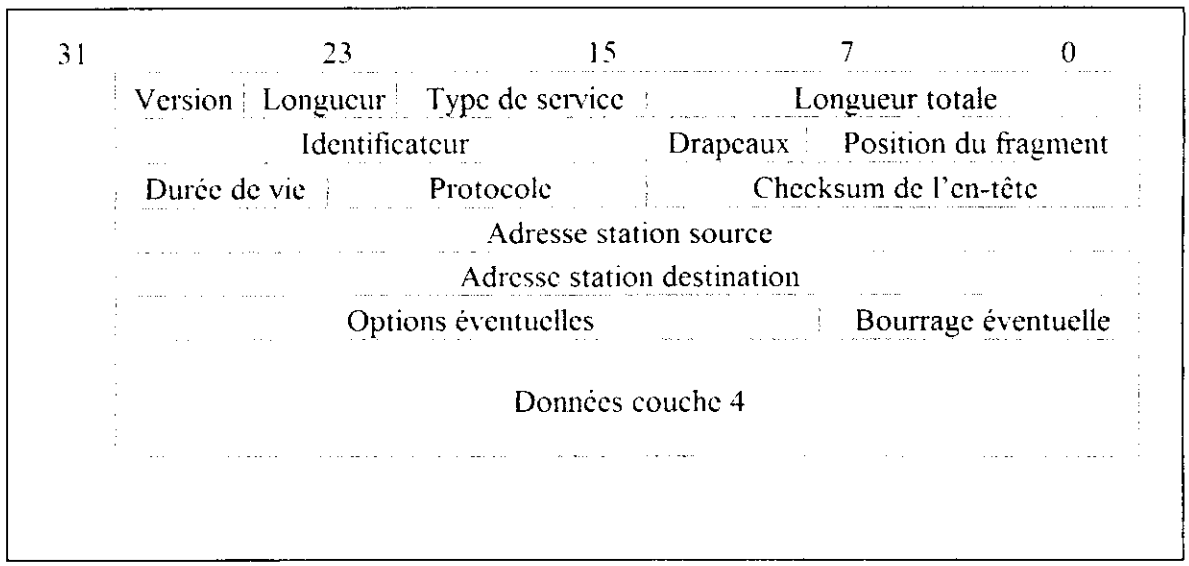


Figure 3.4 : Format de paquet IP

- Longueur totale : longueur totale du fragment (en-tête et données) exprimée en nombre d'octets.
-Identificateur : identifie le paquet pour la fragmentation (tous les fragments d'un même paquet portent le même numéro).
- Drapeaux : gère la fragmentation sur 3 bits (le premier réserver) suivant le format : DF, MF.
    1. le bit DF (Don 't Fragment) demande au routeur de ne pas fragmenter le paquet
    2. Le bit MF (More Fragment) est positionné à 1 dans tous les fragments, sauf le dernier.
- Position du fragment : indique par multiple de 8 octets la position du fragment dans le paquet courant. Tous les fragments du paquet, sauf le dernier, doivent donc avoir pour longueur des multiples de 8 octets. Avec un codage sur 13 bits, le maximum pour un paquet est de 8192 fragments.
- Durée de vie (TTL, Time to live) : ce champ indique le nombre maximal de routeurs à travers lesquels le datagramme peut passer. Ainsi ce champ est décrémenté à chaque passage dans un routeur, lorsque celui-ci atteint la valeur critique de 0, le routeur détruit le datagramme. Cela évite l'encombrement du réseau.
- Protocole : indique le protocole de la couche supérieure (1 pourICMP, 2 pour IGMP ,6 pour TCP, 17 pour UDP).
- Somme de contrôle de l'en-tête (header checksum) : 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 : Ce champ représente l'adresse IP de la machine émettrice, il permet au destinataire de répondre.
-Adresse IP destination : adresse IP du destinataire du message.
- Options : utilisées pour le contrôle ou la mise au point.

L'adressage Internet :
Chaque machine susceptible d'être connectée à l'extérieur de son réseau local possède une adresse IP en principe unique.
Une autorité internationale le NIC (Network Information Center) attribue des numéros à chaque réseau. Les adresses codées sur 32 bits comportent deux partie : le numéro de réseau (Net id) et le numéro de la machine sur le réseau (Host_id). Le NIC n'alloue que les numéros de réseau. L'affectation des numéros complets est à la charge des administrateurs réseaux. Suivant l'importance du réseau, plusieurs classes d'adressage sont possibles.

0	Net_id (adr.réseau sur7bits)	Host_id(adr.station sur 24 bits)	Classe A
10	Net_id (adr.réseau sur 14 bits)	Host_id(adr.station sur 16 bits)	Classe B
110	Net_id (adr.réseau sur 21 bits)	Host_id(adr.station sur 8 bits)	Classe C
1110	Adr.Multicast(28 bits)		Classe D
1111	Format indéfini (28 bits)		Classe E

Figure 3.5 : Format des adresses IP

Les adresses sur 32 bits sont exprimées par octet (soit quatre nombres compris entre 0 et 255) notées en décimal et séparés par des points : 137.15.223.2.
Les différentes classes d'adresse correspondent donc à des nombres appartenant aux plages suivantes :

Classe A :1.0.0.0 à 126.0.0.0,soit 126 réseau et 16 777 214 machines.
Classe B :128.1.0.0 à 191.254.0.0,soit 16 382 réseau et 65 535 machines.
Classe C :192.0.1.0 à 223.255.254.0,soit 2 097 150 réseau et 254 machines
Classe D :224.0.0.1 à 239.255.255.255, soit 268 435 455 adresses de groupe
Classe E : 240.0.0.0 à 255.255.255.254.

Figure 3.6 : Nombre des réseau et machines dans chaque classe

La classe A : représente donc les réseaux de grande envergure (ministère de la défense, réseaux d'IBM,...) dont la plupart se trouvent aux États-unis.
La classe B : désigne les réseaux moyens (universités, centres de recherches...).
La classe C : représente les petits réseaux régionaux.
Le classe D : ne désigne pas une machine particulière sur le réseau, mais un ensemble de machines voulant partager la même adresse et ainsi participer à un même groupe : adresses de

groupe de diffusion(multicast). Ces adresses sont choisies arbitrairement par les concepteurs des applications concernées (News, multimédia...).
Les autres adresses sont particulières ou réservées :
- l'adresse dont la partie basse est constituée de bits à 0 est une adresse réseau ou sous-réseau, 212.92.27.0
- l'adresse dont la partie basse est constituée de bits à 1 est une adresse de diffusion (broadcast), 157.42.255.255
- 127.0.0.1 est une adresse de bouclage (localhost, loopback) et permet l'utilisation interne de TCP/IP sans aucune interface matérielle.
- 0.0.0.0 est une adresse non encore connue, utilisée par les machines ne connaissant pas leur adresse IP au démarrage.
Le nombre d'attribution d'adresses IP a suivi ces dernières années une croissance presque exponentielle, ce qui a conduit à une saturation. Une nouvelle norme IPV6 doit remplacer la version 4 actuelle du protocole IP et offrira un codage des adresses sur 128 bits.

3.2.2. Le protocole ICMP (Internet Control Message Protocol)

Le ICMP, défini par la RFC 792, fait partie intégrante de IP .ICMP envoie des messages qui se chargent des fonctions suivantes pour TCP/IP:

1. Contrôle de flux : quand les datagrammes arrivent trop vite pour être traités, l'hôte de destination renvoie un Source Quench Message (message d'arrêt de la source) ICMP vers l'émetteur. Ce message demande a l'émetteur à la source d'arrêter temporairement d'envoyer des datagrammes.
2. Détection de destinations inaccessibles: Quand une destination est inaccessible, le système qui détecte le problème envoie un message Destination Unreachable à la source du datagramme.
3. Redirection des routes: Une passerelle envoie le message ICMP Redirect pour dire à un hôte d'utiliser une autre passerelle, a priori parce que cette dernière est un meilleur choix .Ce message ne peut être employé que quand l'hôte source se trouve sur le même réseau que les deux passerelles.
4. Vérification des hôtes distants: Un hôte peut envoyer le message ICMP Echo pour voir si le IP d'un système distant est opérationnel .Quand un système reçoit un message d'écho, il renvoie le même paquet à l'hôte source.

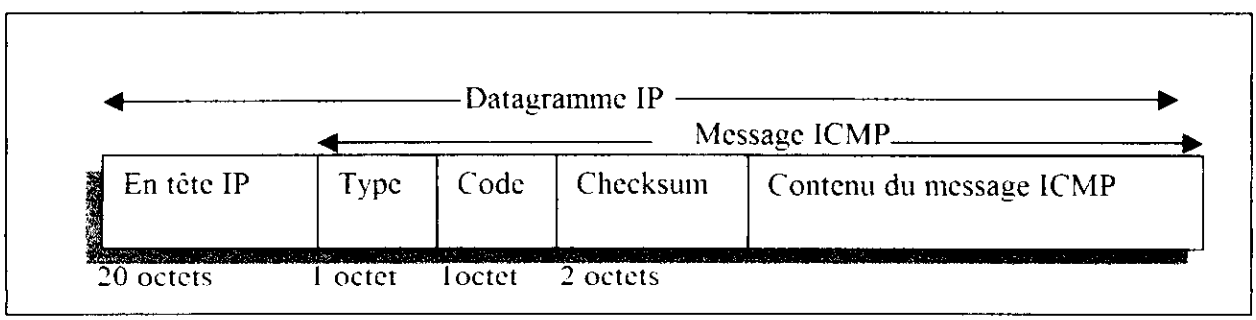


Figure 3.7 : Encapsulation du message ICMP

Le champ type permet de préciser la nature du message, le code permet de préciser éventuellement la nature de ce message

Code ICMP	Message ICMP
0	Echo reply
3	Destination Unreachabele
4	Source Quench
5	Redirect
8	Echo Request
11	Time Exceeded for datagram
12	Paramater Problem on Dtagram
13	Timestamp request
14	Timestamp reply
17	Adress mask request
18	Adress mask reply

Figure 3.8 : Codes ICMP

3.3. Couche transport

La couche transport contient deux protocoles permettant à deux applications d'échanger des données indépendamment du type de réseau emprunté (c'est-à-dire indépendamment des couches inférieures...), il s'agit des protocoles suivants :

- TCP, un protocole mode connecté qui assure le contrôle des erreurs.
- UDP, un protocole mode non connecté dont le contrôle d'erreur est archaïque.

3.3.1. Mode connecté et non connecté

Dans mode connecté, les paquets doivent être remis dans l'ordre d'émission au destinataire et chaque paquet est dépendant des autres. Ils peuvent emprunter des voies physiques différentes. TCP fonctionne en mode connecté.
Dans le mode non connecté, les paquets sont indépendants les uns des autres, Ils peuvent emprunter des voies physiques différentes et ne restent pas forcément séquencés lors de l'acheminement .Il faut donc que chaque paquet soit muni de l'adresse réseau complète (32 bits sous IP).IP fonctionne en mode non connecté.

3.3.2. Le protocole UDP (User Datagram Protocol)

UDP est un protocole sans connexion et permet à une application d'envoyer des messages à une autre application avec un minimum de fonctionnalités (pas de garanties d'arrivée, ni de contrôle de séquencement).
Il n'apporte pas de fonctionnalités supplémentaires par rapport à IP et permet simplement de désigner les numéros de port correspondant aux applications envisagées avec des temps de réponse courts. Un message UDP est désigné dans un paquet IP par une valeur du champ protocole égal à 17.
- Le port source et le port destination : permettent de référencer les applications qui s'exécutent sur les machines locales et distantes. Les valeurs supérieures à 1 000 correspondent à des ports clients et sont affectées à la demande par la machine qui effectue une connexion TCP.
- La longueur : indique la longueur totale du message en octets (données et en-tête).
- La somme de contrôle : est calculée comme pour les paquets IP.

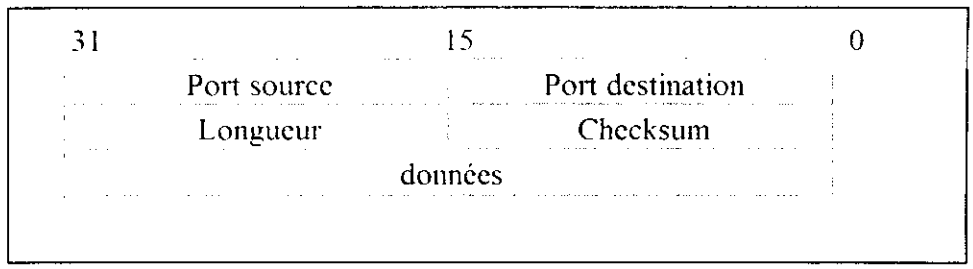


Figure 3.9 : Format d'un message UDP

3.3.3. Le protocole TCP (Transmission Control Protocol)

Les applications qui exigent du protocole de transport qu'il fournisse une transmission fiable des données utilisent TCP/IP parce qu'il vérifie que les données sont correctement transmisse à travers le réseau et dans la séquence appropriée .TCP est un protocole fiable ,orienté connexion. L'utilisation d'un mécanisme appelé "accusé de réception positif avec la retransmission" (PAR Positif Acknowlegment with Retransmission) permet à TCP de garantir des transmissions fiables. Le système PAR envoie de nouveau les données à moins que le système à distance ne lui renvoie à un message précisant que les données sont arrivées correctement. L'unité d'échange de données est appelée "segment". Chaque segment contient un total de contrôle que le destinataire utilise pour vérifier que les données n'ont pas été endommagées pendant leurs transmissions. Si le segment de données est reçu en parfait état, le récepteur envoie un accusé de réception positif à l'émetteur. Dans le cas contraire, le récepteur élimine ce segment de données. Après un délai d'attente déterminé, l'émetteur retransmet les segments pour lesquels aucun accusé de réception positif n'a été reçu.
TCP orienté connexion .Il établit une connexion logique de bout en bout entre les deux hôtes communiquant entre eux.
Ses principales caractéristiques sont :
- établissement et fermeture de la connexion virtuelle, segmentation et réassemblage des données .
- acquittement des datagrammes reçus et retransmission sur absence d'acquittement (un reséquencement est effectué si la couche IP ne les délivre pas dans l'ordre).
- contrôle de flux.
- multiplexage des données issues de plusieurs processus hôtes en un même segment.
- gestion des priorités des données et de la sécurité de la communication.

3.3.3.1 Format des segments TCP

- Les numéros de port permettent de référencer les applications.
- Le numéro de séquence indique le numéro du premier octet transmis dans le segment.
- Le numéro d'acquittement contient le numéro de séquence du prochain octet attendu par l'émetteur.
- La longueur de l'en-tête est codée sur 4 bits et donne le nombre de mots de 32 bits.
- Les bits de contrôle permettent de définir la fonction des messages ainsi que la validité de certains champs :

1. URG = 1 si le champ des priorités est utilisé (pour des demandes d'interruption d'émission par exemple).
2. ACK = 1 si la valeur du champ acquittement est significative.

3. EOM (ou PSH) indique une fin de message (End of Message), les données doivent être transmises (pushed) *à* la couche supérieure.
4. RST (Reset) : demande de réinitialisation de la connexion.
5. SYN : demande d'ouverture de connexion (les numéros de séquence doivent être synchronisés)
6. FIN : fin de connexion.

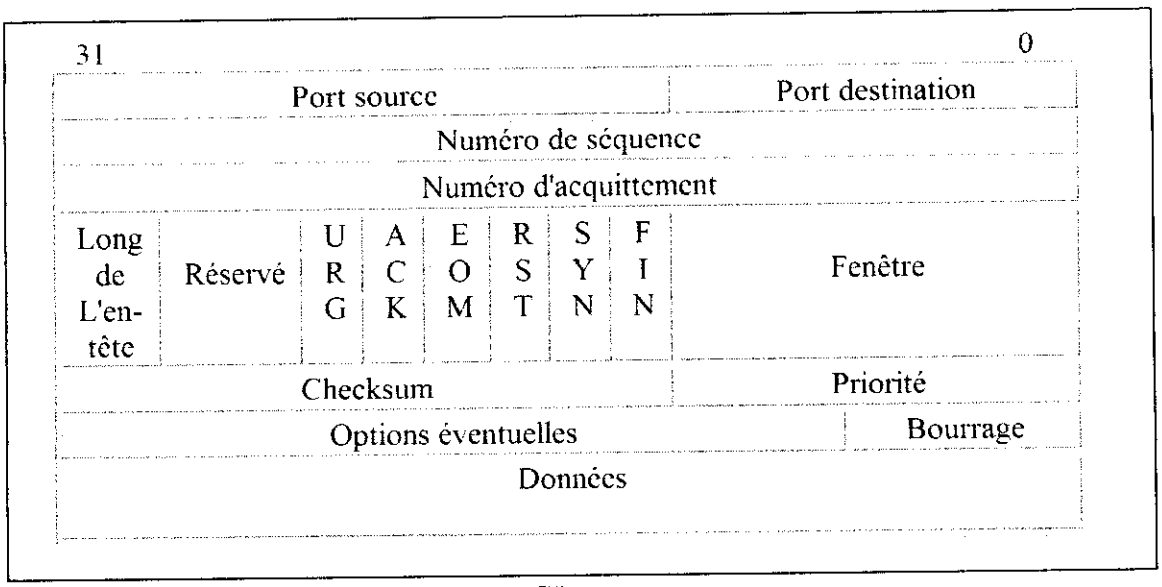


Figure 3.10: format des segments TCP

- Le champ fenêtre (Windows) indique le nombre d'octets que le récepteur peut accepter à partir du numéro d'acquittement.
- Le champ checksum correspond à une somme de contrôle de l'en-tête et du message.
- Le champ priorité contient lors d'une interruption d'émission (URG=1) un pointeur sur les octets de données à traiter en priorité.
- Le champ options permet de définir, par exemple, la taille maximale d'un segment.

3.3.3.2 Ouverture d'une connexion

Après autorisation locale sur chaque station et déclaration d'un identificateur permettant l'application de référencer la connexion, la demande d'ouverture de connexion est transmise à la couche transport qui positionne son bit SYN à 1 (figure suivante). Le numéro séquence initial à l'émission (Initial Send Sequence number, ISS) est délivré, au moment de la demande, par un compteur incrémenté toutes les 4 ms.

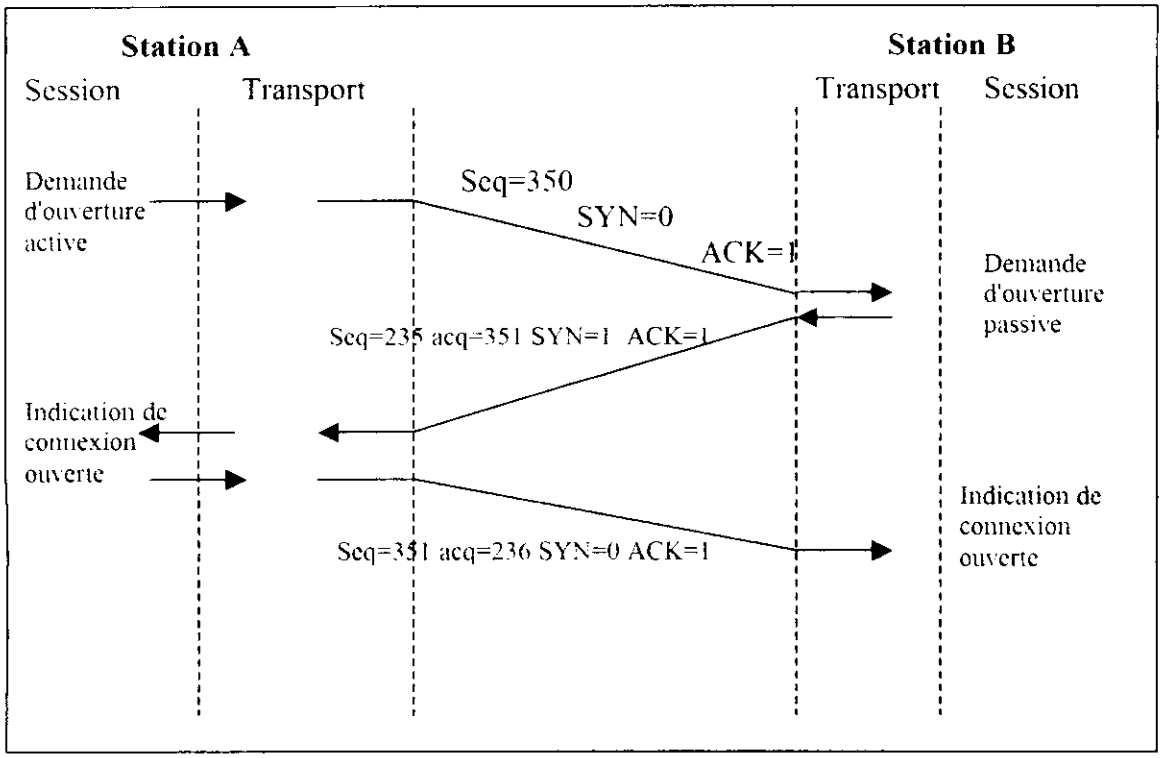


Figure 3.11 : Exemple de connexion réussie

3.3.3.3 Transfert de données

Le transfert de données peut alors commencer avec les numéros de séquence en cours. Le contrôle de flux est réalisé dans les deux sens par les numéros d'acquittement (le bit ACK est alors positionné *à* 1). La taille de la fenêtre de transmission sans acquittement est fixée par le destinataire avant envoi du premier segment.

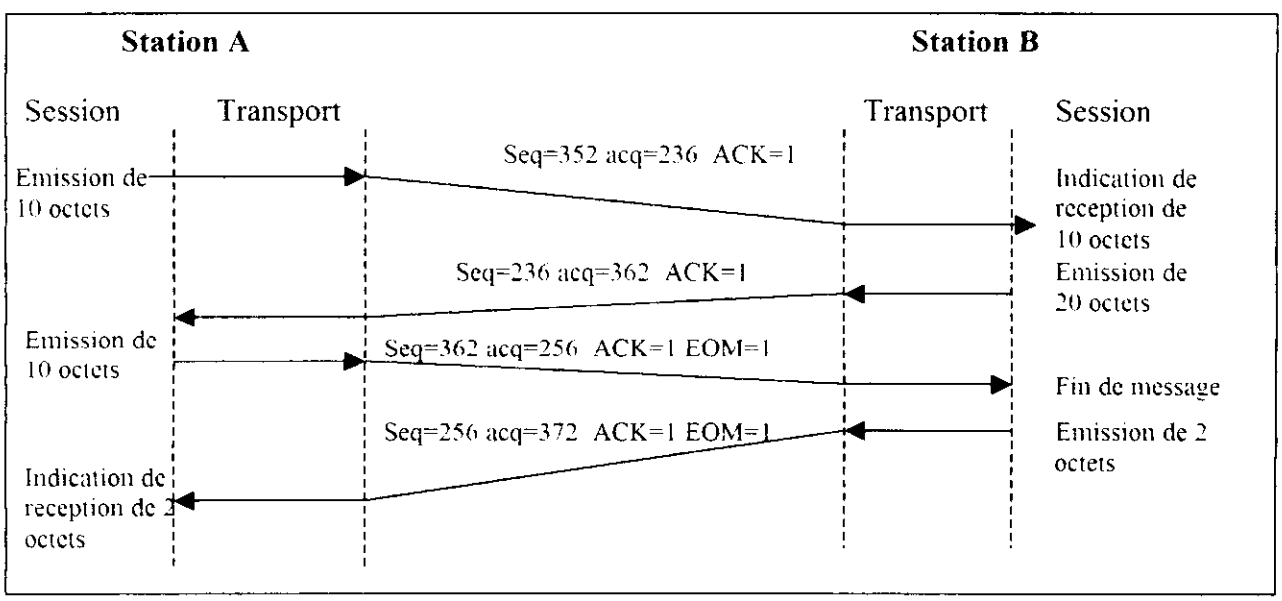


Figure 3.12 : Exemple d'échange TCP

3.3.3.4 Fermeture d'une connexion

La fermeture d'une connexion est réalisée lorsque le récepteur reçoit un en-tête TCP dont le bit FIN est positionné à 1. La demande est traitée dans les deux sens aux niveaux supérieurs avant acquittement.

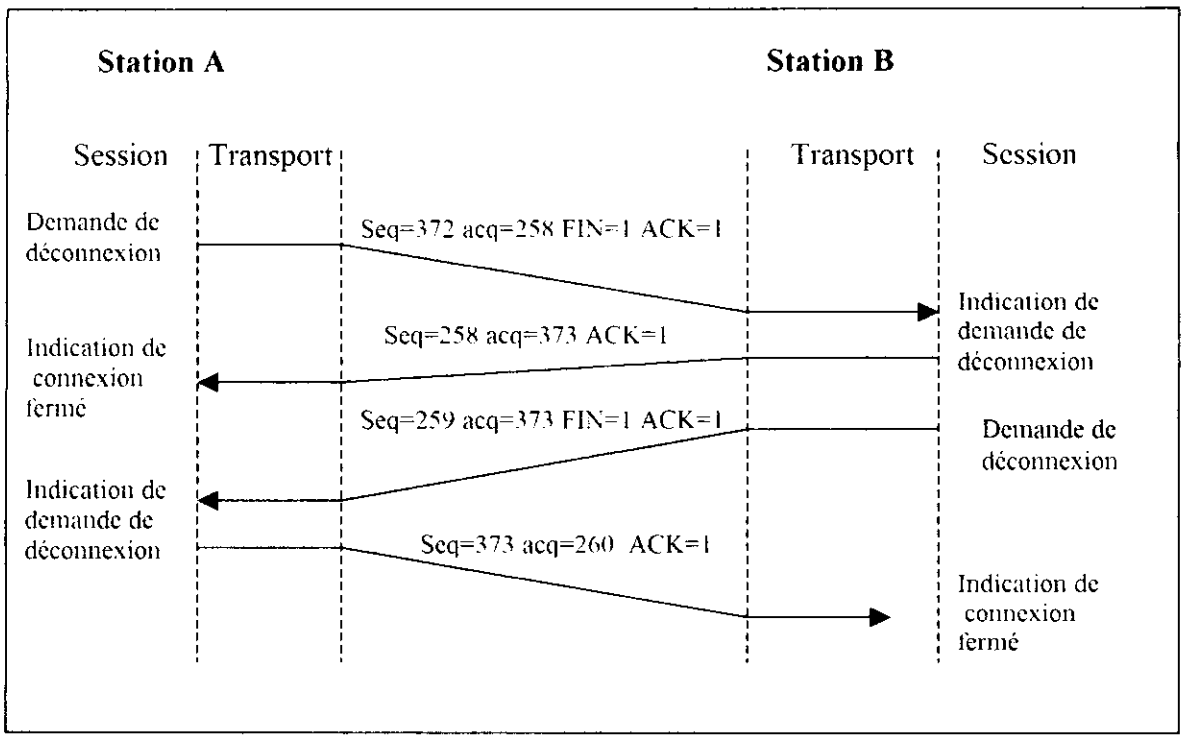


Figure 3.13 : Exemple de fermeture réussie

3.4. Couche application [4]

La couche application constitue le sommet de l'architecture TCP/IP. Cette couche inclut tous les processus qui utilisent les protocoles de la couche transport pour transmettre des données. Il existe de nombreux protocoles d'application, la plupart assurent les services utilisateur entre les nouveaux services sont toujours ajoutés au niveau de cette couche. Les protocoles d'applications les plus répandus sont:

- TELNET : Le protocole de terminal de réseau (Network Terminal Protocol), qui permet l'ouverture d'une session à distance sur un réseau.
- FTP : Le protocole de transfert de fichiers (File Transfert Protocole), qui est utilisé pour le transfert de fichiers.
- SMTP : Le protocole de transfert de courrier (Simple Mail Transfert Protocole) qui est utilisé pour le courrier électronique.

Autre Applications:

- Le service DNS (Domain Name Service) : Il également appelé Name service. Cette application établit la correspondance entre les adresses IP et les noms attribués aux machines-périphériques du réseau.
- Le protocole RIP (Routing Information Protocol) : Le routage est l'un des principaux éléments du fonctionnement de TCP/IP. Les systèmes en réseau utilisent RIP pour échanger des informations concernant le routage des données.

- Le protocole NFS (Network File System) : Il permet de partager des fichiers entre différentes machines-hôtes du réseau.

La figure suivante illustre la hiérarchie des protocoles dans un ordinateur imaginaire.
Cette illustration ne sert qu'à nous aide à visualiser les relations existant entre les nombreux protocoles sur une machine unique.

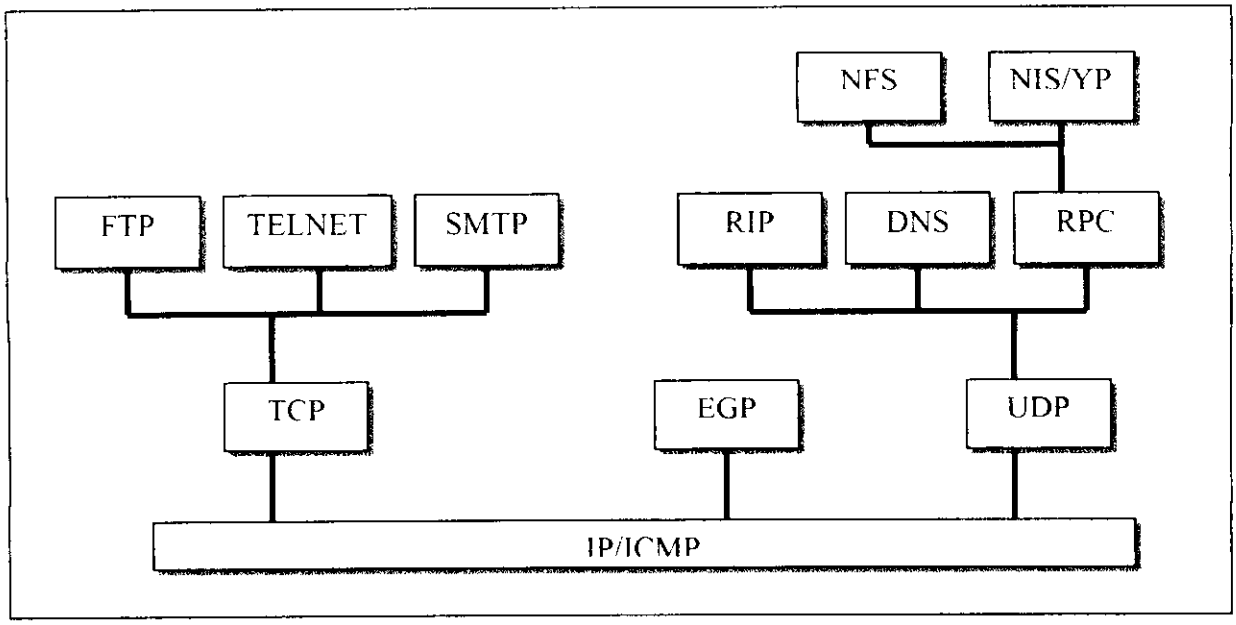


Figure 3.14 : Protocoles TCP/IP

3.5. Le masque de réseau (Netmask)

La partie de l'adresse Internet administrée localement (host_id) peut être découpée en deux parties : une adresse de sous-réseau et une adresse de numéro de machine.
Le système d'exploitation doit déterminer l'information désignant le sous-réseau et l'information désignant la machine. Cette structuration est employée, par exemple, dans les algorithmes de routage pour savoir si deux machines se trouvent sur le même sous-réseau.
Un masque de sous réseau ou Netmask a le même format qu'une adresse Internet. Les bits à 1 désignent la partie sous réseau de l'adresse et les bits à 0 la partie numérotation des machines sur le sous-réseau.

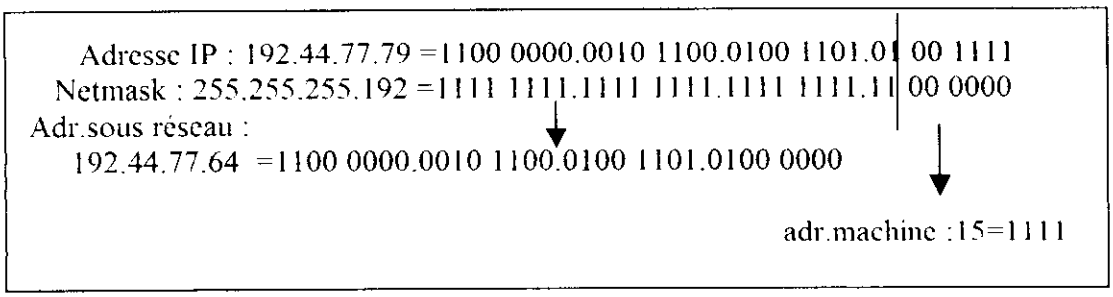


Figure 3.15 : Exemple d'utilisation du masque

Dans cet exemple de réseau de classe C, les 2 bits de poids fort des 8 bits disponibles sont utilisés pour identifier le sous-réseau. Il est ainsi possible de distinguer 4 adresses de sous-réseaux (192.44.77.0, 192.44.77.64,192.44.77.128, 192.44.77.192).

Le masquage peut servir à séparer localement deux sous-réseaux correspondant à des entités différentes (administration, services techniques . . .).

Masques de sous-réseau par défaut pour les classes standard:
- classe A : 255.0.0.0
- classe B : 255.255.0.0
- classe C : 255.255.255.0

3.6. Le routage [4]

Le routage d'un paquet consiste à trouver le chemin de la station destinatrice à partir de son adresse IP.
Si le paquet émis par une machine ne trouve pas sa destination dans le réseau ou sous-réseau local, il doit être dirigé vers un routeur qui rapproche le paquet de son objectif. Il faut par conséquent que toutes les stations du réseau possèdent l'adresse du routeur par défaut. Chaque routeur doit donc connaître l'adresse du routeur suivant lorsque la machine de destination n'est pas sur les réseaux ou sous-réseaux qui lui sont raccordés. Il doit gérer une table de routage de manière statique ou dynamique.

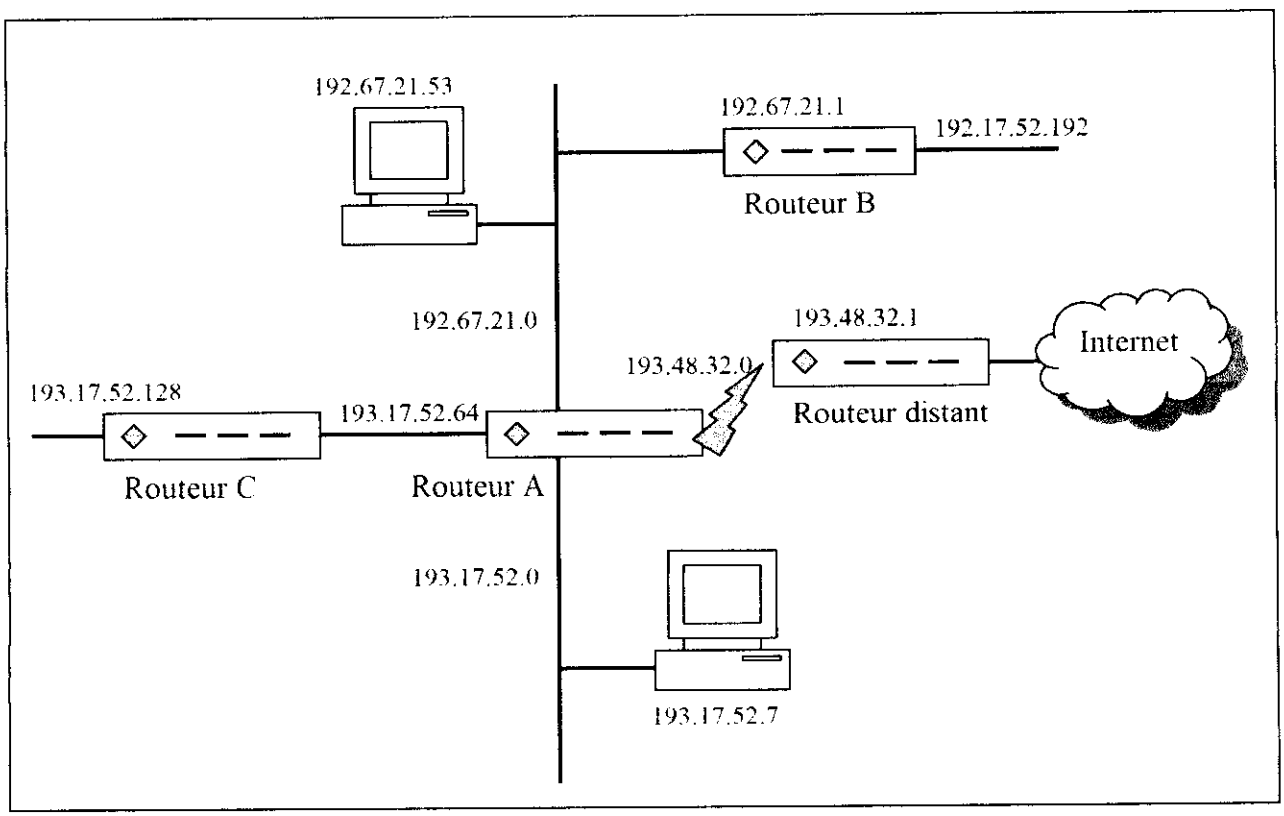


Figure 3.16 : Exemple de routage

La table de routage du routeur A est :
192.67.21.0 is directly connected , Ethernet 0.
193.48.32.0 is directly connected , Ethernet 1.
193.17.52.0 is directly connected , Ethernet 2.
193.17.52.64 is directly connected, Ethernet 3.
193.17.52.0 is subnetted (mask is 255.255.255.192), 4 subnets.

193.17.52.128 via 193.17.52.67
193.17.52.192 via 192.67.21.1
0.0.0.0 via 193.48.32.1
Les quatre premières lignes identifient les assignations d'adresses IP à des interfaces physiques, le routeur possède donc 4 ports Ethernet.
La cinquième ligne précise, en fonction de la valeur du masque, qu'il existe 4 sous-réseaux (193.17.52.0, 193.17.52.64, 193.17.52.128 et 193.17.52.192).
Les lignes suivantes concernent des routages statiques. La première entrée indique que lorsque le routeur reçoit un paquet dont l'adresse de destination est sur le réseau 193.17.52.128, le paquet doit être envoyé au routeur 193.17.52.67.
La dernière entrée définit le routage par défaut : si aucune des routes définies précédemment ne convient, le paquet est renvoyé vers la machine 193.48.32.1 qui est un routeur distant.
Dans le cas du routage statique, la table est établie une fois pour toutes. Ce type de routage simple peut être utilisé pour un réseau local avec une connexion externe.
Pour le routage dynamique, la table est mise à jour périodiquement à l'aide de protocoles spécifiques. Les routeurs envoient régulièrement la liste des réseaux ou des sous-réseaux que l'on peut atteindre par eux .Ce qui permet aux autres routeurs de mettre à jour leurs table de routage. Ils évaluent dynamiquement la meilleure route vers chaque réseau ou sous-réseaux.
Les principaux protocoles de routage dynamique sont :
- RIP (Rout ing Information Protocol) qui utilise une technique de diffusion (broadcast) périodique.Les transferts se font à l'aide de datagrammes UDP.
- EGP (Exterior Gateway Protocol) qui limite la transmission de la table au routeur voisin(dialogue). Les transferts se font à l'aide de datagrammes IP.

4. Conclusion

Cette partie nous a permis d'avoir un aperçu global sur le passage des données à travers les différentes couches de la pile TCP/IP.
Pour l'émission ou la réception des données il nous faut adapter ces données pour qu'elles soient compatibles avec le support de transmission. C'est ce qu'on va aborder dans le chapitre suivant.

IV. ETHERNET

1. Introduction

Afin de mieux comprendre la conception d'une trame, ce chapitre expose les différents champs d'une trame. Une présentation des différentes erreurs liées à la transmission et réception des données a été aussi faite.

2. L'historique du standard

2.1. The Ethernet

Au début des années 1970, la société Xerox travaillait sur des systèmes bureautiques ouverts et des embryons des réseaux locaux .Elle conçut une version expérimentale d'Ethernet fonctionnant à 3 Mbit/s, sur du câble coaxial de 75 ohm et pouvant couvrir jusqu'à un kilomètre, mais les technologies évoluaient encore.

En collaboration avec DEC (Digital Equipment Corporation) et Intel, Xerox publia le livre bleu the Ethernet en septembre 1980 (version 1.0). Le nom signifiant réseau (net ou network) de l'ether (le câble passif dans le cas présent), les modifications à venir n'étant que des amélioration des composants matériels (pas la méthode, la topologie).

En novembre 1982, ils publièrent la version 2.0 (AA-K759B-TK), qui était assez complète, mais partiellement incompatible avec la version précédente.

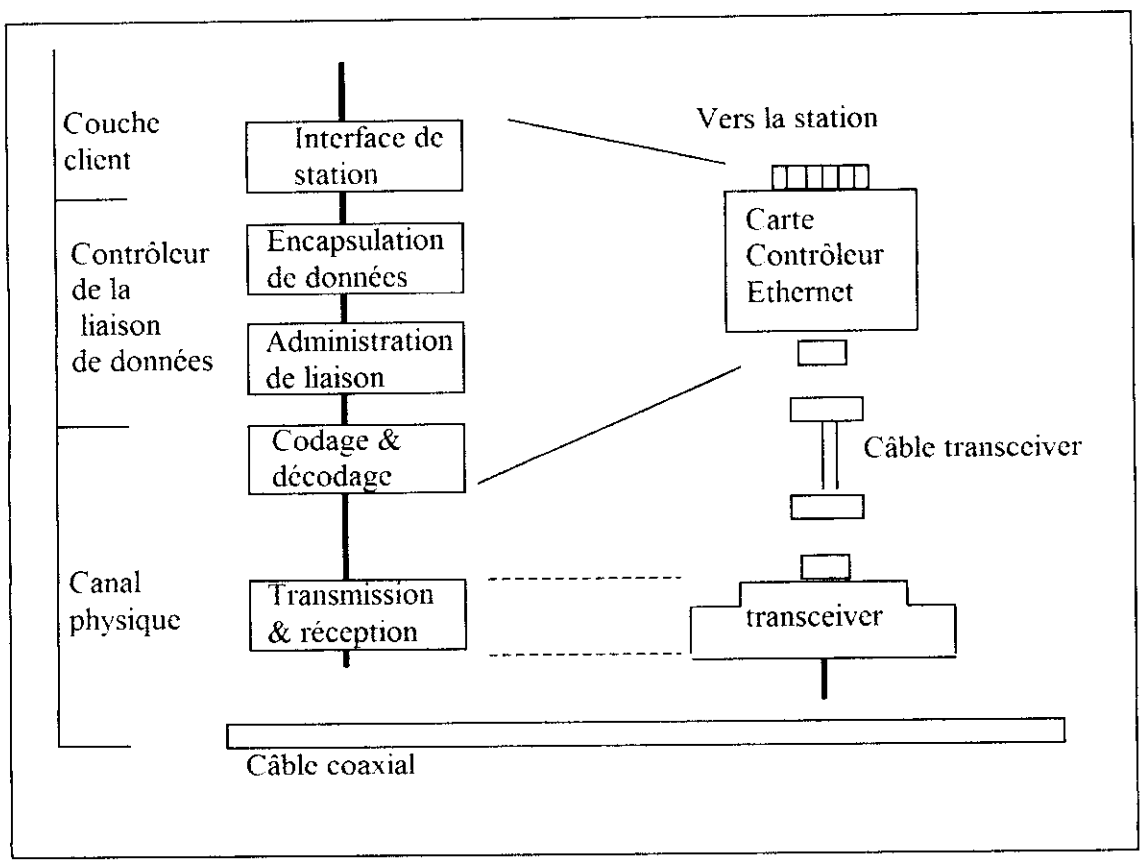


Figure 4.1 : Architecture et implémentation typique d'Ethernet

2.2. Le standard IEEE 802.3

L'IEEE s'empare donc des spécifications Ethernet et les reformule en 1985 dans la publication d'un standard : ANSI/IEEE 802.3 (IS/DIS 8802-3). Il s'agit alors d'un document ,certes in ingrat et donc d'abord difficil, mais complet et qui ne laisse plus d'ambiguïté .Le parallélisme avec le modèle OSI est établi, et l'on divise donc la couche 2 en deux pour insérer Ethernet dans les couches MAC et physique.

Les différences entre Ethernet V2.0 et IEEE802.3

La principale différence réside dans le fait que le 3eme champ MAC de la trame, le champ type, a changé de signification, et est devenu un champ longueur. Les adresses sont dotées d'un seconde bit particulier Individual/Universal, placé après le bit Unicast/Multicast.De plus les adresses peuvent théoriquement avoir 16 u48bits et non plus seulement 48. Le brochage du câble AUI (Attachment Uni Interface) s'est enrichi d'une paire supplémentaire: controlout(très rarement utilisée) et d'un blindage individuel par paire, ...etc.

2.3. La norme ISO 8802-3

La normalisation date de février 1989, ce qui est quand même assez éloigné de la parution d'Ethernet V2.0, et montre qu'une normalisation effective peut être tardive, et donc en retard sur le marché .Une nouvelle éditions de l'ISO 8802-3 a été publiée en 1993.

3. La méthode d'accès

La méthode d'accès constitue la principale procédure qui est à la charge de la couche MAC. Ethernet est basé sur le CSMA/CD qui est décrit en langage PASCAL dans la norme.

3.1. Le principe du CSMA/CD

La méthode CSMA/CD consiste à laisser chacun libre de gérer ss émissions en fonction de ses besoins et de la disponibilité du média.
En l'absence de trafic à transmettre, la station reste silencieuse et écoute (ou reçoit) les paquets qui circulent sur le câble .Quand la machine a besoin de parler (d'émettre), elle va agir indépendamment des autres, puisqu elle ne sait rien d'elles .Chaque machine ayant à tout instant la possibilité de débuter une transmission de manière autonome, la méthode d'accès est distribuée, et elle est dite à accès multiple (Multiple Access: Ma).
La machine observe donc le média en cherchant à détecter une porteuse (Carrier Sense:CS).Si aucune trame n'est transmit, elle ne trouve pas de porteuse (média et liber). Elle envoie donc es paquets sur le support physique, mais reste cependant à l'écoute du résultat de son émission pendant quelque temps, afin d vérifier qu'aucune autre machine n'a suivi le même comportement qu'elle, au même instant.
Dans le cas de transmission simultanément, les signaux brouillent, et sont perdus pour tout le monde. La méthode d'accès étant à détection de collision (Collision Detect: CD).
De façon à minimiser le risque de rencontrer une deuxième collision avec la même machine, chacune attend pendant un délai aléatoire avant de tenter une nouvelle émission .Cependant, de manière a ne pas saturer un réseau qui s'avérerait déjà très chargé, la machine n'essaiera pas indéfiniment de retransmettre un paquet, le paquet est éliminé après un certain nombre d'essais infructueux.
On peut parvenir à un état de blocage complet du réseau en cas de trafics trop élevés.

3.2. Le fonctionnement du MAC

La couche MAC a pour fonction de gérer le CSMA/CD, mais aussi de communiquer avec la couche supérieure pour lui offrir ses services, ainsi le MAC doit savoir formater les paquets de

données en trames, reconnaître son adresse dans le champ destinataire, et vérifier la validité des trames en réception.
MAC encapsule les informations u niveau supérieur dans une trame, en plaçant le paquet qu'on lui demande de transmettre dans le champ de données de la trame. Il remplit les autres champs MAC à l'aide des indications accompagnant les données. Ainsi, il renseigne l(adresse physique de la station destinataire avec les informations de la couche 3 , l'adresse source à partir de l'adresse physique de sont propre coupleur,le type en identifiant la couche pour la quelle il travaille , ou la longueur en comptant le nombre d'octets qu'on lui confie.
Enfin, il calcule le champ de contrôle de parité élaboré qu'est le CRC, et qui termine la trame.
Sur le diagramme suivant, extrait de la norme, on trouve les grandes étapes de l'émission d'une trame:

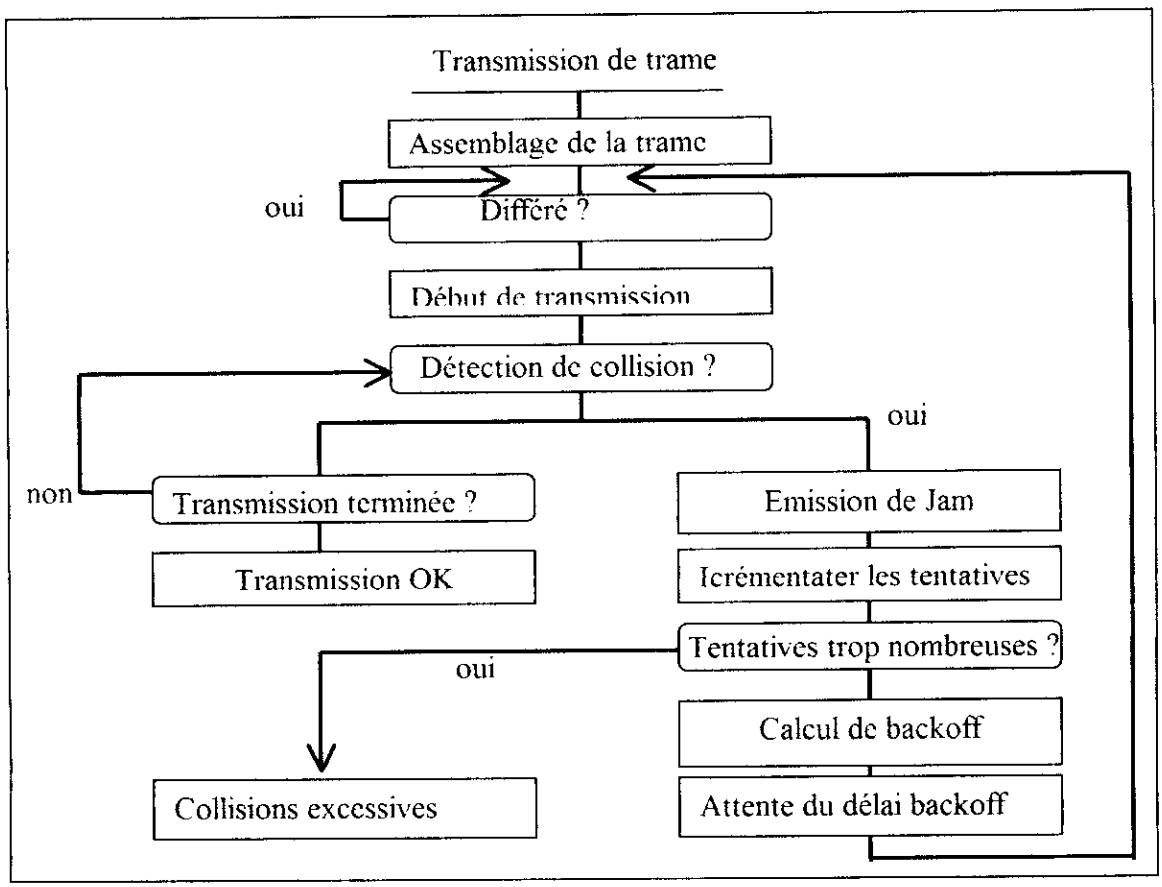


Figure 4.2 : Bloc – diagramme de l'émission d'une trame [1].

En réception, il est vérifier que la suite de bits peut composer une trame correcte .Pour cela un certain nombre de critères sont employés .Le nombre totale de bits ne doit pas être trop faible. L'adresse destination doit être l'adresse physique de la station, ou concerner un groupe dont la station fait partie .La trame ne doit pas être trop longue. Le calcule du contrôle de parité sur les champs reçus doit donner un résultat identique à la valeur transportée par la trame .Le nombre total de bits reçus doit être divisible par huit, sinon la trame restituée et non significative.

D'autres méthode dérivées du CSMA, comme le CSMA/DCR(Deterministic Collision Resolution), permettent de garantir une limite du délai avant émission .Dans le cas du

CSMA/DRC cela est rendu possible par une procédure déterministe de résolution de conflit ,et non probabiliste.

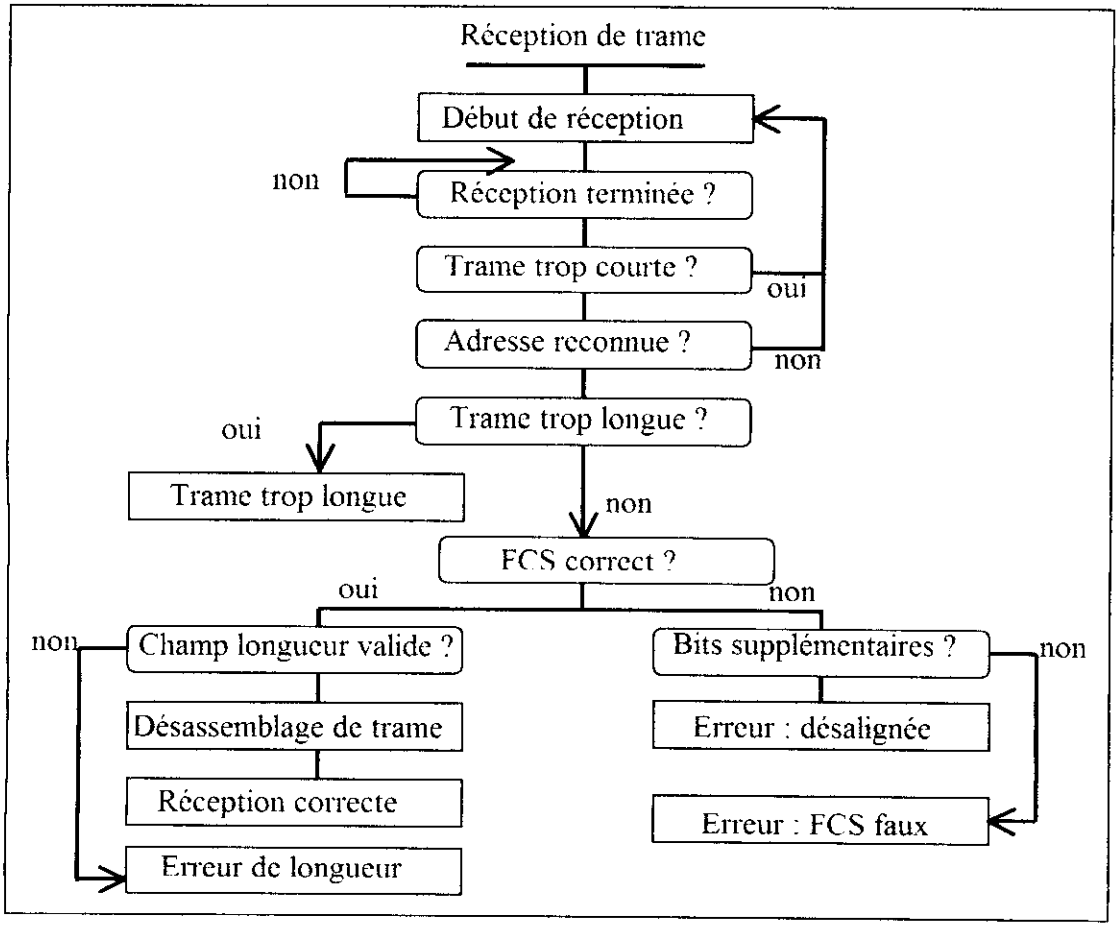


Figure 4.3 : Bloc-diagramme de la réception d'une trame [1]

4. La couche physique

La couche physique définit toutes les paramètres électrique et mécanique relatifs au support de communication. Elle décrit aussi les éléments responsables de transmission, de la réception et de la génération des signaux sur le média (les tranceivers et les répéteurs).
Le code employé par Ethernet est toujours de types Manchester, que ce soit sur câble coaxial, sur paire torsadée, sur fibre optique ou sur le câble AUI. Mais les niveaux électriques sont différents; dans le premier cas la composante continue est non nulle, et le signal évolue entre 0 et -2 volt, dans le second cas le signal est symétrique et évolue entre +/-1 volt.

De manière générale, le spectre de puissance de signaux Ethernet possède ses principaux pics de fréquence entre 5 et 10 Mhz (des harmoniques étant présentes de part et d'autre).Les chiffres d'atténuation des média sont donc données sur la plage de fréquences allant de 5à10Mhz.
Une unité de base de comptage du temps est souvent employée dans les normes Ethernet, il s'agit du bit-time, qui représente le délai équivalent à l'émission d'un bit, soit encore 0.1 us =100 ns (à

10Mbit/s).Une deuxième unité de temps est le slot-time. Il s'agit de la durée minimum d'une trame valide (512 bits).

5. Le formatage des paquets et les erreurs

Les données transitant sur un réseau Ethernet sont encapsulées dans une entité appelée trame. Lorsque divers problèmes se produisent sur le réseau (tels que les collisions), les fragments de paquets en résultant peuvent ne plus avoir les caractéristiques d'un signal valide, ce sont des erreurs.

5.1. La trame

La trame est la structure élémentaire permettant de faire transiter des données sur le réseau. Elle est définie au niveau MAC, et suit un certain nombre de règles. Ainsi les champs qui la composent identifient la station émettrice, la station destinataire, et le type des données transportées (indiqué par le protocole de niveau3). Elle possède de plus un contrôle de parité élaboré qui est placé sur les quatre derniers octets de la trame.
La transmission se faisant en mode série sur le média, les données présentées sous la forme d'octets, qui sont transis dans l'ordre .Tandis que les bits de chaque octet sont émis de celui de poids le plus faibles à celui de poids fort.

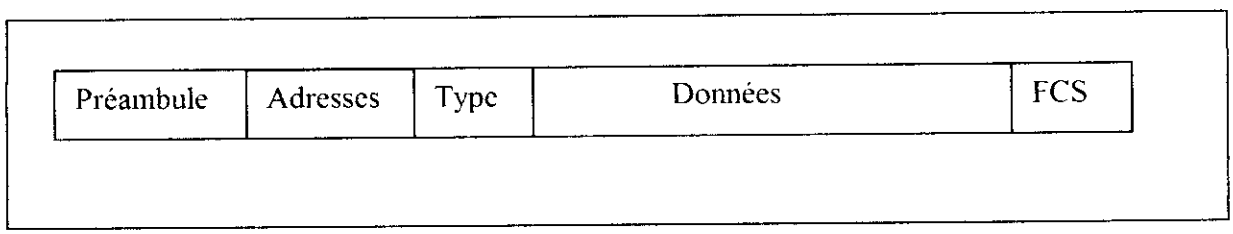
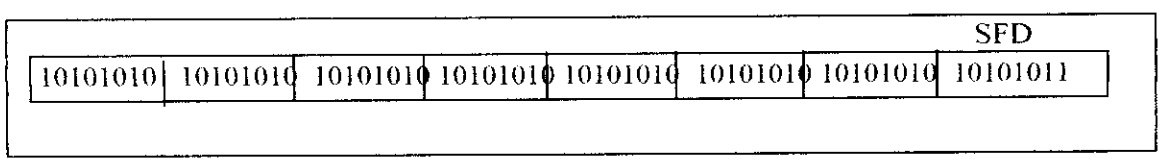


Figure 4.4 : Les différents champs de la trame Ethernet

5.1.1. Préambule

En réalité, le préambule précède la trame et permet à l'horloge du récepteur de synchroniser sur l'horloge de l'émetteur (on n'oubliera pas que le mode de transmission est asynchrone).
Etant envoyé pour assurer la stabilisation de circuits de décodage, il est prévu qu'une partie du préambule puisse être perdue.
Le préambule est formé d'une suite de bits successivement à 1 et 0, et il est par la même dénué de toute information spécifique .Il peut être décomposé en deux sous-champs, un premier de 56 bits alternativement à 1et 0, suivi d'un sous champ nommé SFD(Starting Frame Delimiter),délimiteur d début de trame, qui fait huit bits de long, en continuant la séquence précédente, excepté le tout dernier bit qui est à 1.Ce doublé à 1 permet au récepteur de savoir que commence véritablement la trame et que les bits qui suivent composent donc des champs significatifs.
Pour conclure, voici la représentation du préambule complet, sachant qu'il est transmis de gauche à droite:



5.1.2. Les adresses MAC

La trame comporte deux adresses, qui sont l'adresse du destinataire et l'adresse de l'émetteur. L'adresse destination est transmise la première, puis vient l'adresse source.
La raison d'être de ces champs est bien sûr de permettre à la machine visée par le message de se connaître comme destinataire, mais aussi de pouvoir identifier la machine ayant généré la trame circulant sur le réseau.
Les deux adresses ont des structures à peu prés semblables .Elle font 6 octets de long .Les trois premiers octets identifient de façon biunivoque le constructeur de la carte coupleur .Les numéros de ces trois octets sont attribués aux constructeurs par une organisation unique au niveau mondial, ce qui assure la cohérence du système.
Les trois octets suivants donnent le numéro du coupleur chez ce constructeur (256^3 possibilités=16.78 millions).L'ensemble forme donc un numéro unique pour chaque machine ne comportant qu'une carte Ethernet.
Les adresses Ethernet sont représentées de manière conventionnelle en base hexadécimale, avec un tiret séparant chacun des 6 octets.

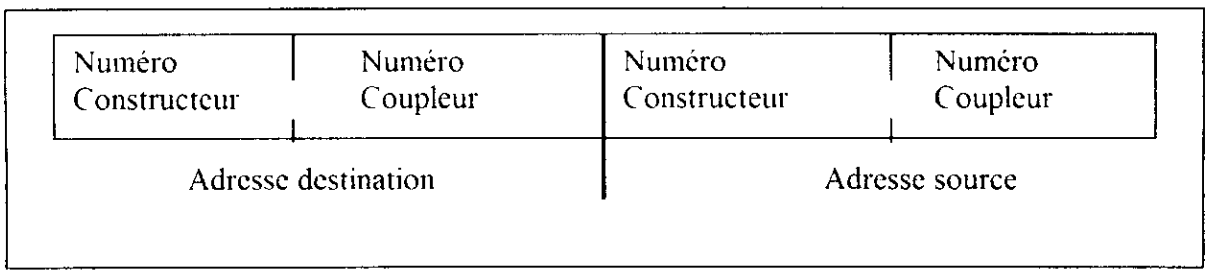


Figure 4.5 : Position et découpage des deux adresses MAC

En fait, ce que l'on vient de n'est véritablement applicable que lorsqu'il s'agit d'envoyer une trame vers une seule machine, car Ethernet permet également d'atteindre plusieurs cibles simultanément, en faisant ce qui s'appelle de la diffusion .il y a le choix entre émettre vers tout le monde (diffusion générale nommée Broadcast en anglais), ou vers un groupe de stations (multicast en anglais).
L'adresse de destination broadcast est FF-FF-FF-FF-FF-FF, en hexadécimal (ce qui correspond à une suite de 1 en binaire), alors que l'adresse de groupe a pour caractéristique le premier bit transmis à 1(soit un premier octet impair) tandis que les autres bits des trois premiers octets conserver le numéro du constructeur dont les machines sont visées .Ainsi, le groupe est nécessairement celui des machines d'un même constructeur.
Il est à noter qu'une adresse source correspondant à l'adresse exacte de la carte coupleur qui émet, elle ne peut logiquement pas avoir de bit de groupe à 1, et donc que son premier octet est nécessairement pair.
Dans la structure IEEE 802, le premier bit d'une adresse(longue ou courte) est dénommé Individual/Group(I/G).Cependant, pour les adresses longues, le deuxième bit transmis a aussi une signification particulière : il indique si l'adresse est de type universelle ou si elle est administrée localement .dans la second cas, les 46 bits qui suivent sont choisis par l'utilisateur, et ne sont pas nécessairement les numéros du constructeur et du coupleur,ce second bit est dénommé Universally/Locally administered (U/L).

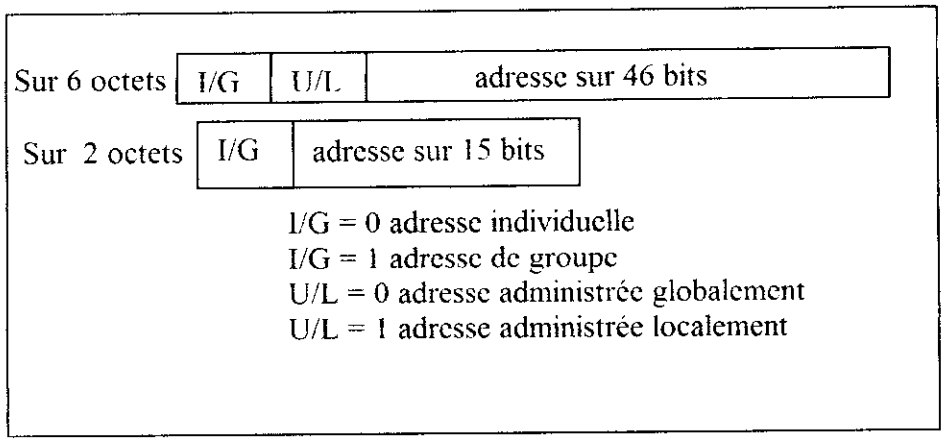


Figure 4.6 : Format des adresses IEEE 802.3

5.1.3. Le champ type/longueur

Ce champ de deux octets a été défini dans le standard Ethernet pour indique le type de protocole de niveau 3 employé pour transporter le message.
Ce pendant, ce champ a changé de signification avec la normalisation (IEEE 802.3 puis IS 8802-3) et comporte, selon la norme, une information de longueur sur le champ de données (le champ suivant).
Cette information possède toute fois un intérêt, soit quant le champ de données n'est pas entièrement rempli, soit pour vérification de la cohérence entre la longueur totale de la trame reçue et le nombre d'octets communiqué par ce champ.
Quoi qu'il en soit, la question importante qui se pose est de savoir comment les trames Ethernet peuvent être différenciées des trames IEEE 802.3 sur le réseau.
Les deux peuvent parfaitement coexister, ce qui est souvent le cas dans la réalité .On sait que la longueur du champ données est compris entre 46 $[(2E)_H]$ et 1500 $[(5DC)_H]$.On décide donc que si le contenu du champ suivant les deux adresses dépasse 5DC , il s'agit d'un champ type et donc d'une trame Ethernet , dans le cas contraire, il doit s'agir d'un champ longueur et d'une trame IEE 802.3.

5.1.4. Les données

Le champ de données contint le paquet de niveau LLC ou 3. Il ne possède donc pas de signification propre pour Ethernet (au niveau MAC). Le champ est vu comme une suite de 46 à 1500 octets .Le seul traitement effectué sur les données sera le calcul du CRC.
Enfin, si mois de 46 octets sont fournis par la couche supérieure, le champ de données est complété par le PAD.

5.1.5. Le PAD

Le PAD ou séquence de bourrage, ne sert qu'à remplir le champ de données pour obtenir au moins 46 octets.
Dans le cas d'Ethernet, le niveau MAC de la trame ne véhiculait aucune information sur le nombre d'octets de données. La différentiation entre les octets utiles et le remplissage devait donc réalisée par les couches supérieures. Avec IEEE 802.3, le champ longueur indique si le champ de données contient un PAD, et quelle est la longueur de celui-ci.

5.1.6. Le FCS

FCS signifie Frame Check Sequence, soit séquence de vérification de trame .Il s'agit donc d'un champ de quatre octets placés en fin de la trame et permettant de valider l'intégrité de la trame après la réception.

Il utilise un CRC (Cyclic Redundancy Check, c'est -à–dire un code de redondance cyclique) calculé à l'aide d'un polynôme générateur de degré 32 .Il englobe les deux champs adresses, le champ type/longueur et les données (PAD compris), et sert donc à la station réceptrice pour décider si la trame est parfaitement correcte et peut être transmise à la couche supérieure (LLC ou niveau 3). Un détail au sujet de la transmission du FCS : il est le seul champ de la trame à être transmis en commençant par le bit de poids fort (coefficient de X^{31} en tête, coefficient de X^0 en queue).

5.1.7. Le temps inter-trames

Une machine ne peut émettre toutes les trames qu'elle a à transmettre les unes à la suite des autres. Il est obligatoire qu'un temps d'au moins 9.6 us soit laissé entre la chute du signal occupant le média et le début de la trame émise, c'est le délai inter-trame ou inter-frame gap (ou spacing) en anglais.

Ce silence permet aux circuits électroniques de récupérer l'état de repos du média (absence de signal), et éventuellement, aux autres stations désirant transmettre de prendre la main à ce moment-là.

5.1.8. Résumé

pour reprendre les éléments importants de la composition d'une trame , nous allons donner ci-dessous une représentation de la trame type avec ses différents champs.

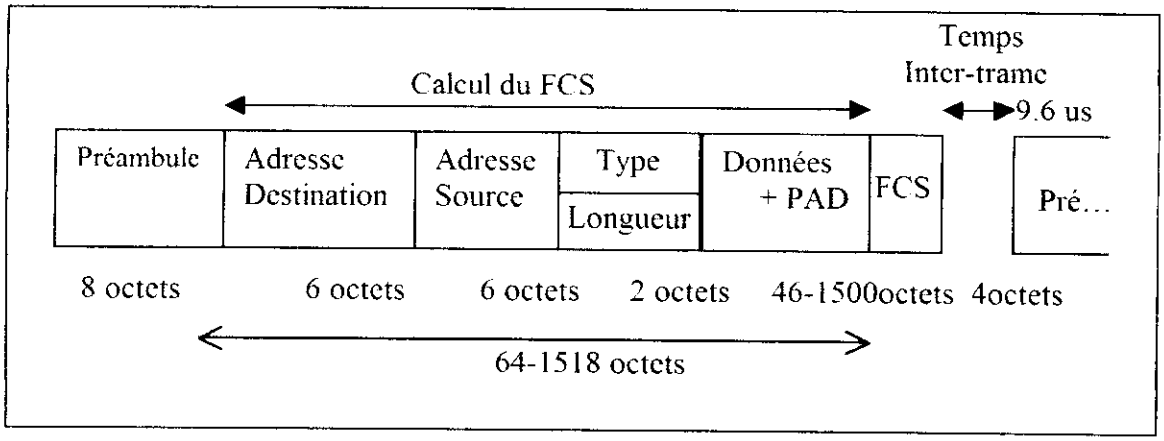


Figure 4.7 : Vu schématique de la structure d'une trame complète

5.2. Les trames erronées

Des paquets ne respectant pas du tout, ou imparfaitement, la structure d'une trame cohérente peuvent se trouver sur le média .Ils peuvent être le résultat d'incidents tels qu'une collision, le débranchement brutal d'une machine, ou le mauvais fonctionnement d'un ou plusieurs matériels du réseau .Dans tous les cas, au moins un des champs du message enfreint une des règles de constitution d'une trame. Une terminologie précise peut donc être utilisée pour désigner chaque paquet erroné.

En effet la couche MAC de chaque station élimine immédiatement les trame incorrectes,et ne garde trace d'aucun champ du paquet erroné (sauf certains logiciels internes de surveillance et de statistiques).

5.2.1. Le Runt

Le Runt est le terme anglais employé pour désigner une trame trop courte, c'est-à-dire faisant moins de 64 octets.
Dans la plupart des cas, il s'agit d'une trame tronquée pour une raison ou une autre (souvent à cause d'une collision), et ce qu'il reste de la trame d'origine n'a plus de signification .En effet on ne sait pas nécessairement si l'on a reçu le champ de données complet et si les quatre derniers octets sont ceux de CRC. Cette trame pouvant réellement résulter d'une collision, sa présence est parfaitement acceptable sur un réseau sain. Cependant, dans ce cas, la trame Runt sera généralement désalignée (nombre d'octets non entier) et avec un CRC faux (bad FCS).

5.2.2. Le Jabber

Le Jabber est une trame trop longue, qui possède donc plus de 1518 octets. Ce type d'erreur ne doit théoriquement jamais se produire sur un réseau sain.
Cependant, si la trame contient entre 1500 et 3000 octets, ou bien plusieurs dizaines de milliers d'octets, les causes qui sont à son origine différent.
Dans le premier cas, on peut supposer qu'il s'agit de superposition de deux trames longues entrées en collision sans que cela n'ait été détecté .le fait qu'une collision soit passée inaperçue est révélateur d'un problème important .Certes, une trame peut être complètement perdue lors de son entre dans la couche MAC (si son émission subit 16 collision successives).
Dans le second cas, le paquet n'a probablement pas une structure de trame, et il doit être produit par un composant défectueux qui reste beaucoup trop longtemps en état d'émission.
Cette panne doit être localisée et réparée rapidement car elle peut être très pénalisante pour le réseau.

5.2.3. La trame désalignée

La trame désalignée, ou misaligned en anglais, est une trame dont le nombre de bits n'est pas divisible par huit, et qui ne peut donc pas être restituée sous le format d'une suite d'octets entiers.
Dans la pratique la trame désalignée peut avoir une longueur quelconque (de 64 à 1518 octets, plus ou moins quelques bits), mais possède presque toujours un CRC faux.

5.2.4. Le bad FCS

La trame ayant un bad FCS (soit, en français, un CRC erroné) est une trame pour la quelle le CRC recalculé par la machine réceptrice, ou lectrice, ne correspond pas aux quatre derniers octets de la trame reçue.
Cela peut se produire quand un ou plusieurs bits de la trame sont faux (pour cause de mauvaise transmission, d'interférence, etc.), et l'on se trouve alors dans le cas où le calcul de redondance trouve toute sa raison d'être.
En conclusion, la trame avec bad FCS est soit une trame complète dont au moins un bit n'a pas été reçu tel qu'il avait été transmis, soit un reste de collision.

5.3. La collision [1]

La collision est le phénomène de la superposition de deux signaux(à priori deux trames) sur le média. La collision ne se produit bien entendu que si deux émetteurs sont entrés en action simultanément (ou dans un laps de temps suffisamment court).

En effet en cherche à éviter que la collision passe inaperçue, et que le réseau, alors incapable de détecter le problème survenu, ne récupère pas la trame, ou ne remonte pas l'information à la couche supérieure.
Il existe un moyen d'éviter ce genre d'événement difficile à gérer .Pour cela il faut s'assurer que la durée de transmission d'une trame est toujours supérieure à deux fois le temps de transit entre les deux points d'émission.

5.3.1. L'algorithme de backoff

La méthode d'accès employée pour Ethernet est de types CSMA/CD persistant. De plus celle-ci est capable de reprendre son processus de transmission après l'occurrence de collisions. Comme nous allons le voir, la méthode d'accès d'Ethernet gère les tentatives de retransmission après l'incident avec une répartition aléatoire dans un intervalle de temps croissant.
Précisément, Ethernet fait appel à l'algorithme de backoff pour traiter les collisions lors de tentatives d'envoi de trame. Lorsque l'émission d'une trame est perturbée par une collision, la machine produite un Jam, puis cesse toute émission en attendant que le média redevienne libre. Après un moment, choisi aléatoirement de manière à ne pas redémarrer simultanément avec son "concurrent", elle tente de nouveau d'envoyer son message, et boucle donc sur la même procédure d'émission que l'instant précédent. Si un problème grave empêche toute transmission, le niveau MAC d'Ethernet prévu pour que la station arrête sa série de tentatives, et cela tout simplement en éliminant la trame à émettre de ses buffers après 16 essais infructueux.
Par ailleurs, si les collisions sont dues à la charge du réseau, plus les stations cherchent à ré --- émettre rapidement leurs données, plus elles chargent le réseau et entraînent un effet d'avalanche tendant vers un blocage complet .contre cette éventualité, Ethernet a prévu une attente choisie aléatoirement dans une fenêtre croissante de délais possibles. Ainsi, la fenêtre de choix est de 2 possibilités (0 ou 1 slot-time) après la première collision, et croît d'un facteur deux a chaque nouvelle tentative. Enfin, après la dixième tentative, cette fenêtre reste bornée par 1023 slot – time, qui est déjà une valeur importante (1024 x 512 x 0.1us =52,4ms). Les se poursuivent donc, jusqu'à la seizième si besoin, avec cette même fenêtre de délai.

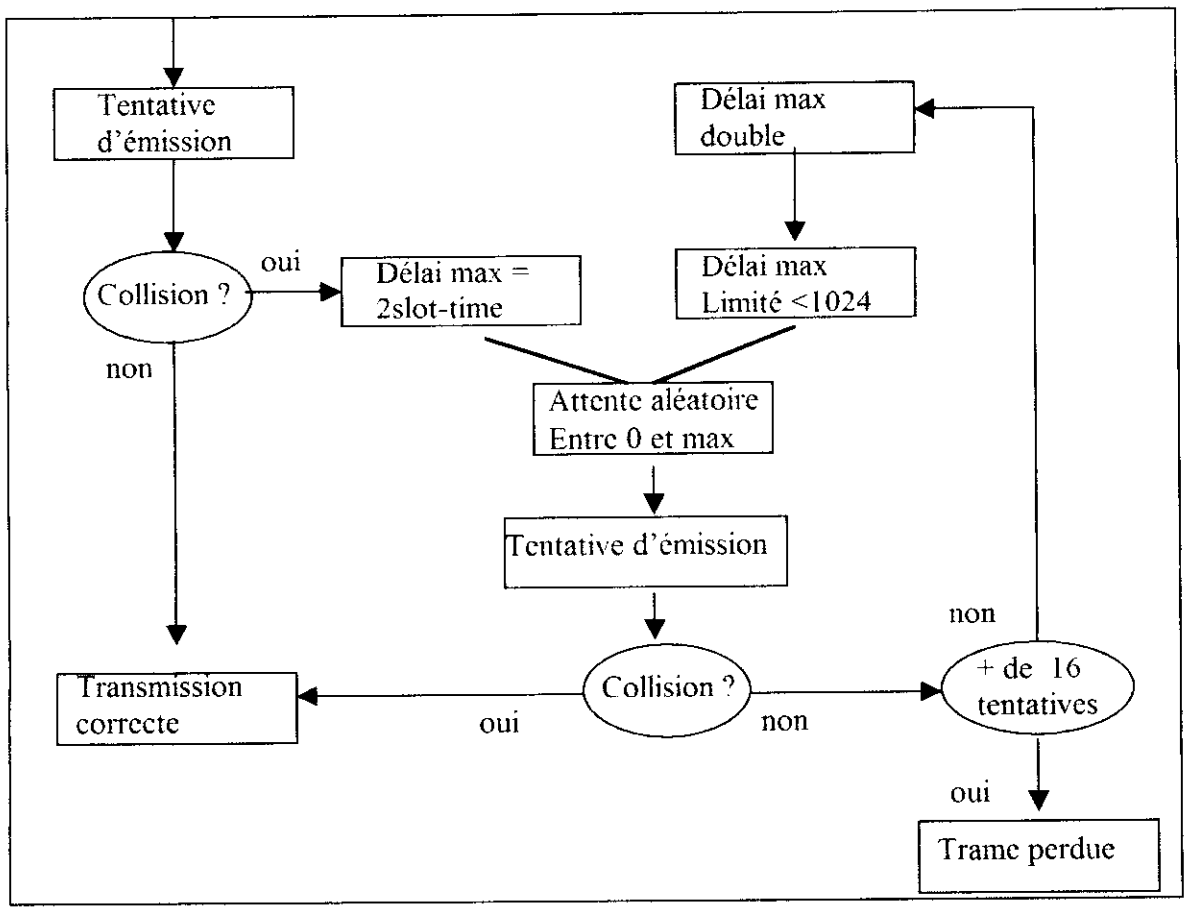


Figure 4.8 : Bloc-diagramme résumant le fonctionnement du backoff [1]

5.3.2. Le Jam

Le Jam est un signal sans signification intrinsèque, mais qui assure aux éléments ayant détecté une collision sur leurs émissions que le reste du réseau a bien perçu cette collision. Ceci est réalisé tout simplement en faisant durer la collision au moins 32 bit-time, par l'émission d'une suit de bits quelconques à partir de l'instant où l'émetteur détecte une collision.
Il est noter qu'il est tout de même demandé que la valeur de ces 32 bits soit différente de CRC qui correspondrait au début de trame déjà transmis, ceci pour éviter toute confusion. Dans la réalité la séquence transmise est toujours la même. Ce signal est aussi appelé " enforcement de collision ", ce qui évoque bien son objectif.

5.4. L'idle

L'idle est un signal d'attente permettant d'occuper volontairement et régulièrement le média pendant les périodes d'inactivité. Ce signal n'est utilisé que sur les liaisons en point à point (paire torsadée et fibre optique) et fournit une information sur la présence d'un élément actif à l'autre extrémité. Il s'agit d'une sécurité visant à protéger le réseau d'une panne sur une liaison. Ainsi une machine ignorant que la paire réception de son transceiver est défectueuse pourrait émettre sans respect du CSMA et provoquer de nombreuses collisions.

6. Conclusion

Pour construire une trame, il faut connaître ses différents champs ; et les faire remplir un par un, pour qu'ils puissent être envoyés sous un format Ethernet, par assurer cette émission avec un minimum de collision.
L'émission des trames peut se faire par un micro-contrôleur; et un circuit qui intègre les étapes précédentes. C'est ce qu'on appelle une carte d'interface Ethernet.

V. ETUDE DE PROJET

1. Introduction

Ce projet consiste en l'exploitation d'une carte comportant un microcontrôleur 68HC11de la famille Motorola, qui dispose d'une liaison série de type RS232, afin de construire une carte d'interface Ethernet. La carte permet des communiquer, a travers le réseau Ethernet, entre un client (lecteur carte), et un serveur (ordinateur), le système résultant permet contrôle d'accès. Il est nécessaire d'intégrer le composant CS8900A de Cirrus Logic qui se comporte comme un périphérique 8 bits intégrant la couche liaison en Hardware. Puis, il est convenu de développer des programmes permettant un contrôle à distance.

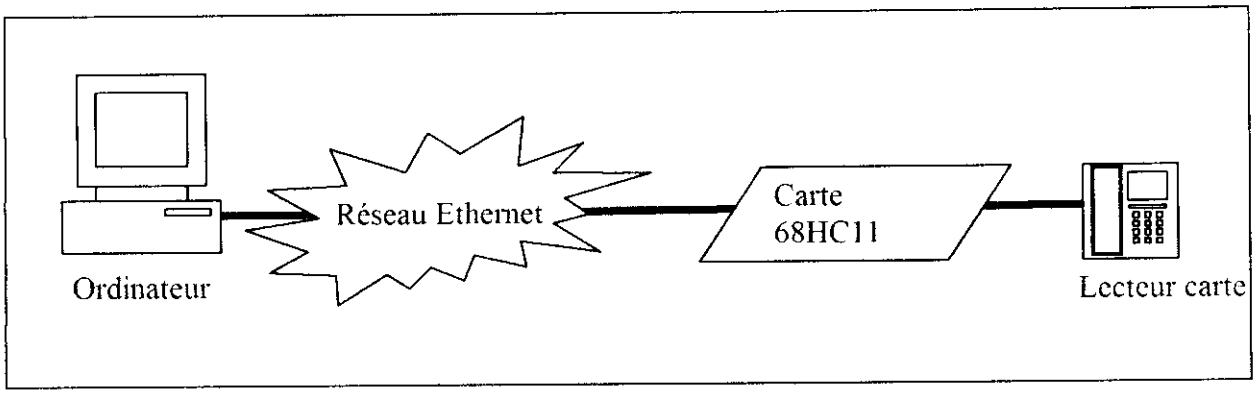


Figure 5.1 : Schéma du projet

Matériel disponible :

- carte 68hc811e2.
- RAM 32Ko.
- Port série +Max 232.
- Circuit imprimé de 68hc11 avec ses annexes existe sous Orcad.
- Connecteurs.

2. Partie hardware

La partie Hardware constitue la partie la plus difficile à développer, en raison du nombre important d'étapes avant l'obtention de la carte final.

2.1. Schéma de principe initial

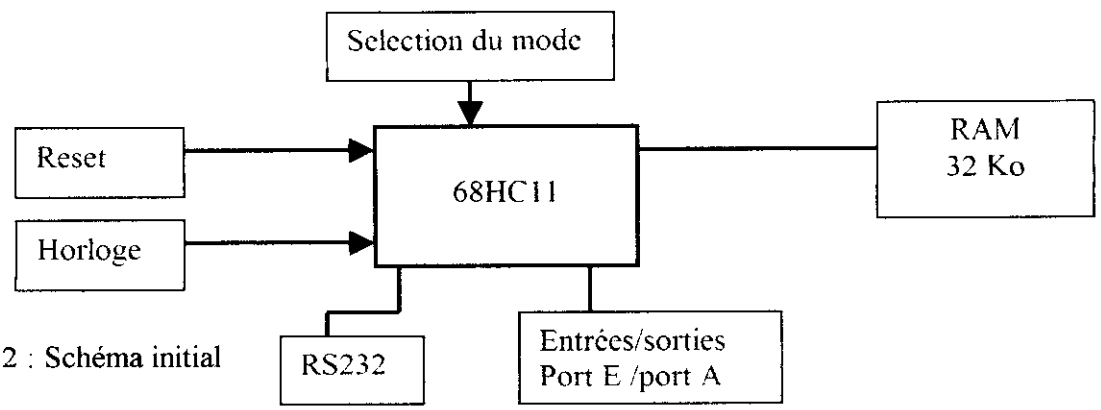


figure 5.2 : Schéma initial

La carte initiale est articulée autour d'un microcontrôleur 68HC11A8 de MOTOROLA sur laquelle est adressée jusqu'à 64K octets (16 Bits d'adresse : A0..A15) de ressources externes pour le mode étendu du processeur. Parmi ces ressources, la RAM de 32K est adressée initialement en $0000. Des interrupteurs permettent de configurer le microcontrôleur dans l'un de ses 4 modes (Single, Extended, Bootstrap ou Test).
L'interface RS232, MAX232, réalise l'adaptation en tension des signaux (des signaux polaires 0-5V en signaux bipolaires –12V/+12V) circulant sur la liaison série RS232, qui permet le chargement du programme dans le microcontrôleur 68HC11 et la communication avec un ordinateur.
Quant aux entrées / sorties du microcontrôleur, le port E est un port d'entrée 8 bits (PE0...PE7), les ports A, PA0...PA2 sont des entrées, et les ports PA3...PA7 sont des sorties.

2.2. Schéma électrique initial

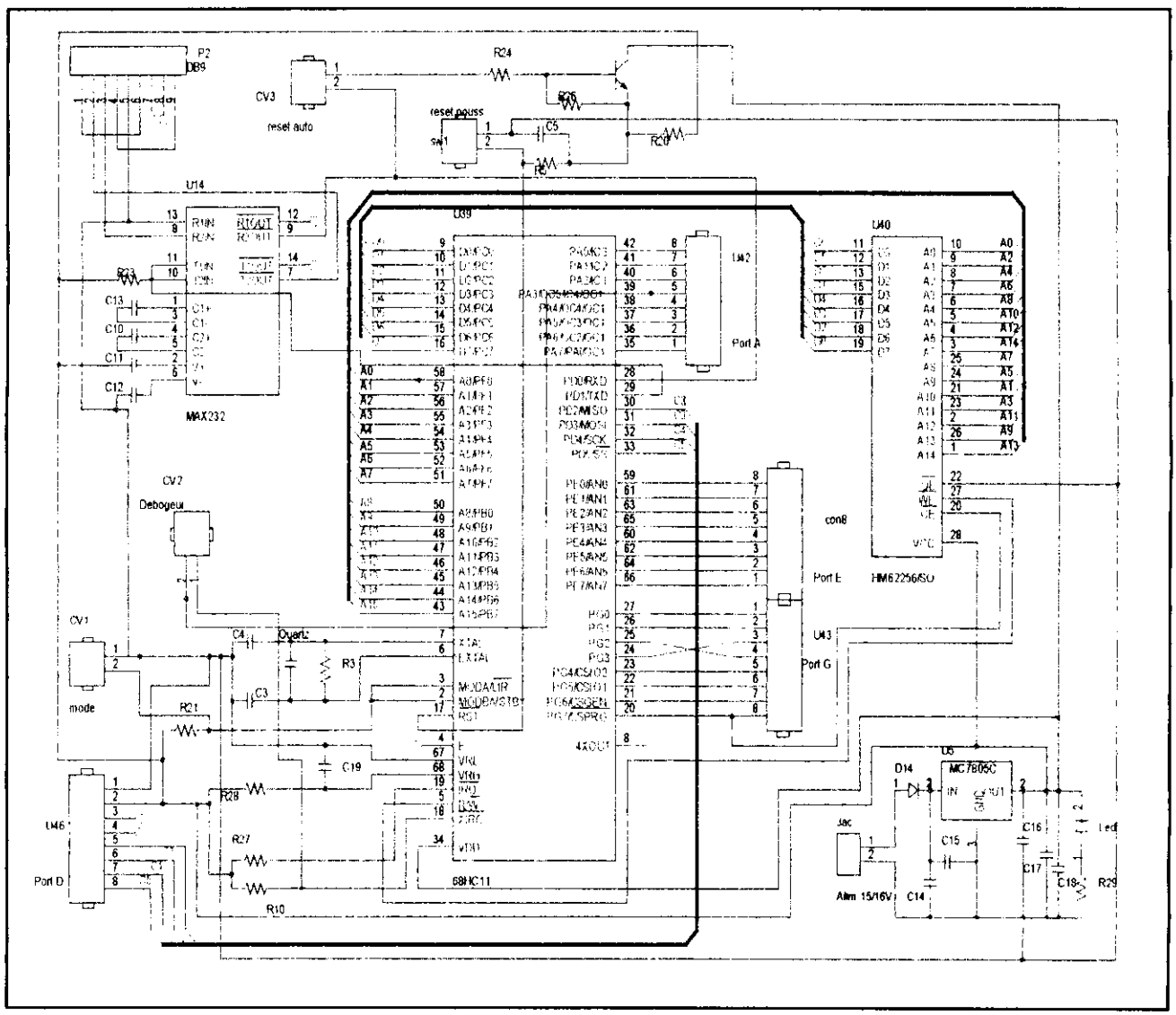


Figure 5.3 : Schéma électrique

2.3. Travail à effectuer

Le travaille a effectuer et d'ajouter l'interface Ethernet chip CS8900A adressée comme périphérique 8 bits. Il y a aussi le PLD qui permet le décodage d'adresse, et suivant le cas, il génère des signaux de sélection permettant au microcontrôleur d'effectuer ses accès mémoire avec le bon composant.

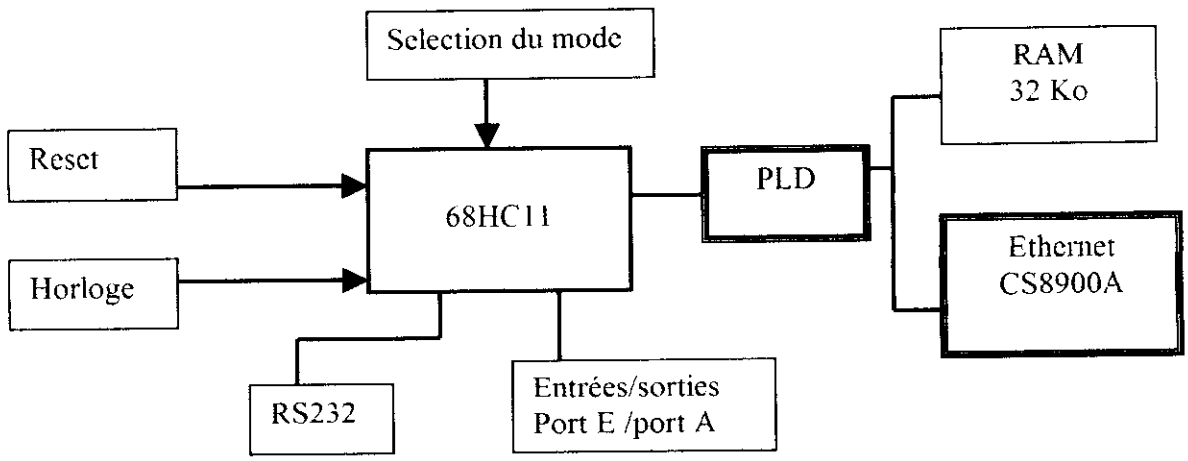


Figure 5.4 : Schéma complet

2.4. Décodage d'adresse

Interprétation

· Lorsque A15 vaut 0 , on accède à la RAM adressée au adresses $0XXX.
· Lorsque A15 vaut 1, et A14.A13.A12 vaut 000, c'est à dire pour les adresses débutant en $8000, on accède au chip Ethernet adressé aux adresses $8XXX.

D'ou le tableau suivant :

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RAM	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
CS8900A	1	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X

Se qui nous donne :

RAM = A15*.
CSEther = A15.

Cartographie Mémoire avec interface Ethernet

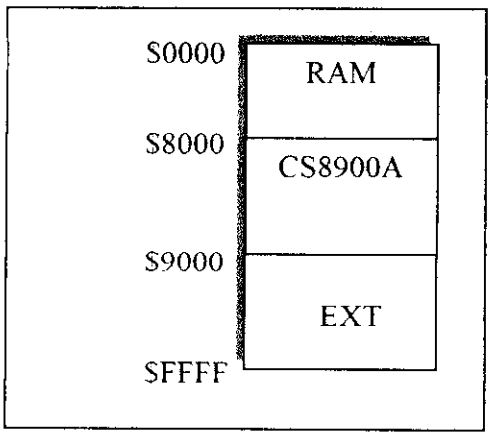


Figure 5.5 : Adressage mémoire des ressources externes

Sans oublier que suivant le mode de configuration du microcontrôleur on aura les mapping suivants :

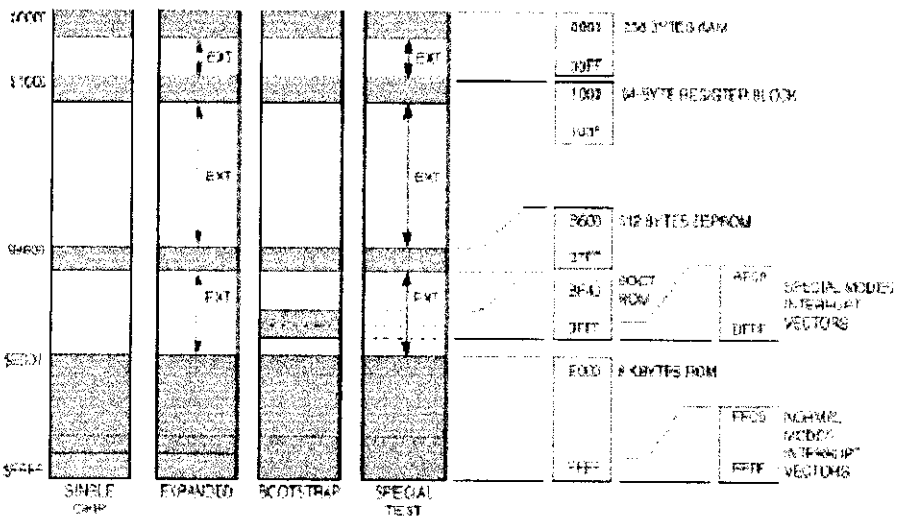


Figure 5.6 : Mode de configuration dc 68hc11

Circuit à logique câblée pour PLD

CE*=(A15*.E)*
AEN*=(A15.E)*
OE*=((A15.R/W).E)*
r/w*=(R/W*.E)*
IOR*=((R/W.E).(PA3*.A15))*
IOW*=((R/W*.E).(PA3*.A15))*
MEMR*=((R/W.E).(PA3.A15))*
MEMW*=((R/W*.E).(PA3.A15))*

Avec:
Les sorties CE (Chip Enable) : signal de synchronisation des échanges de données du 68HC11 en mode étendu.
Les sorties OE (Ouput Enable) sont générées à partir des CE et du signal de lecture afin de valider les sorties des données. Suivant l'état du signal PA3 (port A3 du 68HC11), c'est soit les signaux MEM soit les signaux IO qui sont générés pour les accès en écriture ou en lecture du CS8900A.

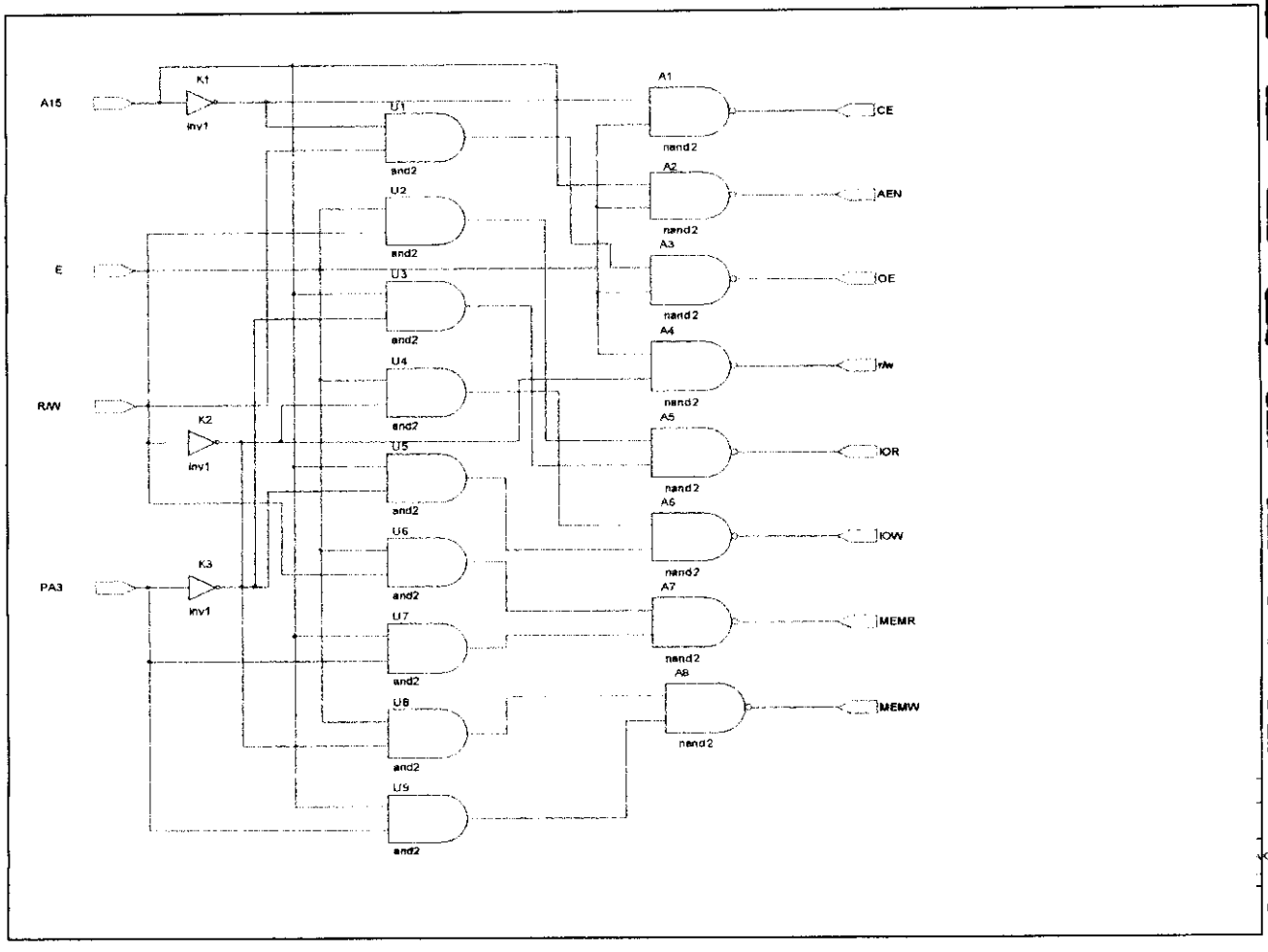


Figure 5.7 : Logique câblée (Actel 40MX)

2.5. Utilisation du CS8900A

Le chip CS8900A est optimisé dans sa conception pour un câblage sur des bus de type ISA Concernant notre cas de figure, le CS8900A doit pouvoir fonctionner en mode 8 bits (adressage des données sur le port C du 68HC11), et connecté à bus système qui n'est pas ISA. Il faut donc trier les signaux susceptibles d'interfacer correctement le chip avec le microcontrôleur 68HC11 :

- Les 8 bits de données du chip SD0...SD7 sont reliés au port C du 68HC11.
- SD8...SD15 sont à la masse. (puisqu'on est en données 8 bits)
- Les adresses SA0...SA11 (adressage jusqu'à 0FFF comprend facilement les 4Ko de données du chip Ethernet) connectées au bus d'adresse du 68HC11. (voir Packet Page Adresses)
- SA12...SA19 sont à la masse.
- ELCS* à la masse car LA non utilisé.
- EECS, EESK et EED Out non utilisés car pas de ROM.
- EEDI*, CHIPSEL* la masse.
- DMACK* en VCC car pas de DMA.
- CSOUT* non utilisé car pas de ROM de Boot.
- DO+/-, CI+/-, DI+/- non utilisés car l'AUI non utilisé.
- SLEEP à VCC car pas de Hardware Sleep ni Standby.
- RESET actif bas donc nécessité d'un inverseur à l'entrée.
- SBHE* à VCC car pas de transfert sur octet haut.
- IOCHRDY* non utilisé.
- INTRQ0 utilisé pour le mode MEM.
- AEN* sert de chip select du CS8900A.
- MEMW*, MEMR*, IOR*, IOW* sont les signaux de lecture et d'écriture sur le chip suivant son mode de fonctionnement.
- LANLED* et LINKLED* sont reliés à des leds lumineuses.
- XTAL 1,XTAL2 sont les broches pour relier le quartz de 20MHz.
- MEMCS16* et IOCS16* non utilisés car on est en mode 8bits.
- RES connecté à une résistance de 4.99K vers la masse.
- BSTATUS* non utilisé.
- TEST* non utilisé car pas de Boundary Scan Test.
- DVDD à 5V numérique.
- DVSS à la masse numérique.
- AVDD à 5V analogique.
- AVSS à la masse analogique.

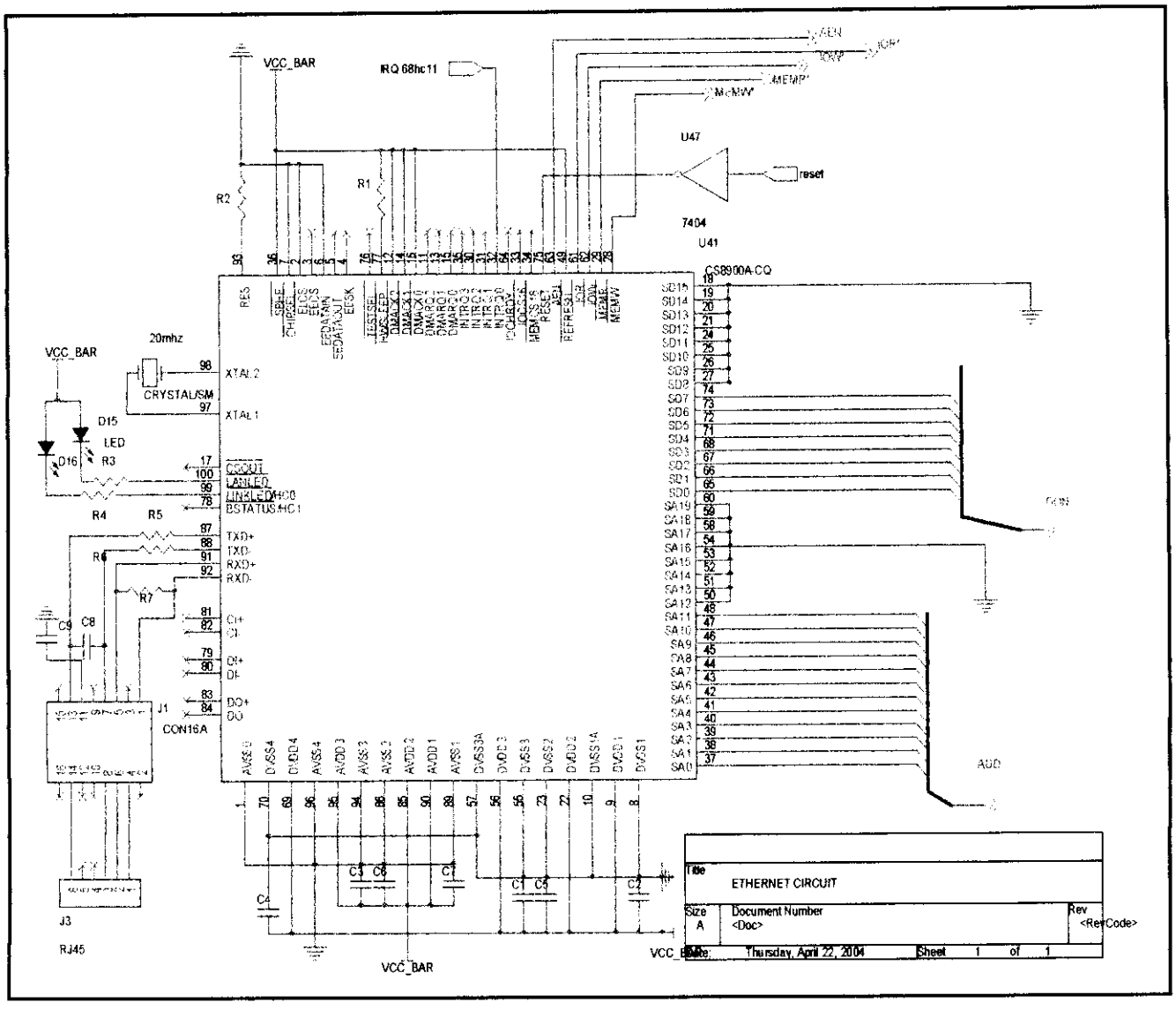


Figure 5.8 : Schéma électrique de CS8900A

2.6. Schéma électrique final avec interface Ethernet

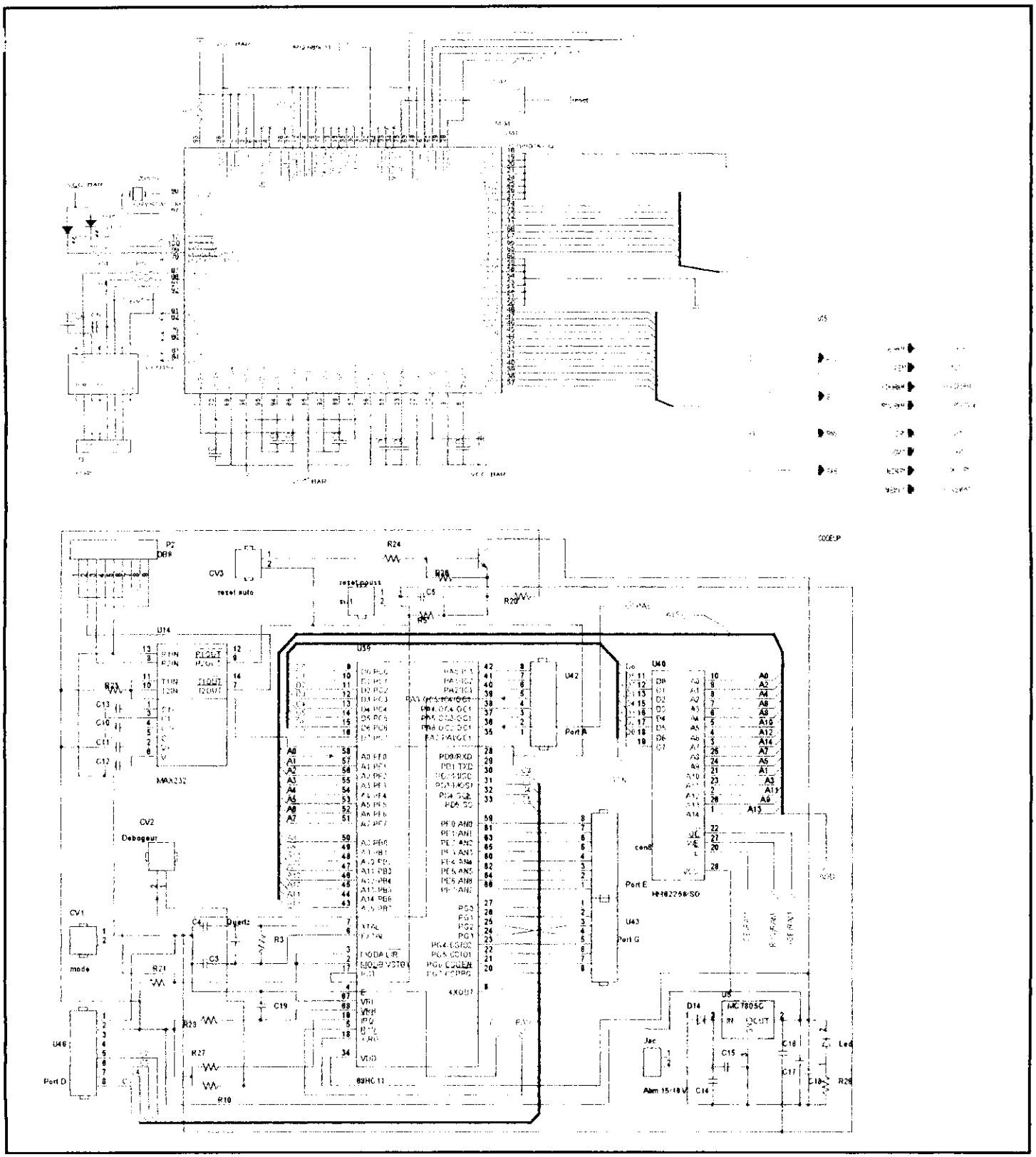


Figure 5.9 : Schéma d'ensemble avec le microcontrôleur et toutes ces ressources.

3. Partie software

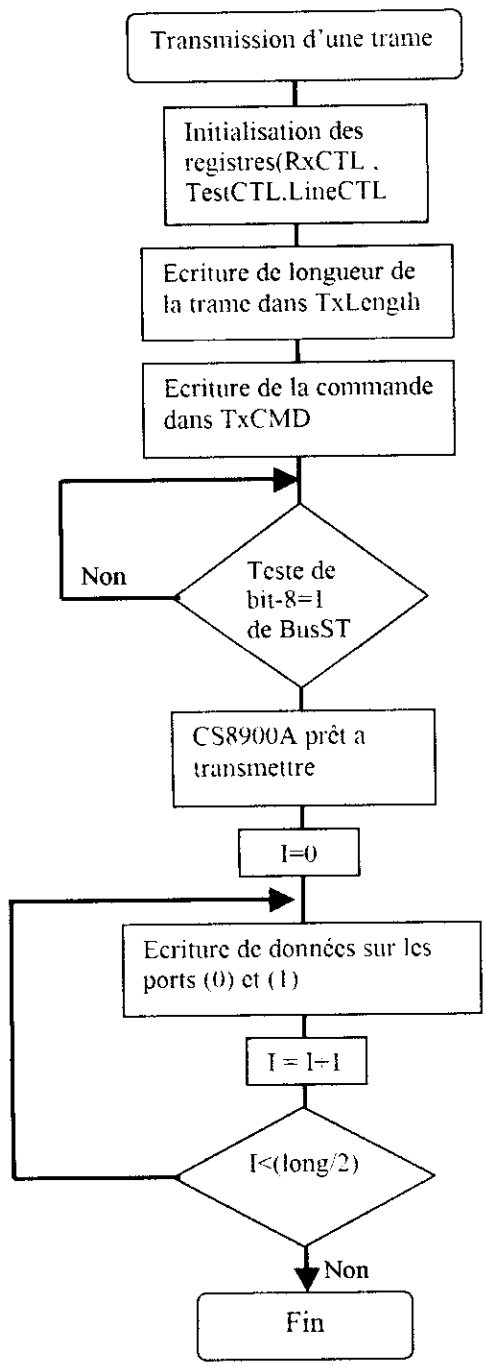
Nous allons maintenant donner les organigrammes concernant la transmission et la réception au niveau de la carte Ethernet et au niveau des couches supérieurs.

3.1. Au niveau de carte Ethernet 68HC11

Transmission d'une trame

Avant tout fonctionnement du CS8900A, on effectue une initialisation des registres de contrôle de transmission RxCTL, TestCTL et LineCTL.

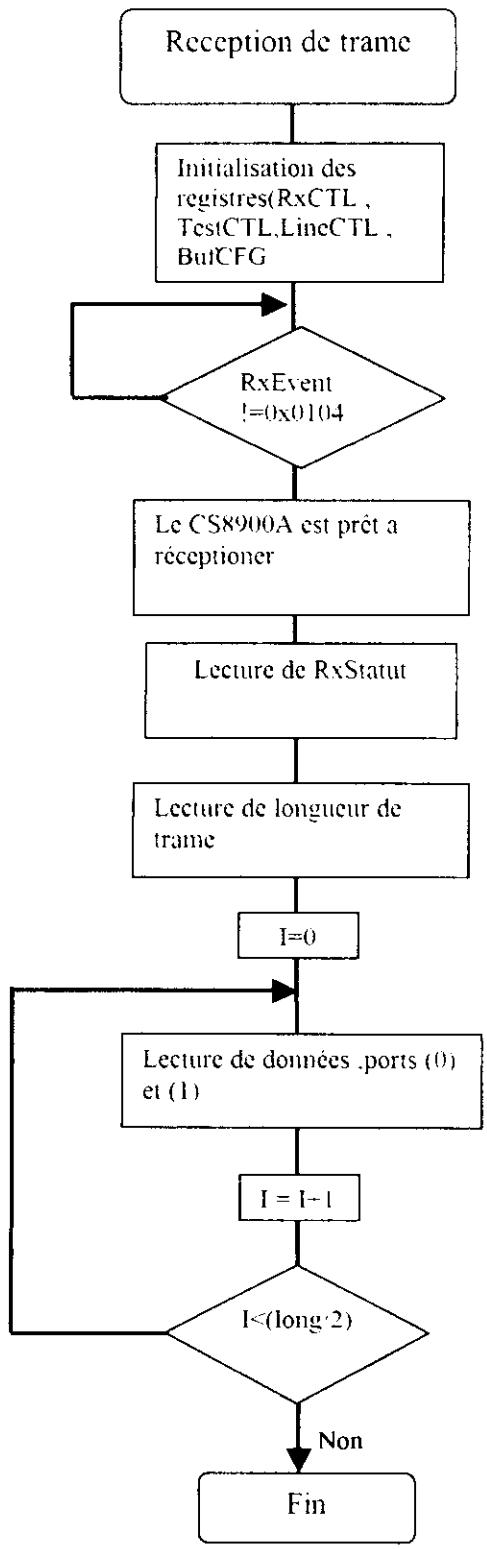
Une fois tout ces registres configurés, on écrit dans le registre TxCMD la valeur 0x00C0, ce qui permet la transmission après que tout le bits soient écrit dans le mémoire tampon du CS8900A, on écrit ensuite la longueur de la trame à transmettre dans le registre TxLENGTH dont l'adresse est 0x8306. On configure le registre BusST pour permettre la transmission, et enfin on effectue une série d'écriture sur les registres 0x8300 puis 0x8301 de la trame à transmettre. Une fois finie, le composant effectue la transmission automatique de trame.



Réception d'une trame

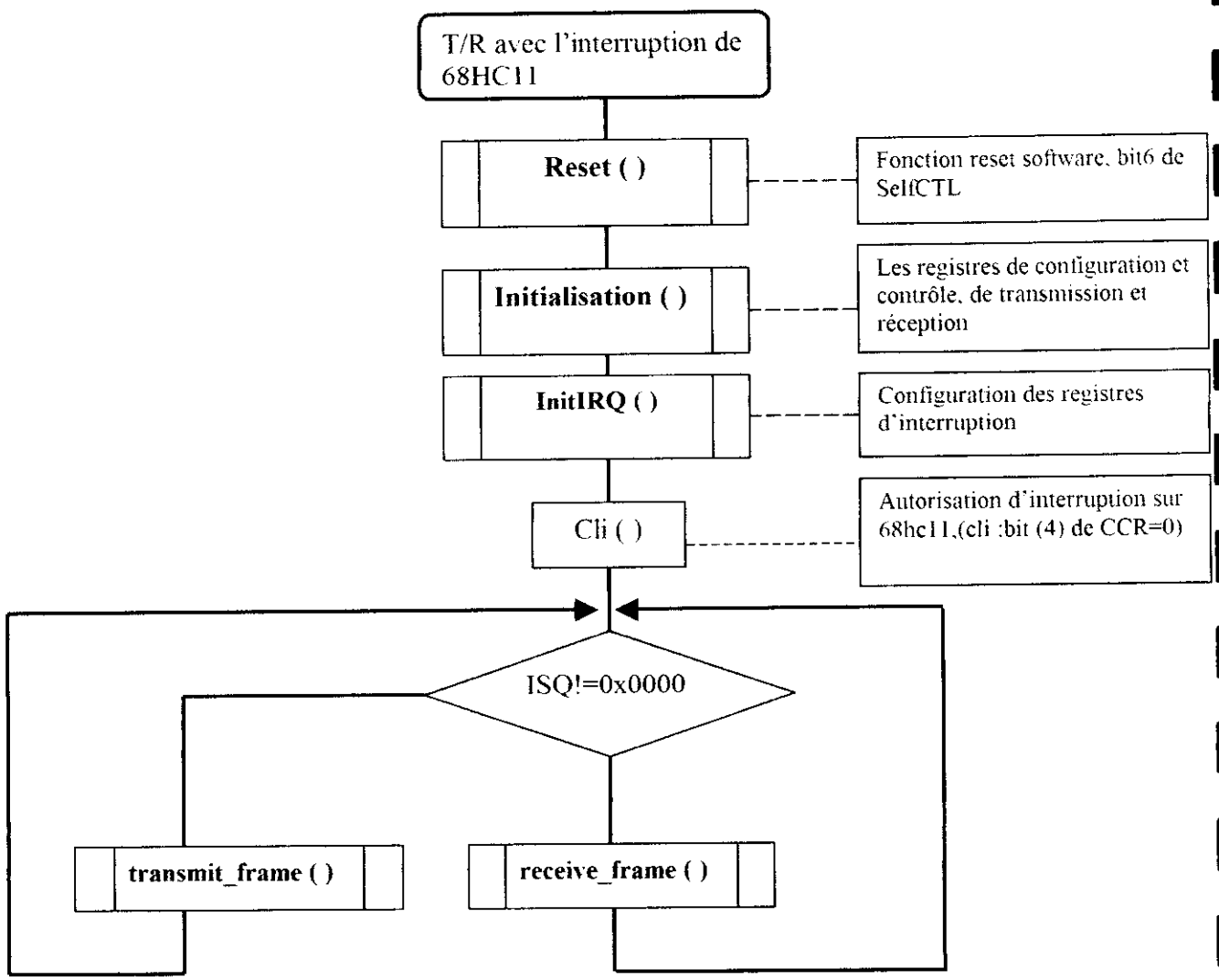
Avant tout fonctionnement du CS8900A, on effectue une initialisation des registres de contrôle de transmission RxCTL, TestCTL, LineCTL et BufCFG.

Une fois tout ces registres configurés, on scrute le registre RxEvent indiquant l'état du CS8900A (attente/réception), puis ensuite on lit le statut/longueur_trame/données par une suite de lecture 8bits des registres d'adresses 0x7300 et 0x7301.



Transmission et réception avec l'interruption de 68hc11

La transmission et réception avec l'interruption de 68hc11 est fonctionne comme suit :
La carte est en mode de transmission. Une fois le CS800A, ayant reçu une trame il envoie un signal d'interruption vers le microcontrôleur pour ainsi exécuter le programme associé au mode de réception.



Remarque

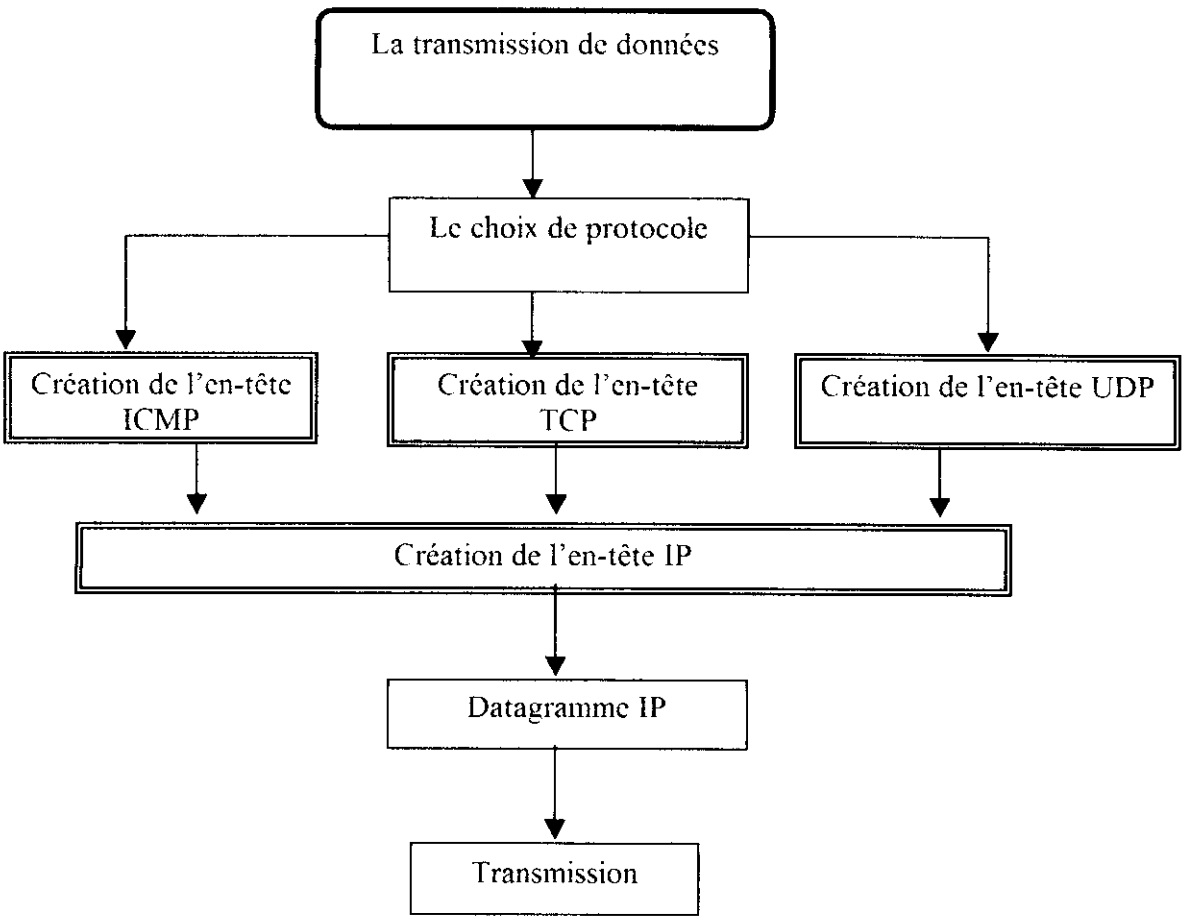
J'ai pris en considération que les registres sont déjà définis.

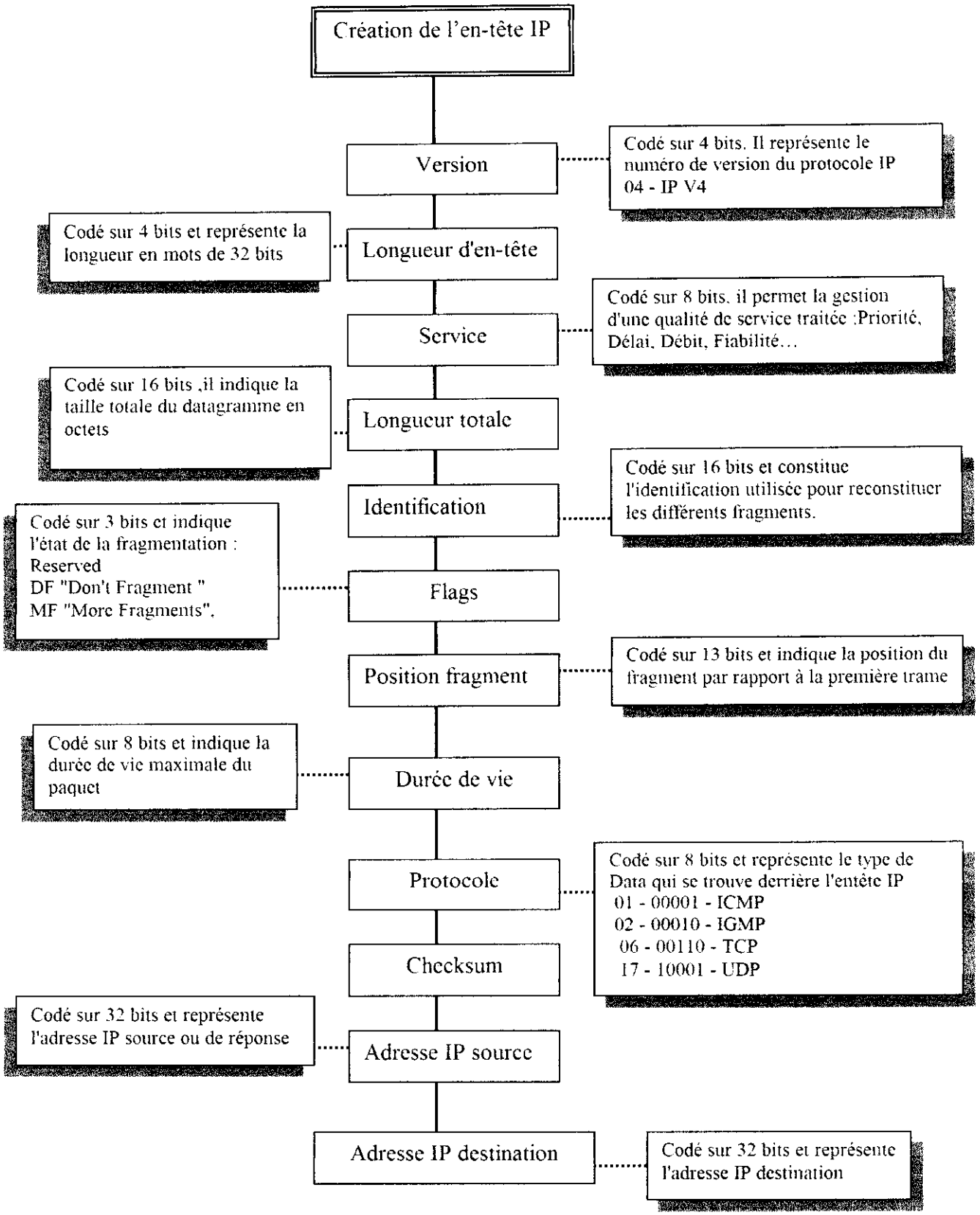
3.2. Au niveau des couches supérieur

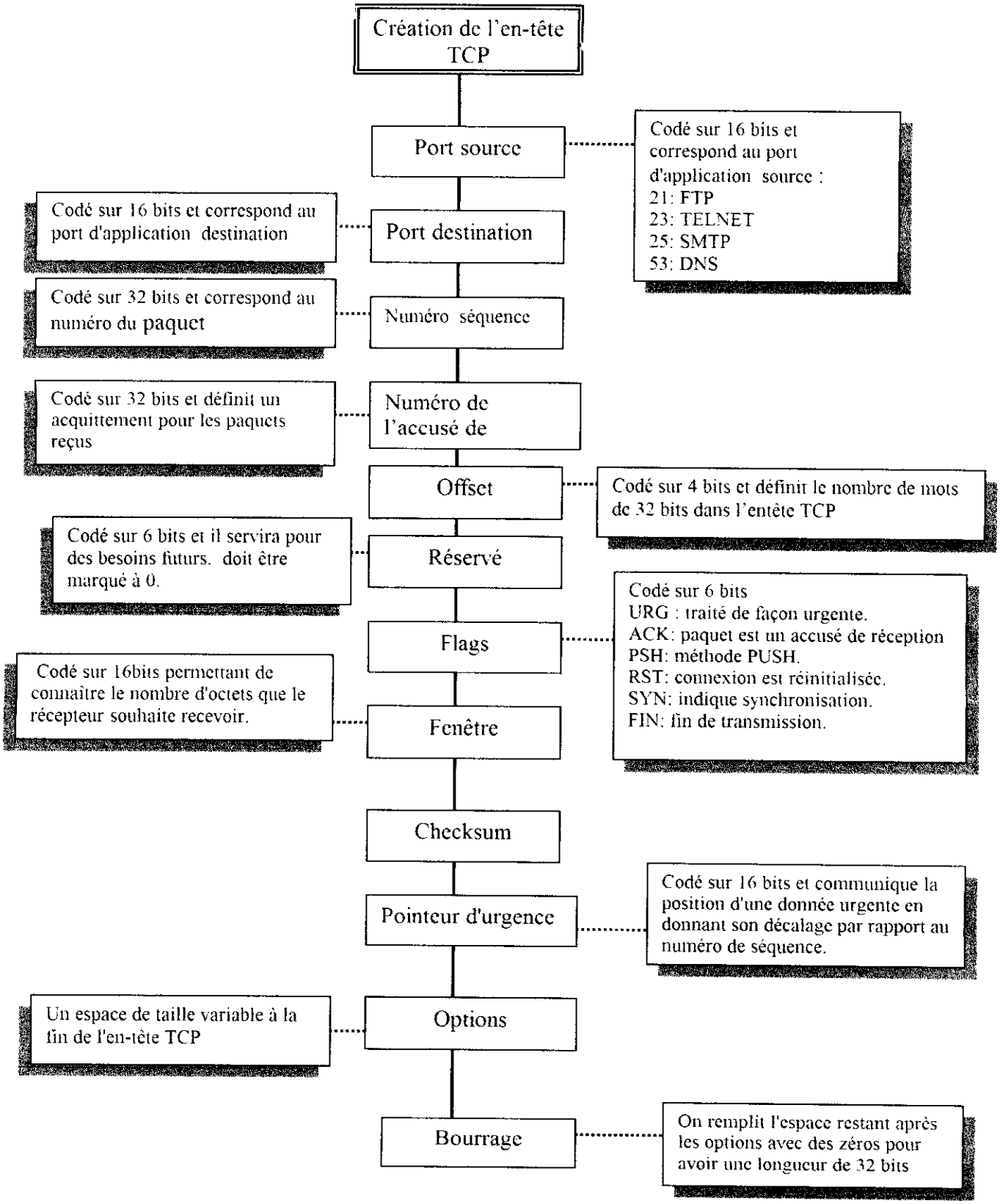
on trouve après la couche laisson,les différentes autres couches, qui intègre plusieurs protocoles(IP,TCP,UDP,ICMP,...).

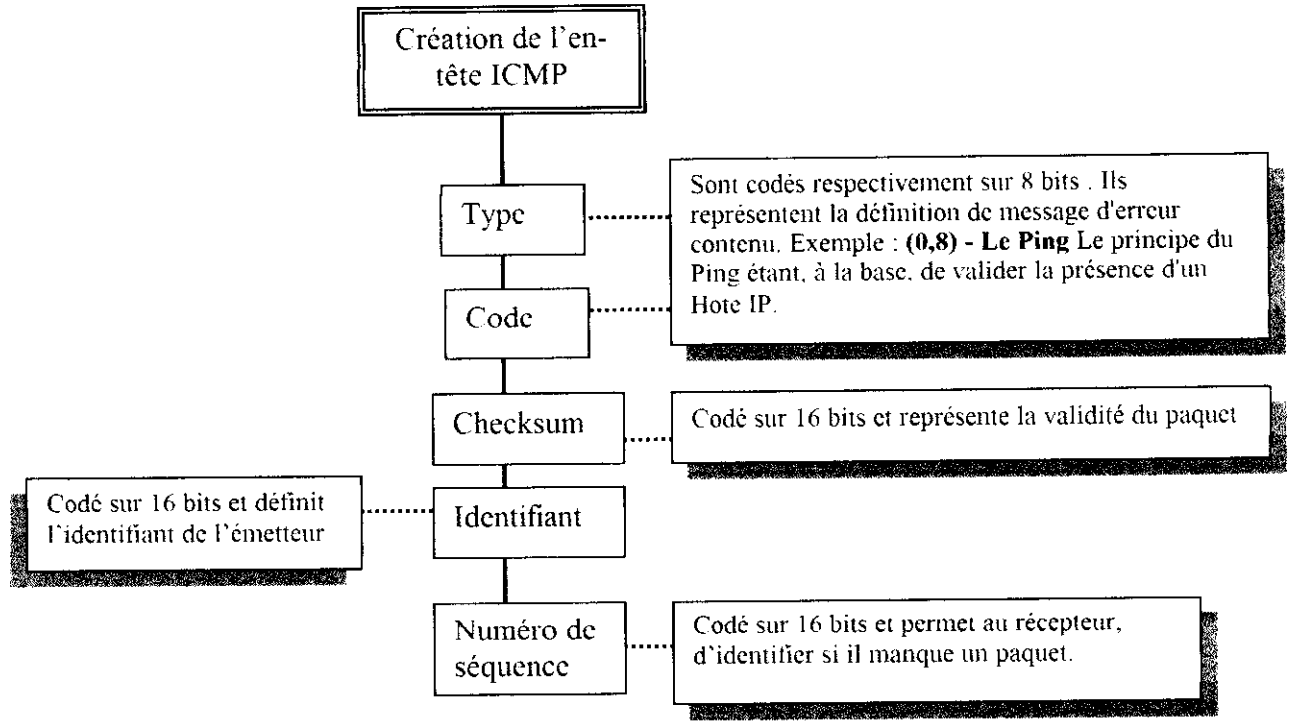
Je donne les organigrammes pour la transmission et la réception des données,et,les organigrammes de chaque en tête spécifier a un protocole.

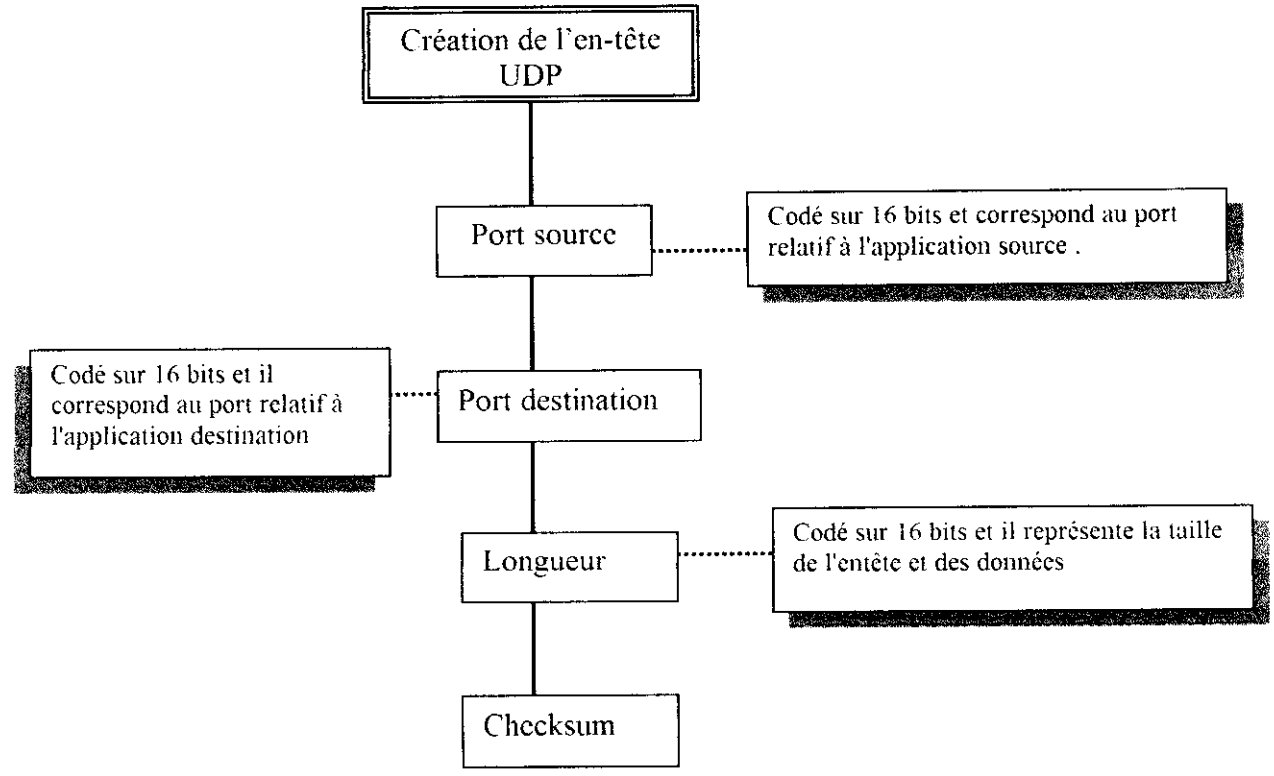
La transmission



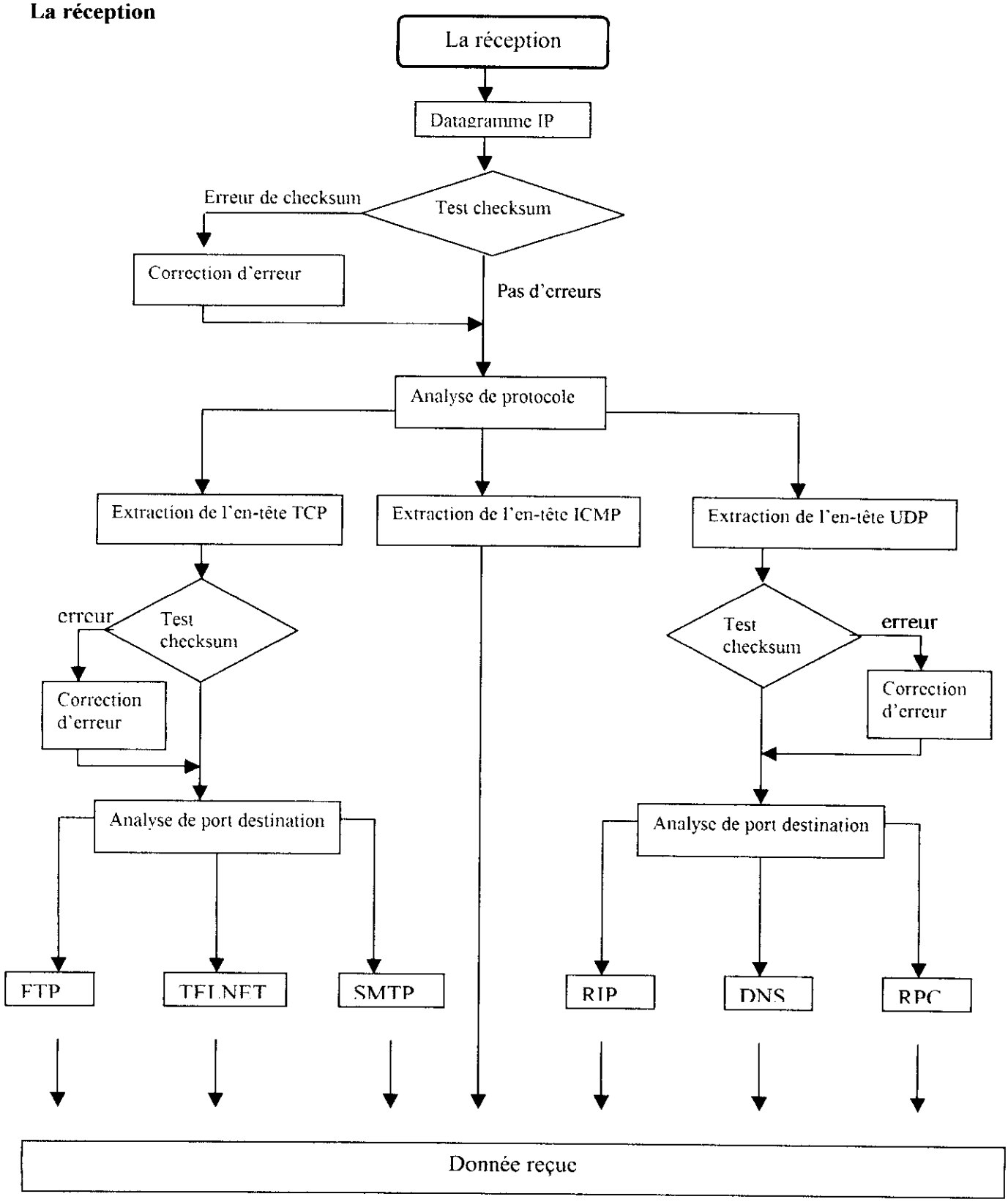

Création de l'en-tête IP
Version
Codé sur 4 bits. Il représente le numéro de version du protocole IP 04 - IP V4
Codé sur 4 bits et représente la longueur en mots de 32 bits
Longueur d'en-tête
Service
Codé sur 8 bits, il permet la gestion d'une qualité de service traitée :Priorité, Délai, Débit, Fiabilité…
Codé sur 16 bits ,il indique la taille totale du datagramme en octets
Longueur totale
Identification
Codé sur 16 bits et constitue l'identification utilisée pour reconstituer les différents fragments.
Codé sur 3 bits et indique l'état de la fragmentation : Reserved DF "Don't Fragment " MF "More Fragments".
Flags
Position fragment
Codé sur 13 bits et indique la position du fragment par rapport à la première trame
Codé sur 8 bits et indique la durée de vie maximale du paquet
Durée de vie
Protocole
Codé sur 8 bits et représente le type de Data qui se trouve derrière l'entête IP 01 - 00001 - ICMP 02 - 00010 - IGMP 06 - 00110 - TCP 17 - 10001 - UDP
Checksum
Codé sur 32 bits et représente l'adresse IP source ou de réponse
Adresse IP source
Adresse IP destination
Codé sur 32 bits et représente l'adresse IP destination


Création de l'en-tête TCP
Port source
Codé sur 16 bits et correspond au port d'application source :
21: FTP
23: TELNET
25: SMTP
53: DNS
Codé sur 16 bits et correspond au port d'application destination
Port destination
Codé sur 32 bits et correspond au numéro du paquet
Numéro séquence
Codé sur 32 bits et définit un acquittement pour les paquets reçus
Numéro de l'accusé de
Offset
Codé sur 4 bits et définit le nombre de mots de 32 bits dans l'entête TCP
Codé sur 6 bits et il servira pour des besoins futurs. doit être marqué à 0.
Réservé
Codé sur 6 bits
URG : traité de façon urgente.
ACK: paquet est un accusé de réception
PSH: méthode PUSH.
RST: connexion est réinitialisée.
SYN: indique synchronisation.
FIN: fin de transmission.
Flags
Codé sur 16bits permettant de connaître le nombre d'octets que le récepteur souhaite recevoir.
Fenêtre
Checksum
Pointeur d'urgence
Codé sur 16 bits et communique la position d'une donnée urgente en donnant son décalage par rapport au numéro de séquence.
Un espace de taille variable à la fin de l'en-tête TCP
Options
Bourrage
On remplit l'espace restant après les options avec des zéros pour avoir une longueur de 32 bits


Création de l'en-tête ICMP
Type
Code
Sont codés respectivement sur 8 bits . Ils représentent la définition de message d'erreur contenu. Exemple : **(0,8) - Le Ping** Le principe du Ping étant, à la base, de valider la présence d'un Hote IP.
Checksum
Codé sur 16 bits et représente la validité du paquet
Codé sur 16 bits et définit l'identifiant de l'émetteur
Identifiant
Numéro de séquence
Codé sur 16 bits et permet au récepteur, d'identifier si il manque un paquet.


Création de l'en-tête UDP
Port source
Codé sur 16 bits et correspond au port relatif à l'application source .
Codé sur 16 bits et il correspond au port relatif à l'application destination
Port destination
Longueur
Codé sur 16 bits et il représente la taille de l'entête et des données
Checksum

La réception



Développement

D'abord on commence par la définition des registres pour la programmation

```
/*****************************************/
/* Définition des Registres du CS8900A   */
/*****************************************/

/* Controle et Configuration */

#define RxCFG    0x0102      /*détermine comment les trames seront transférées et les types
                               de trame causeront des interruptions*/
#define RxCTL    0x0104      /* il définit le type de trame à accepter,et la configuration d'adresse
                               destination*/
#define TxCFG    0x0106      /* chaque bit dans ce registre est une interruption*
#define TxCMD    0x0108      /* comment la trame devrait être transmis*/
#define BufCFG   0x010a      /* chaque bit dans BufCFG est un interruption*/
#define LineCTL  0x0112      /*détermine la configuration de MAC et l'interface physique*/
#define SelfCTL  0x0114      /* commande l'opération des sorties de LED et les modes de
                               bas-puissance*/
#define BusCTL   0x0116      /*commande l'opération de bus ISA */
#define TestCTL  0x0118      /*commande les modes de test de CS8900A*/

/* Status et Evenements */

#define ISQ          0x0120  /*fournir l'information d'interruption*/
#define RxEvent      0x0124  /*statut de courante reception*/
#define TxEvent      0x0128  /* donne le statut d'événement du dernier paquet transmis*/
#define BufEvent     0x012c  /*donne le statut de tampon de transmission et réception*/
#define RxMISS       0x0130  /*compteur indique le nombre des paquet perdue a cause de
                               manque d'espace dans la mémoire*/
#define TxCOL        0x0132  /*compteur inerémente au commencement de transmission, si il
                               y a collision */
#define LineST       0x0134  /* rapporte le statut de l'interface physique Ethernet*/
#define SelfST       0x0136  /* rapporte le statut de l'interface EEPROM*/
#define BusST        0x0138  /* décrit le statut du courante transmission*/
#define TDR          0x013c  /*contient le nombre de période de 10Mhz depuis la première
                               transmission sur AUI, et le premier collision ou badCRC*/
```

Après on passe à la définition des ports de mode 8 bits

```
/*************************************/
/* Definition des ports I/O du CS8900A */
/*************************************/
#define IO_BASE         0x8300              /*adresse de base de l'espace I/O */
#define PP_RxTx         IO_BASE + 0x0000    /*port de Transmission et de
                                              Réception de données (port(0)16 bits*/
#define PP_TxCMD        IO_BASE + 0x0004    /*Commande de transfert*/
#define PP_TxLength     IO_BASE + 0x0006    /*taille de transmission*/
#define PP_ISQ          IO_BASE + 0x0008    /*interruption ISQ*/
#define PP_POINTER      IO_BASE + 0x000a    /*pointeur sur une adresse*/
#define PP_DATA         IO_BASE + 0x000c    /*pointeur de données pour le transfert*/
```

Il faut spécifier un emplacement permis à stocker les données

```
/* Zone Mémoire pour récupérer/envoyer */
#define RxBuffer   0x7000   /*tampon de réception*/
#define TxBuffer   0x7500   /*tampon d'émission*/
```

Lecture et écriture des données de 16 bits

```
/* fonction d'écriture dans une adresse(out16, out16inv) */

void out16(int adresse,int donnee)  /*fonction d'ecriture 16 bits d'une donnée dans une  adresse*/
{
char  donnee_l,donnee_h;            /* déclaration des donnees à  poids faible, fort */
char *ptrl = (char *)adresse;       /* pointeur sur poids faible */
char *ptrh = (char *)(adresse+1);   /* pointeur sur poids fort*/
donnee_l = donnee & 0xff;           /* Récupération de  PFb */
donnee_h = (donnee >> 8) & 0xff;    /* Récuperation de  PFt */
*ptrl = donnee_l;                   /*Octet faible --> 8 faible */
*ptrh = donnee_h;                   /*Octet fort --> 8 fort */
}
```

La même que précédente mais les bits forts(donnee_h) dans adresse faible(ptr_l)

```
void out16inv(int adresse,int donnee)
{
char donnee_l,donnee_h;
char *ptrl = (char *)adresse;              /* pointeur sur poids faible */
char *ptrh = (char *)(adresse+1);          /* pointeur sur poids fort */
donnee_l = donnee & 0xff;                  /* Recup PFb */
donnee_h = (donnee >> 8) & 0xff;           /* Recup PFt */
*ptrl = donnee_h;                          /*Octet fort --> S faible */
*ptrh = donnee_l;                          /*Octet faible --> S fort */
}
```

```
/* fonction lecture depuis une adresse */

int in16L(int adresse)
{char donnee_l,donnee_h;
char *ptrl = (char *)adresse;          /* pointeur sur poids faible */
char *ptrh = (char *)(adresse+1);      /* pointeur sur poids fort */
int donnee;
donnee_l = *ptrl;                      /* Recup PFb de S faible */
donnee_h = *ptrh;                      /* Recup PFt de S fort */
donnee = ( ( donnee_h << 8 ) + donnee_l ) & 0xffff;
return(donnee);                        /*renvoi donnée*/
}
```

```
int in16H(int adresse)
{
char donnee_l,donnee_h;
char *ptrl = (char *)adresse;
char *ptrh = (char *)(adresse+1);
int donnee;
donnee_h = *ptrh;          /* Recup PFt de S fort */
donnee_l = *ptrl;          /* Recup PFb de S faible*/
donnee = ( ( donnee_h << 8 ) + donnee_l ) & 0xffff;
return(donnee);
}
```

```
/*fonction d'écriture dans un registre*/

void outReg16(int Registre, int Donnee)
{
out16( PP_POINTER, Registre);/*écriture de l'adresse*/
out16( PP_DATA , Donnee );/*écriture de donnée*/
}
```

```
/*fonction de lecture depuis un registre*/

int inReg16(int Registre)
{
int Donnee;
out16( PP_POINTER, Registre); );/*écriture de l'adresse*/
Donnee = in16(PP_DATA);/* récupération de donnée*/
return(Donnee);
}
```

La fonction d'initialisation pour les registres

```
void initialisation(void)
{
/* Configuration de RxCTL (CRC, toutes S) */
outReg16( RxCTL, 0x0180);
/* Configuration du registre TestCTL : (pas de link inputs(DisablT), half duplex(FDX),avec Bacoff,pas
de LoopBack) */
outReg16(TestCTL, 0x0099);
/* Configuration LineCTL (10BaseT,SerTxOn,SerRxOn) */
outReg16( LineCTL, 0x00D3);
/* Configuration de BufCFG */
outReg16( BufCFG, 0x000B);
}
```

Fonction reset

```
void reset(void)
{
/* Bit 6 a"1 "du registre SelfCTL  */
outReg16(SelfCTL, 0x0040);
}
```

Fonction d'interruption

```
void initIRQ(void)
{
/* Configuration BufCFG: interruption a chaque passage d'une trame(RxDestiE bit F) 800B*/
 outReg16( BufCFG, 0x800B);
/* Configuration RxCFG: (RxOiE. trame reçue avec erreur) */
outReg16( RxCFG, 0x0103);
/* Selection de l'interruption IRQ0 ,0022 l'adresse de numéro d'adresse"page 42 datashet"*/
outReg16(0x0022,0x0000);
/* Configuration de BusCTL: (EnableIRQ) */
outReg16( BusCTL, 0x8017);
}
```

Programme de transmission d'une trame

```
/*transmission of frame*/
int transmit_frame(int Length, int *data)
{
int j,var;
SIZE_OF_FRAME_TX = Length;/*la tail de donnée*/
/*  Ecriture de la commande TxCMD */
out16(PP_TxCMD , 0x00c9 );/*transmission apres la trame complet*/
/*  Ecriture de la longueur de la trame en OCTECT */
out16(PP_TxLength , SIZE_OF_FRAME_TX);
/*  Lecture de BusST */
var = inReg16(BusST); /*le statut de courante opération*/
/*  Test du Bit Rdy4TxNow: On devrait lire BusST=0118h ,bit8=1(prêt a transmettre)*/
if ( var!= 0x0118)
{
return(FAILURE);/*le CS8900A n'est pas prêt*/
}
/*  Copie de la trame entiere de TxBuffer dans le chip */
/*  Ecriture de  données */
for (j = 0; j < (SIZE_OF_FRAME_TX/2); j++)
{
out16inv( PP_RxTx, *(data++) );
}
return(SUCCESS);
}
```

Programme de reception

```
int receive_frame(int *data)
{
int i,var;
 *Lecture de RxEvent, le statut de dernière trame reçue */
var = in16H(PP_RxTx);
/*Lecture de Length of Frame */
SIZE_OF_FRAME_RX = in16h(PP_RxTx);
/* Ecriture de la Length dans une zone memoire */
out16(0x4100,SIZE_OF_FRAME_RX);
if ( var!= 0x0104)
{
return(FAILURE); *la trame et avec erreur de CRC et le length  non valide*/
}

/*Recuperation de la trame entiere en RxBuffer */
for (i=0; i<(SIZE_OF_FRAME_RX/2); i++)
{
var = in16L(PP_RxTx);
out16( (int)&data[i],var);        /* sauvegarde en zone data */
}
return(SUCCESS);
}
```

Finalement le programme principale

```
#define cli()   asm("cli") *autorisation d'interruption*/
#define SUCCESS       0
#define FAILURE       1
 void main()
{
/* Variables Globales */
int SIZE_OF_FRAME_TX;
int SIZE_OF_FRAME_RX;
reset ( ) ;
initialisation ( ) ;
initIRQ ( ) ;
cli ( ) ;
int  etarecep, i , etatran;
for ( ; ; )
{
/*reception tanque il y a interruption*/
while(i = inReg16L(ISQ)!=0x0000)
  etarecep = receive_frame(*RxBuffer);
 *transmission, pas d'interruption*/
etatran = transmit_frame(64, *TxBuffer);
}
}
```

```
/*****************************************
** programme de transmission des données **
*****************************************/
#include <stdio.h>
#include <string.h>

/* l'en-tête IP*/
typedef struct iphdr
{
   unsigned char  verlen;
   unsigned char  ihl;
   unsigned char  tos;
   unsigned short tot  len;
   unsigned short id;
   unsigned char  flags;
   unsigned short position;
   unsigned char  ttl;
   unsigned char  protocol;
   unsigned short checksum;
   unsigned int   saddr;
   unsigned int   daddr;
} IP  HDR;

/* l'en tête ICMP*/
typedef struct icmphdr
{
        unsigned char  type;
        unsigned char  code;
        unsigned short checksum;
        unsigned short id;
        unsigned short seq;
}ICMP  HDR ;

/* l'en tête UDP*/
typedef struct udphdr
{
   unsigned short srcport;
   unsigned short dstport;
   unsigned short length;
   unsigned short checksum;
} UDP  HDR;

/*l'en tête TCP*/
typedef struct tcphdr
{
   unsigned short sport;
   unsigned short dport;
   unsigned int seqnum;
   unsigned int acknum;
   unsigned char offset;
```

```
    unsigned char reserved;
    unsigned char flags;
    unsigned short window;
    unsigned short checksum;
    unsigned short pointer;
} TCP_HDR;

/* calcul de checksum*/
unsigned short checksum(unsigned short *addr, int len)
{
                register int nleft = len;
                register unsigned short *w    addr;
                register unsigned short answer;
                register int sum = 0;

                while (nleft > 1) {
                        sum += *w++;
                         nleft -= 2;
                }
                if (nleft == 1){
                        unsigned short u    0;
                        *(unsigned char *)(&u) = *(unsigned char *)w;
                        sum += u;
                }
                sum = (sum >> 16) + (sum & 0xffff);
                sum += (sum >> 16);
                 answer = ~sum;
                return (answer);
}
 void main()
 {
        char message[255];
        unsigned char r;
        unsigned int i,k,j;
        unsigned short p,n,v;
        printf("donner le message a transmettre\n");
    scanf("%s",&message);
    j=strlen(message);
char entete[50];
iphdr *ip=(iphdr*)entete;
icmphdr *icmp=(icmphdr*)(entete + sizeof(struct iphdr));
udphdr *udp=(udphdr*)(entete+sizeof(struct iphdr));
tcphdr *tcp=(tcphdr*)(entete + sizeof(struct iphdr));
printf("**********************************\n");
printf("calcule de l'en tete IP\n");
printf("-----------------------------------------\n");
   printf("donner la version\n");
        scanf("%i",&k);
   ip->verlen= k;
        printf("donner la IHL\n");
```

```
        scanf("%i",&k);
        ip->ihl=k;
        printf("donner le tos\n");
        scanf("%i",&k);
        ip->tos=k;
        printf("donner l'identification\n");
        scanf("%i",&k);
        ip->id=k;
        printf("donner le flags\n");
        scanf("%i",&k);
        ip->flags=k;
        printf("donner position \n");
        scanf("%i",&k);
        ip->position=k;
        printf("donner le TTL\n");
        scanf("%i",&k);
        ip->ttl=k;
        printf("donner le protocole\n");
        scanf("%i",&k);
        ip->protocol=k;
        printf("donner l'adresse source\n");
        scanf("%i",&k);
        ip->saddr=k; //remplacez par votre ip...
        printf("donner l'adresse distination\n");
        scanf("%i",&k);
        ip->daddr=k; //remplacez par le pc à pinger, google par ex...
        ip->checksum =0;
        ip->checksum=checksum((unsigned short *)ip, sizeof(struct iphdr));

/* couche transport*/
        if(ip->protocol==1)
        { printf("*************************************\n");
                printf(" faire remplir le champ ICMP\n");
                printf("---------------------------------------------\n");
                printf("donner l'type\n");
           scanf("%i",&k);
           icmp->type=k;
                printf("donner le code\n");
        scanf("%i",&k);
           icmp->code=k;
                printf("donner l'identification\n");
        scanf("%i",&k);
                icmp->id=k;
                printf("donner le numero de sequence\n");
        scanf("%i",&k);
                icmp->seq=k;
           icmp->checksum=0;
           icmp->checksum=checksum((unsigned short *)icmp, sizeof(struct icmphdr));
           ip->tot_len=(sizeof(struct iphdr)+sizeof(struct icmphdr));
                printf("*************************************\n");
```

```
                printf(" l'en tete ICMP est:\n");
                printf("\n");
                printf("%X%X%X%X%X%X\n",icmp->type,icmp->code,icmp->checksum,icmp-
>id,icmp->seq);

        }
                else if(ip->protocol==17)
        {printf("*************************************\n");
        printf("remplir le champ UDP\n");
        printf("---------------------------------------------\n");
        printf("donner le port source\n");
        scanf("%i",&k);
        udp->srcport=k;
        printf("donner le port distination\n");
        scanf("%i",&k);
    udp->dstport=k;
        printf("donner le length\n");
        scanf("%i",&k);
    udp->length=k;
    udp->checksum=0;
    udp->checksum=checksum((unsigned short *)udp, sizeof(struct udphdr));
        ip->tot_len=(sizeof(struct iphdr)+sizeof(struct udphdr));
      printf("*************************************\n");
                printf(" l'en tete UDP est:\n");
                printf("\n");
                printf("%X%X%X%X\n",udp->srcport,udp->dstport,udp->length,udp-
>checksum);

        }
                else if(ip->protocol==6)
                {printf("*************************************\n");
                 printf("faire remplire le champ TCP\n");
      printf("---------------------------------------------\n");
                 printf("donner le port source\n");
            scanf("%i",&k);
            tcp->sport=k;
                 printf("donner le port destination\n");
            scanf("%i",&k);
      tcp->dport=k;
      printf("donner le numero de sequence\n");
            scanf("%i",&k);
                 tcp->seqnum=k;
                 printf("donner l'acquittement\n");
            scanf("%i",&k);
      tcp->acknum=k;
                 printf("donner le offset \n");
            scanf("%i",&k);
      tcp->offset=k;
      tcp->reserved=0;
                 printf("donner le flags\n");
```

```
        scanf("%i",&k);
    tcp-> flags=k;
            printf("donner la fenetre\n");
        scanf("%i",&k);
    tcp-> window=k;
    tcp-> checksum=0;
            printf("donner le pointer\n");
        scanf("%i",&k);
    tcp-> pointer=k;
    tcp->checksum=checksum((unsigned short *)tcp, sizeof(struct tcphdr));
        ip->tot_len=(sizeof(struct iphdr)+sizeof(struct tcphdr));
   printf("************************************\n");
            printf(" l'en tete TCP est:\n");
            printf("\n");
   p=tcp->offset;
            p=(p<<12);
            n=tcp->reserved;
            n=(n<<10);p=p+n+(tcp->flags & 0x3F);
    printf("%X%X%X%X%X%X%X%X%X\n",tcp->sport,tcp->dport,tcp->seqnum,tcp-
>acknum,p,tcp->window,tcp->checksum,tcp->pointer);

            }
            /*le message en hexadecimale*/
            printf("************************************\n");
        printf("  le message en hexa est: \n        \n");
   printf("************************************\n");
            for(i=0;i<j;i++)
   printf("%X",message[i]);
            printf("\n");
   printf("************************************\n");
        printf("  l'en tete IP est: \n        \n");
            printf("************************************\n");
            r=ip->verlen;
            r=(r<<4)+(ip->ihl & 0xf);
            v=ip->flags;
            v=(v<<12);
            v=v+(ip->position & 0xfff);
printf("%X%X%X%X%X%X%X%X%X%X%X\n",r,ip->tos,ip->tot_len,ip->id,v,ip->ttl,ip-
>protocol,ip->checksum,ip->saddr,ip->daddr);
 printf("-----------------------------------------\n");
printf("************************************\n");
printf(" le paquet finale a transmettre est\n");
printf("\n");
printf("%X%X%X%X%X%X%X%X%X%X",r,ip->tos,ip->tot_len,ip->id,v,ip->ttl,ip-
>protocol,ip->checksum,ip->saddr,ip->daddr);
if(ip->protocol==1)
printf("%X%X%X%X%X%X",icmp->type,icmp->code,icmp->checksum,icmp->id,icmp->seq);
else if(ip->protocol==17)
printf("%X%X%X%X%X",udp->srcport,udp->dstport,udp->length,udp->checksum);
else if(ip->protocol==6)
```

```
printf("%X%X%X%X%X%X%X%X",tcp->sport,tcp->dport,tcp->seqnum,tcp->acknum,p,tcp-
>window,tcp->checksum,tcp->pointer);
for(i=0;i<j;i++)
printf("%X",message[i]);
printf("\n");
printf("-----------------------------------------------\n");
getchar();
}
```

Conclusion générale

L'échange des données dans un réseau se fait en générale par ordinateur muni d'une carte réseau. Cependant, notre travail a montré que cet échange de données peut se faire plus simplement avec un micro-contrôleur et un circuit Ethernet.

Le micro-contrôleur fait la gestion de la transmission et la réception des données à partir de circuit Ethernet qui fait la construction des trames.

Pour implémenter cette carte dans le contrôle d'accès il nous faut deux applications communicantes supervisant l'accès.

La conception que nous avons abordée à travers notre projet de fin de cycle a abouti à la proposition d'un modèle à base d'un microcontrôleur couplé à une interface ethernet appropriée pour interagir dans un environnement client serveur dédié au contrôle d'accès.

L'avantage apparaît évident en ce sens qu'on peut coupler aisément tout système minimal à un réseau local donné.

En terme de complément au travail que nous avons effectué, on peut dire qu'il serait intéressant :

- de compléter la conception par une réalisation effective pour valider le modèle proposé
- de développer la partie serveur par l'étude et le développement d'un système d'information dédié à l'application cible.

BIBLIOGRAPHIE

1. Alexis Ferréo : «*Ethernet et ses évolutions*», Addison-wesley,1995.

2. Bernard Beghyn : «*Le micro-contrôleur 68HC11*», HERMES,1997.

3. Andrew S.TANENBAUM :«*RESEAUX :Architectures,protocoles,applications* » ,Prentice HALL,1989.

4. D.BRENT CHAPMAN, ELIZABETH.ZWICKY : «*FIREWALLS: La sécurité sur Internet*»,O'Reilly International THOMSON.

5. Datasheets crystal concernant le CS8900A.

6. Datasheet du 68HC11 de MOTOROLA.

WEBOGRAPHIE

-1- http://www.commentcamarche.com
-2- http://www-elec.enserb.fr/~kadionik
-3- http://www.cirrus.com
-4- http://www.motorola.com
-5- http://www.securiteinfo.com/
-6- http://www.tracehabil.com/
-7- http://www.excem.fr/

ANNEXE CS8900A

L'interface Ethernet CS8900

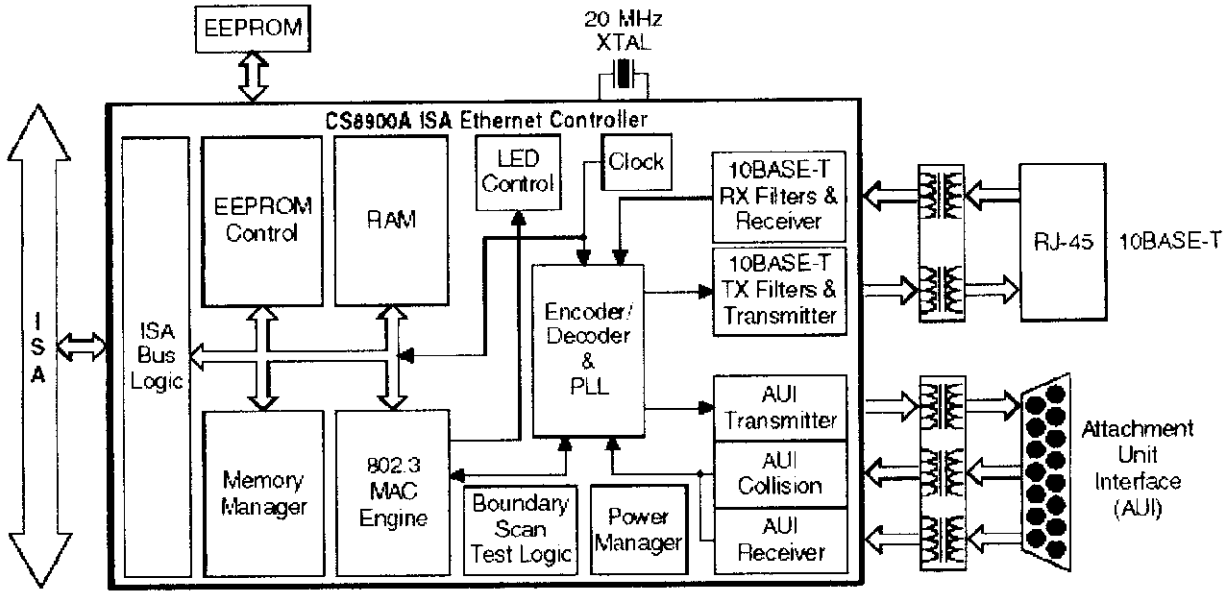
Le CS8900 est une interface Ethernet permettant la réception et l'émission des trames Ethernet sur le réseau. Ce composant est connecté au réseau via un connecteur RJ45. Le bus de données du CS8900 est de 16 bits.
Ce composant effectue une première analyse des trames Ethernet reçues en vérifiant la valeur du CRC, ainsi que la taille de ces mêmes trames.

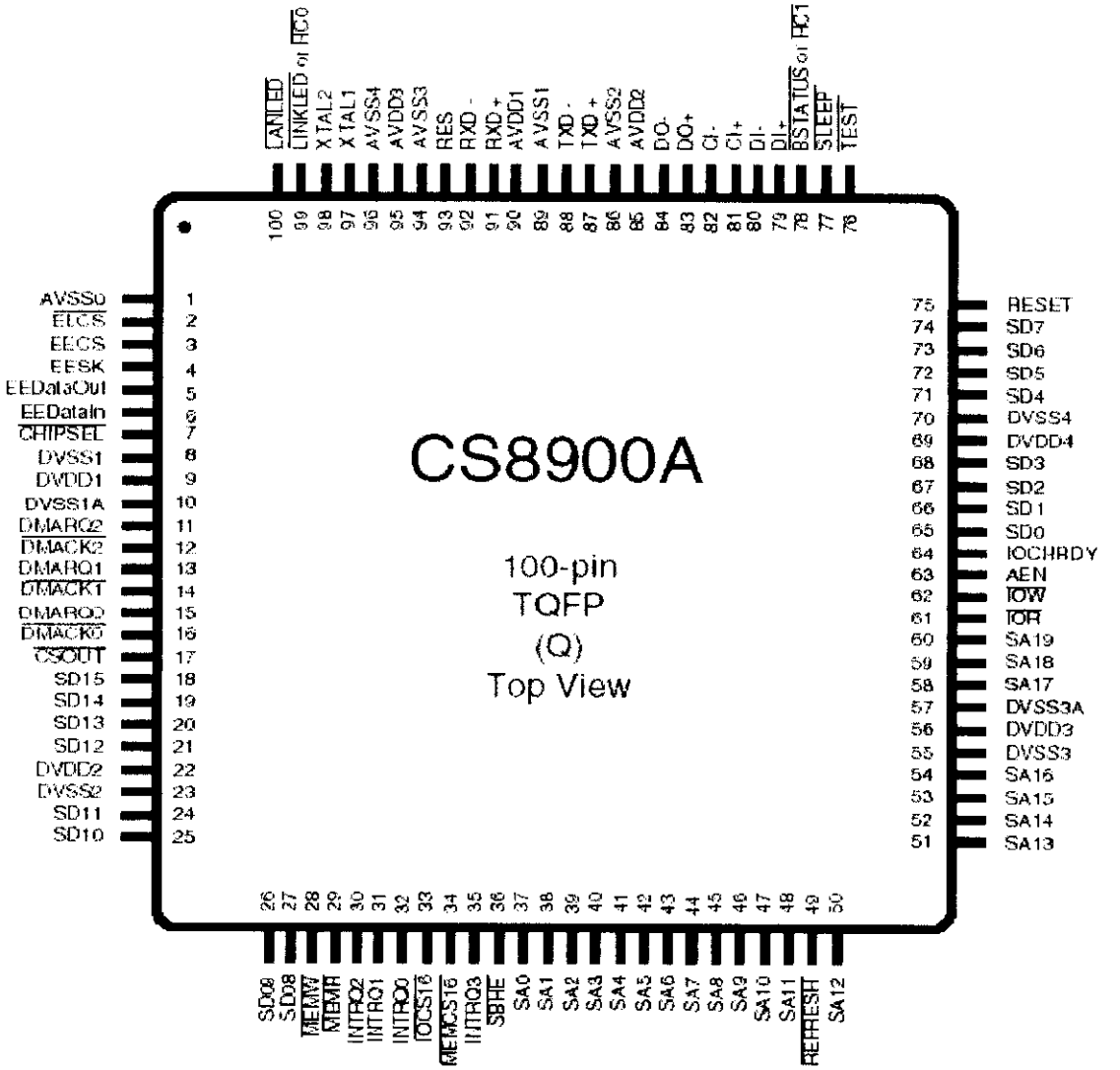
Caractéristiques

On peut trouver dans ce circuit:

1. une interface pour le bus ISA (Industry Standard Architecture).
2. les lignes d'interruption (4) et 3 canaux DMA
3. Contrôleur Ethernet 16 bits.
4. Modes de fonctionnement en 8 bits (I/O)
5. 4Ko de RAM intégré dans le Chip.
6. Unité MAC (Media Access Control), qui gère les transmissions et réceptions de trames, conformément à la norme Ethernet IEEE 802.3.
7. Interface EEPROM pour la configuration personnalisée au démarrage.
8. Interface bout de ligne entièrement analogique avec 10Base-T et AUI (Attachment Unit Interface).
9. Package TQFP permettant le design de cartes occupant une surface inférieure à 10cm carré.
10. Design possible du chip sur carte mère ou carte d'adaptation.

Architecture





Brochage

Description rapide des Broches du CS8900A pour la conception Hardware:

Interface du Bus ISA

SA0...SA19 : Bus d'adresse du système pour décodage des accès aux espaces I/O ou Memory.
SD0...SD15 : Bus bidirectionnel de données format 16 bits.
RESET : Broche d'initialisation asynchrone actif sur état HAUT.
AEN : Entrée Adresse Enable.
Si Test* est HAUT, cette entrée indique au CS8900Aque le contrôleur du système DMA prend le contrôle du bus ISA .
MEMR* : Memory Read. Entrée active sur état BAS indique que l'on exécute une opération de lecture.

MEMW* : Memory Write. Entrée active sur état BAS indique que l'on exécute une opération d'écriture.
MEMCS16* : Sortie pour le mode MEM 16bits
REFRESH* : Entrée pour indiquer qu'un cycle de rafraîchissement DRAM est en progression. Lorsque REFRESH* est BAS, MEMR*, MEMW*, IOR*, IOW*, DMACK1* et DMACK2* sont ignorées.
IOR* : I/O Read. Indique une sortie du contenu du registre I/O 16 bits sur le bus de données du système lorsqu'une adresse valide est détectée.
Lorsque REFRESH* est BAS, IOR* est ignorée.
IOW* : I/O Write. Le CS8900 écrit les données du bus de données vers le registre I/O 16 bits lorsqu'une adresse valide est détectée.
Lorsque REFRESH* est BAS, IOW* est ignorée.
IOCS16* : Le CS8900A génère cette sortie BAS lorsqu'il reconnaît une adresse du bus ISA qui correspond à l'espace I/O assigné sinon sortie en 3e état.
SBHE* : Entrée indiquant un transfert sur l'octet fort du bus de données (SD8...SD15)
INTRQ0...INTRQ2 : Ces sorties indiquent la présence d'un événement d'interruption.
DMARQ0...DMARQ2 : Sortie HAUT indiquant que le CS8900A demande un transfert DMA. Seule une sortie est utilisée à la fois, les autres sont en 3 e état.
DMACK0...DMACK2 : Entrée indiquant la reconnaissance par l'hôte de la correspondance de la sortie de demande DMA
CHIPSEL* : Entrée générée par un décodeur logique d'adresse latchable externe lorsque une adresse mémoire valide est présente sur le bus ISA.
Si le Mode Memory n'est pas utilisé, CHIPSEL* doit être tirée BAS.
CHIPSEL* est ignorée en mode I/O et DMA du CS8900A.

Interface EEPROM et PROM de Boot

EESK : Horloge Série utilisée pour les transferts de données vers l'EEPROM.
EECS : Sortie pour sélectionner l'EEPROM
EEDataIN : Entrée Série pour la réception des données de l'EEPROM connectée à la broche DO de l'EEPROM.
Permet également de sonder la présence de l'EEPROM.
ELCS* : Signal bidirectionnel pour configurer un décodeur logique d'adresse externe latchable.
Si le LA n'est pas utilisé, ELCS* doit être tiré vers BAS.
EEDataOut : Sortie série pour l'envoie de données vers l'EEPROM, connectée à la broche DI de l'EEPROM.
Si TEST* est BAS, elle devient la sortie pour le Boundary Scan Test.
CSOUT* : Sortie pour sélectionner une PROM de Boot externe lorsque le CS8900A décode une adresse mémoire valide de la PROM de Boot.

Interface 10 Base-T

TXD+, TXD- : Paire différentielle de sortie 10Mb/s pour la transmission de données codées en Manchester vers le
10Base-T.
RXD+, RXD- : Paire différentielle d'entrée 10Mb/s pour la réception de données codées en Manchester vers le 10Base-T.

InterfaceAUI (Attachment Unit Interface)

DO+, DO- : Paire différentielle de sortie 10Mb/s pour la transmission de données codées en Manchester vers l'AUI.
DI+, DI- : Paire différentielle d'entrée 10Mb/s pour la réception de données codées en Manchester vers l'AUI.
CI+, CI- : Paire différentielle d'entrée connectée à la paire de collision de l'AUI.

Pins Généraux

XTAL1...XTAL2 : Pour connexion d'un quartz de fréquence 20MHz.
Si un signal de 20MHz est utilisé au lieu du quartz, il est relié à XTAL1 et XTAL2 libre.
SLEEP* : Entrée permettant l'activation des modes de mise en veille hardware (Suspend et Hardware Standby).
LINKLED* ou HC0* : Sortie pour la détection d'impulsions de connexion valide.
BSTATUS* ou HC1* : Sortie indiquant une activité sur le bus ISA.
LANLED* : Sortie indiquant l'arrivée, la transmission de paquets pour une collision.
TEST* : Entrée utilisée pour mettre le CS8900A en mode Boundary Scan Test.
Pour une utilisation normale, cette pin est laissée HAUT.
RES : Cette entrée est reliée à une résistance de 4.99K +/- 1% pour
DVDD1...DVDD4 : Fournit le 5V +/- 5% aux circuits numériques du CS8900A.
DVSS1...DVSS4 : Masses numériques.
AVDD1...AVDD3 : Fournit le 5V +/- 5% aux circuits analogiques du CS8900A.
AVSS0...AVSS4 : Masses analogiques.

Le fonctionnement du chip CS8900A en mode 8 bits

En raison de sa petite taille, le CS8900A est un bon candidat pour des conceptions avec un bus de données de 8 bits.
Avec le mode 8 bis, les accès sur le cs8900a s'effectuent à partir des 8 port E/S de 16 bits.

Offset	Type	Description
0000h	Lecture / Ecriture	Rx/Tx de la Donnée (Port 0)
0002h	Lecture / Ecriture	Rx/Tx de la Donnée (Port 1)
0004h	Ecriture uniquement	TxCMD (Commande de Transfert)
0006h	Ecriture uniquement	TxLength (Taille de Transmission)
0008h	Lecture uniquement	ISQ Interrupt Status Queue
000Ah	Lecture / Ecriture	PacketPage Pointer
000Ch	Lecture / Ecriture	Donnée PacketPage (Port 0)
000Eh	Lecture / Ecriture	Donnée PacketPage (Port 1)

1. Rx / Tx de la donnée

Ce port est utilisé pour la réception et la transmission des données de et vers le CS8900A
- Port (0) : utilisé pour les opérations de 16 bits

- Port (1) et Port (0): les deux utilisés pour les opérations de 32 bits (le mot faible dans le port (0))

2. TxCMD
   Le hôte écrit la commande dans ce port à chaque commencement de transmission. Cette commande indique au CS8900A comment va être la transmission.

3. TxLength
   On écrit ici la longueur de trame a transmis.

4. ISQ, statut d'interruption:
Ce port contient la valeur courante du registre ISQ (Interrrupt Status Queue)

5. PacketPage Pointer
Les 12 premiers bits indiquent l'adresse interne du registre à accéder pendant l'opération courante, les 3 bits suivants(C, D, E) sont a lecture seule, le dernier bit F indique a ce que l'incrémentation d'adresse est automatique ou non.

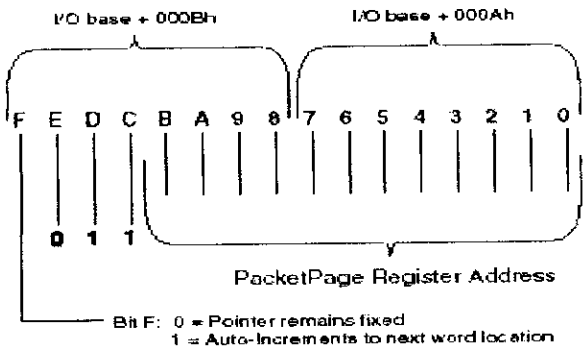


Figure 18. PacketPage Pointer

6. data PacketPage
Un port pour le transfert de données.

Fonctions non soutenues en mode de 8 bits

a. Pas de mode d'interruption en 8 bits, donc un Polling s'impose.
b. Pas d'EEPROM dans ce mode.