REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
– MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE –
ECOLE NATIONALE POLYTECHNIQUE.

DEPARTEMENT D'ELECTRONIQUE

Projet de fin d'études

En vue de l'obtention du
Diplôme d'ingénieur d'état en Electronique

Thème:

Contribution à la réalisation d'un serveur DHCP sur SOC

Proposé et dirigé par :

Mr. R.Sadoune

Réalisé par:

Chabane Narymane
Sarni Mounir

Promotion : Juin 2005

Je dédie ce travail

A mes chers parents, pour leur sacrifices, et leur dévouement,

A ma sœur Kahina et à mon frère Adem,

A mes cousin(e)s et ami(e)s,

A Amina, Sabrina, Fella et Nadia pour m'avoir accompagnée durant ces cinq années,

A la mémoire de mes deux grand-mères,

Particulièrement, à ma très chère amie Almaz.

Narymane.

Je dédie ce travail

A mes parents, pour leur soutien et encouragements durant toutes ces années d'études.

A ma sœur Kenza, et mon frère Yacine,

A la mémoire de ma tante,

A mes ami(e)s, ceux de l'ENP et ceux de l'ENPEI ,

Particulièrement, aux petits diables de Boufarik.

Mounir.

REMERCIEMENTS

Nous tenons à exprimer notre reconnaissance à Mr Sadoune, qui nous a donné l'occasion de travailler sur un sujet passionnant, et pour sa confiance, ses conseils et son aide précieuse et particulièrement pour sa disponibilité et son dévouement.

Nous remercions profondément Mr Ait cheikh de nous avoir fait l'honneur de présider le jury.

Nos remerciements vont aussi à Mlle Moussaoui pour avoir accepté de juger notre travail.

Notre profonde gratitude va à toutes les personnes ayant contribué à notre formation.

Merci à nos camarades d'électronique, particulièrement l'équipe du laboratoire de télécommunications : Rafik, Rahid, Fatima, Amina, Kader, Ramzi, Sabrina, Chafika et Najet... pour leur compagnie, leur aide, et leur humour.

TABLE DES MATIERES

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

LISTE DES FIGURES

LISTE DES TABLEAUX

Acronymes

AH :	Application Header.
ARP:	Address Resolution Protocol.
ASICs :	Application Specific Integrated Circuits.
BAL:	Boite Aux Lettres.
BOOTP:	Bootstrap Protocol.
DH:	Data Header.
DHCP:	Dynamic Host Configuration Protocol.
DNS:	Domain Name Server.
EDK:	Embedded Developpement Kit.
FPGA:	Field Programmable Gate Array.
FTP:	File Transfer Protocol.
HDL:	Hardware Description Language.
Host ID:	Host Identifier.

IANA:	Internet Assigned Numbers Agency.
ICMP:	Internet Control Message Protocol.
IGMP:	Internet Group Management Protocol.
IHL:	Internet Header Length.
INTERNIC:	Internet Network Information Center.
IP:	Internet Protocol.
ISO:	International Standards Organisation.
ITU:	International Telecommunication Union.
LAN:	Local Area Network.
LDAP:	Lightweight Directory Access Protocol.
LOTOS:	Language Of Temporal Ordering Specification.
MAC:	MAChine.
MAN:	Metropolitan Area Network.
MSC:	Message Sequence Chart.
Net ID:	Network Identifier.
NH:	Network Header.
NNTP:	Network News Transfer.
NOC:	Network On Chip.
OS:	Operating System.
OSI:	Open System Interconnection.
PH:	Presentation Header.
RARP:	Reverse Address Resolution Protocol.
RFC:	Request For Comments.

SDL:	Specification and Description Language.
SH:	Session Header.
SMTP:	Simple Mail Transport Protocol.
SOC:	System On Chip.
TCP:	Transport Control Protocol.
TELNET:	TErmiaL Net Protocol.
TH:	Transport Header.
TTL:	Time To Live.
UDP:	User Datagramme Protocol.
UML:	Unified Modelling Language.
VHDL:	Vhsic Hardware Description Language.
VHSIC:	Very High Speed Integrated Circuit.
WAN:	Wide Area Network.

Introduction Générale

Les systèmes électroniques sont de plus en plus omniprésents dans notre vie. L'exemple de l'automobile est édifiant.

Leurs techniques de conception associées ont évolué au cours du temps influencées par les progrès technologiques aussi bien au sens du paradigme (full custom, asics, prototypage) qu'au sens de l'implémentation physique couplés aux outils y afférant. On peut citer comme exemple les progrès des décennies 50-60 (naissance du transistor puis du circuit intégré), 70-80 (Full Custom microprocesseur langages de description) 80/90 (asic circuit programmables), 90-2000 (IP – SOC - NOC).

Les « System On Chip » ou « Systèmes sur puce » font référence à l'intégration de tous les circuits électroniques nécessaires pour réaliser toutes les fonctions utiles à un système autonome sur un seul circuit. ainsi, au lieu de concevoir un produit électronique en assemblant diverses puces et composants sur une carte, la technologie Soc permet à tous les éléments d'être intégrés sur une seule puce aidé par cela par les progrès technologiques sus cités. Le Soc répond à plusieurs exigences résultantes de l'évolution du marché des systèmes

électroniques comme le « time to market », la réduction des coûts de fabrication, l'augmentation des performances au niveau vitesse et consommation, la possibilité de reconfiguration et la rapidité de développement.

Notre projet s'inscrit dans cette perspective qui consiste en la contribution à l'implémentation sur puce d'un service des plus utilisé dans les réseaux TCP/IP ; le service DHCP.

Nous commencerons par donner des notions générales sur les réseaux informatiques, ensuite nous exposerons en détails le fonctionnement du service DHCP. Le 3ème chapitre traitera de la méthodologie de développement à suivre et une description du langage évolué SDL utilisé pour la modélisation du système. L'étape suivante consistera en la présentation du modèle SDL du service DHCP, avec comme finalité la création d'un code C exploitable.

1 Généralités sur les réseaux

1.1 Introduction

Le développement accéléré d'Internet a vu le protocole du même nom prendre une importance considérable. Une configuration IP est nécessaire pour chaque ordinateur cherchant à se connecter sur un réseau TCP/IP, un service effectuant cette tâche s'est vite développé et sa nécessité consacrée.

Avant de présenter ce service, le service DHCP, il convient de mettre en évidence quelques notions sur les réseaux et leurs constituants, définir ce que sont un client et un serveur, et connaître leur comportement respectif. Une présentation générale des différents protocoles de communication utilisés dans les réseaux s'avère également nécessaire. Tel est l'objet de ce premier chapitre.

1.2 Présentation

Un réseau est un ensemble d'objets interconnectés les uns avec les autres. Il permet de faire circuler des éléments entre chacun de ces objets selon des règles bien définies.

- Réseau (Network) : Ensemble d'ordinateurs et périphériques connectés les uns aux autres. (Remarque : deux ordinateurs connectés constituent déjà un réseau).
- Mise en réseau (Networking) : Mise en oeuvre des outils et des tâches permettant de relier des ordinateurs afin qu'ils puissent partager des ressources.

Un **réseau informatique** peut être défini comme étant un ensemble d'ordinateurs reliés entre eux grâce à des lignes physiques et échangeant des informations sous forme de données numériques (valeurs binaires, c'est-à-dire codées sous forme de signaux pouvant prendre deux valeurs : 0 et 1)

Un réseau permet:

- Le partage de fichiers, d'applications ;
- La communication entre personnes (grâce au courrier électronique, la discussion en direct, ...) ;
- La communication entre processus (entre des machines industrielles) ;
- La garantie de l'unicité de l'information (bases de données) ;
- Standardisation des applications.

Les différents types de réseaux ont généralement les points suivant en commun:

- **Serveurs** : ordinateurs qui fournissent des ressources partagées aux utilisateurs par un serveur de réseau ;
- **Clients** : ordinateurs qui accèdent aux ressources partagées fournies par un serveur de réseau ;
- **Support de connexion** : conditionne la façon dont les ordinateurs sont reliés entre eux ;
- **Données partagées** : fichiers accessibles sur les serveurs du réseau ;
- **Imprimantes et autres périphériques partagés** : fichiers, imprimantes ou autres éléments utilisés par les usagers du réseau ;
- **Ressources diverses** : autres ressources fournies par le serveur.

Les différents types de réseau

On distingue généralement deux types de réseaux bien différents, ayant tout de même des similitudes.

- Les réseaux poste à poste (peer to peer / égal à égal)
- Réseaux organisés autour de serveurs (Client/Serveur)

Et particulièrement, différents types de réseaux (privés) selon leur taille (en terme de nombre de machines), leur vitesse de transfert des données ainsi que leur étendue. Les réseaux privés sont des réseaux appartenant à une même organisation. On fait généralement trois catégories de réseaux:

1. **LAN** (Local Area Network)

Il s'agit d'un ensemble d'ordinateurs appartenant à une même organisation et reliés entre eux dans une petite aire géographique par un réseau, souvent à l'aide d'une même technologie (la plus répandue étant Ethernet).

2. **MAN** (Metropolitan Area Network)

Un MAN interconnecte plusieurs LAN géographiquement proches (au maximum quelques dizaines de km) à des débits importants. Ainsi un MAN permet à deux noeuds distants de communiquer comme si ils faisaient partie d'un même réseau local. Un MAN est formé de commutateurs ou de routeurs interconnectés par des liens hauts débits (en général en fibre optique).

3. **WAN** (Wide Area Network)

Un WAN interconnecte plusieurs LANs à travers de grandes distances géographiques. Les débits disponibles sur un WAN résultent d'un arbitrage avec le coût des liaisons (qui augmente avec la distance) et peuvent être faibles. Les WAN fonctionnent grâce à des routeurs qui permettent de "choisir" le trajet le plus approprié pour atteindre un noeud du réseau. Le plus connu des WAN est Internet.

1.3 Le modèle OSI Réseau [1]

OSI signifie *Open Systems Interconnection*, ce qui se traduit par *Interconnexion de systèmes ouverts*. Ce modèle a été mis en place, en 1978 par l'ISO (International Standards Organisation) afin de mettre en place un standard de communications entre les ordinateurs d'un réseau, c'est-à-dire les règles qui gèrent les communications entre des ordinateurs. En effet, aux origines des réseaux chaque constructeur avait un système propre (on parle de système propriétaire). Ainsi de nombreux réseaux incompatibles coexistaient. C'est la raison pour laquelle l'établissement d'une norme a été nécessaire.

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

Le modèle OSI n'est pas une véritable architecture de réseau, car il ne précise pas réellement les services et les protocoles à utiliser pour chaque couche. Il décrit plutôt ce que doivent faire les couches.

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

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

1.3.1 Les différentes couches du modèle

Le modèle OSI comporte 7 couches :

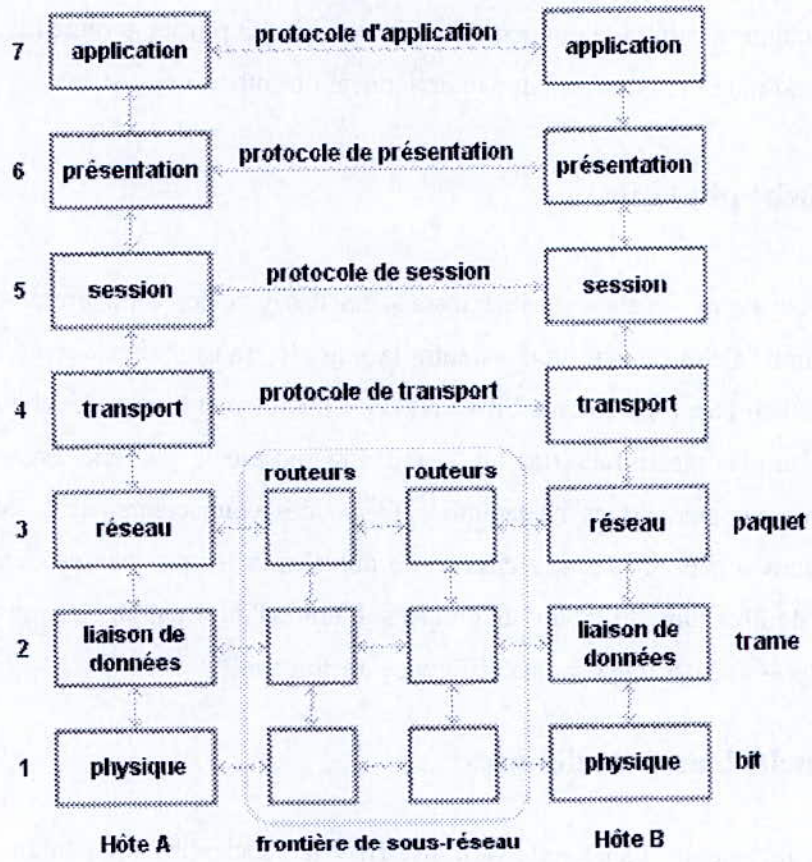


Figure1.1 : Les couches du modèle OSI

Les principes qui ont conduit à ces 7 couches sont les suivants :

- Une couche doit être créée lorsqu'un nouveau niveau d'abstraction est nécessaire ;
- Chaque couche a des fonctions bien définies ;
- Les fonctions de chaque couche doivent être choisies dans l'objectif de la normalisation internationale des protocoles ;
- Les frontières entre couches doivent être choisies de manière à minimiser le flux d'information aux interfaces ;
- Le nombre de couches doit être tel qu'il n'y ait pas cohabitation de fonctions très différentes au sein d'une même couche et que l'architecture ne soit pas trop difficile à maîtriser.

Les couches basses (1, 2, 3 et 4) sont nécessaires à l'acheminement des informations entre les extrémités concernées et dépendent du support physique. Les couches hautes (5, 6 et 7) sont responsables du traitement de l'information relative à la gestion des échanges entre systèmes informatiques. Par ailleurs, les couches 1 à 3 interviennent entre machines voisines, et non entre les machines d'extrémité qui peuvent être séparées par plusieurs routeurs. Les couches 4 à 7 sont au contraire des couches qui n'interviennent qu'entre hôtes distants.

1. La couche physique

La couche physique s'occupe de la transmission des bits de façon brute sur un canal de communication. Cette couche doit garantir la parfaite transmission des données (un bit 1 envoyé doit bien être reçu comme bit valant 1). Concrètement, cette couche doit normaliser les caractéristiques électriques (un bit 1 doit être représenté par une tension de 5 V, par exemple), les caractéristiques mécaniques (forme des connecteurs, de la topologie...), les caractéristiques fonctionnelles des circuits de données et les procédures d'établissement, de maintien et de libération du circuit de données. L'unité d'information typique de cette couche est le bit, représenté par une certaine différence de potentiel.

2. La couche liaison de données

Son rôle est un rôle de "liant" : elle va transformer la couche physique en une liaison a priori exempte d'erreurs de transmission pour la couche réseau. Elle fractionne les données d'entrée de l'émetteur en trames, transmet ces trames en séquence et gère les trames d'acquittement renvoyées par le récepteur. Rappelons que pour la couche physique, les données n'ont aucune signification particulière. La couche liaison de données doit donc être capable de reconnaître les frontières des trames.

La couche liaison de données doit être capable de renvoyer une trame lorsqu'il y a eu un problème sur la ligne de transmission. De manière générale, un rôle important de cette couche est la détection et la correction d'erreurs intervenues sur la couche physique. Cette couche intègre également une fonction de contrôle de flux pour éviter l'engorgement du récepteur. L'unité d'information de la couche liaison de données est la trame qui est composées de quelques centaines à quelques milliers d'octets maximum.

3. La couche réseau

C'est la couche qui permet de gérer le sous-réseau, i.e. le routage des paquets sur ce sous-réseau et l'interconnexion des différents sous-réseaux entre eux. Au moment de sa conception, il faut bien déterminer le mécanisme de routage et de calcul des tables de routage (tables statiques ou dynamiques...).

La couche réseau contrôle également l'engorgement du sous-réseau. On peut également y intégrer des fonctions de comptabilité pour la facturation au volume, mais cela peut être délicat.

L'unité d'information de la couche réseau est le paquet.

4. La couche transport

Cette couche est responsable du bon acheminement des messages complets au destinataire. Le rôle principal de la couche transport est de prendre les messages de la couche session, de les découper s'il le faut en unités plus petites et de les passer à la couche réseau, tout en s'assurant que les morceaux arrivent correctement de l'autre côté. Cette couche effectue donc aussi le réassemblage du message à la réception des morceaux. Cette couche est également responsable du type de service à fournir à la couche session, et finalement aux utilisateurs du réseau : service en mode connecté ou non, avec ou sans garantie d'ordre de délivrance, diffusion du message à plusieurs destinataires à la fois... Cette couche est donc également responsable de l'établissement et du relâchement des connexions sur le réseau.

Un des tous derniers rôles à évoquer est le contrôle de flux. L'unité d'information de la couche transport est le message.

5. La couche session

Cette couche organise et synchronise les échanges entre tâches distantes. Elle réalise le lien entre les adresses logiques et les adresses physiques des tâches réparties. Elle établit également une liaison entre deux programmes d'application devant coopérer et commande leur dialogue (qui doit parler, qui parle...). Dans ce dernier cas, ce service d'organisation s'appelle la gestion du jeton. La couche session permet aussi d'insérer des points de reprise dans le flot de données de manière à pouvoir reprendre le dialogue après une panne.

6. La couche présentation

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

7. La couche application

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

1.3.2 Transmission de données au travers du modèle OSI

Le processus émetteur remet les données à envoyer au processus récepteur à la couche application qui leur ajoute un en-tête application AH (éventuellement nul). Le résultat est alors transmis à la couche présentation.
La couche présentation transforme alors ce message et lui ajoute (ou non) un nouvel en-tête (éventuellement nul). La couche présentation ne connaît et ne doit pas connaître l'existence éventuelle de AH ; pour la couche présentation, AH fait en fait partie des données utilisateur.
Une fois le traitement terminé, la couche présentation envoie le nouveau "message" à la couche session et le même processus recommence.
Les données atteignent alors la couche physique qui va effectivement transmettre les données au destinataire. A la réception, le message va remonter les couches et les en-têtes sont progressivement retirés jusqu'à atteindre le processus récepteur, comme décrit ci-dessous.

Remarque : L'en-tête comporte des informations telles que l'adresse source et l'adresse de destination de la trame mais non des informations émanant du champ de données.

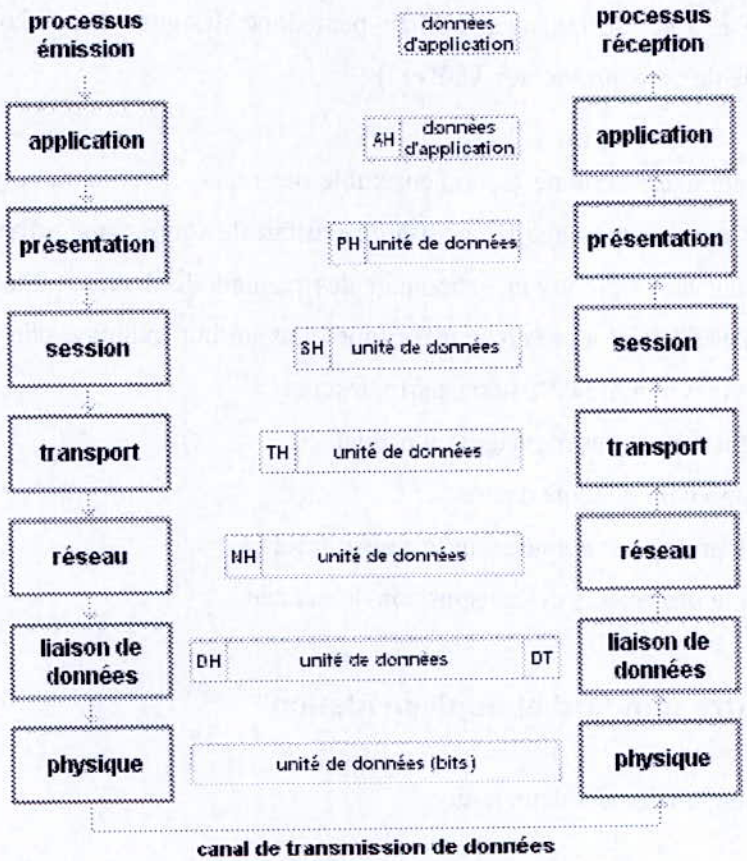


Figure1.2: Encapsulation de données

Le concept important est le suivant : il faut considérer que chaque couche est programmée comme si elle était vraiment horizontale, c'est à dire qu'elle dialoguait directement avec sa couche paire réceptrice. Au moment de dialoguer avec sa couche paire, chaque couche rajoute un en-tête et l'envoie (virtuellement, grâce à la couche sous-jacente) à sa couche paire.

1.4 Le modèle TCP/IP[1]

Le sigle TCP/IP signifie «**Transmission Control Protocol/Internet Protocol**».
TCP/IP désigne communément une **architecture réseau**, mais cet acronyme désigne en fait 2 protocoles étroitement liés : un protocole de transport, TCP (Transmission Control Protocol) qu'on utilise "par-dessus" un protocole réseau, IP (Internet Protocol). Ce qu'on entend par "modèle TCP/IP", c'est en fait une architecture réseau en 4 couches dans laquelle les protocoles TCP et IP jouent un rôle prédominant, car ils en constituent l'implémentation la

plus courante. Par abus de langage, TCP/IP peut donc désigner deux choses : le modèle TCP/IP et la suite de deux protocoles TCP et IP.

TCP/IP représente d'une certaine façon l'ensemble des règles de communication sur Internet et se base sur la notion adressage IP, c'est-à-dire le fait de fournir une **adresse IP** à chaque machine du réseau afin de pouvoir acheminer des paquets de données. Etant donné que la suite de protocoles TCP/IP a été créée à l'origine dans un but militaire, elle est conçue pour répondre à un certain nombre de critères parmi lesquels :

- Le fractionnement des messages en paquets ;
- L'utilisation d'un système d'adresses ;
- L'acheminement des données sur le réseau (routage) ;
- Le contrôle des erreurs de transmission de données.

Différence entre standard et implémentation

TCP/IP regroupe globalement deux notions :

- La notion de **standard** : TCP/IP représente la façon dont les communications s'effectuent sur un réseau.
- La notion d'**implémentation** : l'appellation TCP/IP est souvent étendue aux logiciels basés sur le protocole TCP/IP. TCP/IP est en fait un modèle sur lequel les développeurs d'applications réseau s'appuient. Les applications sont ainsi des implémentations du protocole TCP/IP.

Afin de pouvoir appliquer le modèle TCP/IP à n'importe quelles machines, c'est-à-dire indépendamment du système d'exploitation, le système de protocoles TCP/IP a été décomposé en plusieurs modules effectuant chacun une tâche précise. De plus, ces modules effectuent ces tâches les uns après les autres dans un ordre précis, on a donc un système stratifié.
Le modèle TCP/IP s'est progressivement imposé comme modèle de référence en lieu et place du modèle OSI. Cela tient tout simplement à son histoire. En effet, contrairement au modèle OSI, le modèle TCP/IP est né d'une implémentation ; la normalisation est venue ensuite.

Les 4 couches

Protocoles utilisés	Modèle TCP/IP	Modèle OSI
DHCP/DNS/….	Couche application	Couche application
		Couche Présentation
		Couche session
TCP / UDP	Couche Transport	Couche transport
IP / ARP /ICMP / RARP / IGMP	Couche Internet (IP)	Couche réseau
	Couche Accès réseau	Couche Liaison de donnée
		Couche Physique

Figure1.3 : Correspondance entre le modèle OSI et TCP/IP

Comme on peut le remarquer, les couches du modèle TCP/IP ont des tâches beaucoup plus diverses que les couches du modèle OSI, étant donné que certaines couches du modèle TCP/IP correspondent à plusieurs couches du modèle OSI.

Les rôles des différentes couches sont les suivants :

- **Couche Accès réseau** : elle spécifie la forme sous laquelle les données doivent être acheminées quel que soit le type de réseau utilisé
- **Couche Internet** : elle est chargée de fournir le paquet de données (datagramme)
- **Couche Transport** : elle assure l'acheminement des données, ainsi que les mécanismes permettant de connaître l'état de la transmission
- **Couche Application** : elle englobe les applications standard du réseau (Telnet, SMTP, FTP, ...)

Aussi, comme à chaque niveau, le paquet de données change d'aspect alors, les appellations changent suivant les couches :

- Le paquet de données est appelé **message** au niveau de la couche Application
- Le message est ensuite encapsulé sous forme de **segment** dans la couche Transport
- Le segment une fois encapsulé dans la couche Internet prend le nom de **datagramme**

- Enfin, on parle de **trame** au niveau de la couche Accès réseau

1. La couche Accès réseau

La couche accès réseau est la première couche de la pile TCP/IP, elle offre les capacités à accéder à un réseau physique quel qu'il soit, c'est-à-dire les moyens à mettre en oeuvre afin de transmettre des données via un réseau.

Ainsi, la couche accès réseau contient toutes les spécifications concernant la transmission de données sur un réseau physique, qu'il s'agisse de réseau local (Anneau à jeton - token ring, ethernet), de connexion à une ligne téléphonique ou n'importe quel type de liaison à un réseau. Elle prend en charge les notions suivantes :

- Acheminement des données sur la liaison ;
- Coordination de la transmission de données (synchronisation) ;
- Format des données ;
- Conversion des signaux (analogique/numérique) ;
- Contrôle des erreurs à l'arrivée ;
- ...

Heureusement toutes ces spécifications sont transparentes aux yeux de l'utilisateur, car l'ensemble de ces tâches est en fait réalisé par le système d'exploitation, ainsi que les drivers du matériel permettant la connexion au réseau (ex : driver de carte réseau).

2. La couche Internet :

La couche Internet est la couche "la plus importante" (elles ont toutes leur importance) car c'est elle qui définit les datagrammes, et qui gère les notions d'adressage IP. Elle permet l'acheminement des datagrammes (paquets de données) vers des machines distantes ainsi que de la gestion de leur fragmentation et de leur assemblage à réception.

Du fait du rôle imminent de cette couche dans l'acheminement des paquets, le point critique de cette couche est le **routage.**

La couche Internet contient les 5 protocoles suivants :

a. Le protocole IP [2]: (Internet Protocol) Ce protocole permet l'élaboration et le transport des datagrammes IP (les paquets de données), sans toutefois en assurer la "livraison". En réalité le protocole IP traite les datagrammes IP indépendamment les uns des autres en définissant leur représentation, leur routage et leur expédition. Les données contenues dans les datagrammes sont analysées (et éventuellement modifiées) par les routeurs permettant leur transit.

Voici ce à quoi ressemble un datagramme IP, constitué d'une en-tête suivie d'un champ de données:

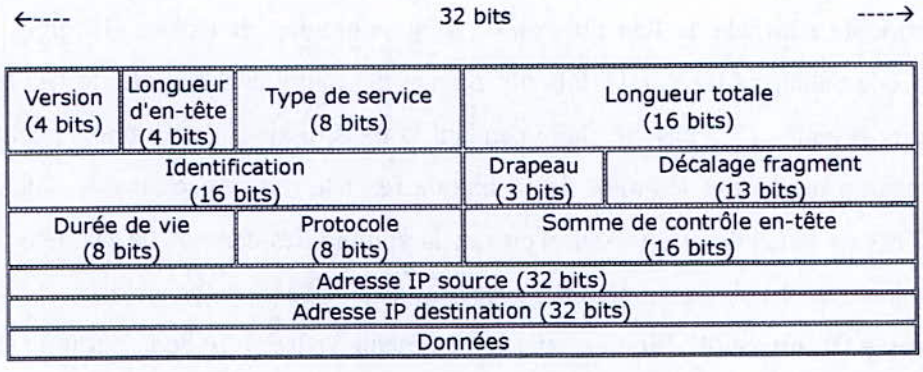


Figure 1.4 : Structure d'un datagramme IP

Signification des différents champs:

- **Version (4 bits)** : il s'agit de la version du protocole IP que l'on utilise (actuellement on utilise la version 4 IPv4) afin de vérifier la validité du datagramme. Elle est codée sur 4 bits.
- **Longueur d'en-tête, ou IHL pour Internet Header Length (4 bits)** : il s'agit du nombre de mots de 32 bits constituant l'en-tête (nota : la valeur minimale est 5). Ce champ est codé sur 4 bits.
- **Type de service (8 bits)** : il indique la façon selon laquelle le datagramme doit être traité.
- **Longueur totale (16 bits)**: il indique la taille totale du datagramme en octets. La taille de ce champ étant de 2 octets, la taille totale du datagramme ne peut dépasser 65536 octets. Utilisé conjointement avec la taille de l'en-tête, ce champ permet de déterminer où sont situées les données.
- **Identification, drapeaux (flags) et déplacement de fragment** sont des champs qui permettent la fragmentation des datagrammes, ils sont expliqués plus bas.
- **Durée de vie appelée aussi TTL, pour Time To Live (8 bits)** : 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 par les datagrammes perdus.

- **Protocole (8 bits)** : ce champ, en notation décimale, permet de savoir de quel protocole est issu le datagramme
  - ICMP : 1
  - IGMP : 2
  - TCP : 6
  - UDP : 17
- **Somme de contrôle de l'en-tête, ou en anglais header checksum (16 bits)** : ce champ contient une valeur codée sur 16 bits qui permet de contrôler l'intégrité de l'en-tête afin de déterminer si celui-ci n'a pas été altéré pendant la transmission. La somme de contrôle est le complément à un de tous les mots de 16 bits de l'en-tête (champ somme de contrôle exclu). Celle-ci est en fait telle que lorsque l'on fait la somme des champs de l'en-tête (somme de contrôle incluse), on obtient un nombre avec tous les bits positionnés à 1
- **Adresse IP source (32 bits)** : ce champ représente l'adresse IP de la machine émettrice, il permet au destinataire de répondre
- **Adresse IP destination (32 bits)** : adresse IP du destinataire du message

b. Le protocole ARP (*Address Resolution protocol*) : ce protocole a un rôle phare parmi les protocoles de la couche Internet de la suite TCP/IP, car il permet de connaître l'adresse physique d'une carte réseau correspondant à une adresse IP.

c. Le protocole ICMP (*Internet Control Message Protocol*) : il s'agit d'un protocole qui permet de gérer les informations relatives aux erreurs aux machines connectées. Etant donné le peu de contrôles que le protocole IP réalise, il permet non pas de corriger ces erreurs mais de faire part de ces erreurs aux protocoles des couches voisines.

d. Le protocole RARP (*Reverse Address Resolution Protocol*) : signifie *Protocole ARP inversé*, il s'agit donc d'une sorte d'annuaire inversé des adresses logiques et physiques.
En réalité le protocole RARP est essentiellement utilisé pour les stations de travail n'ayant pas de disque dur et souhaitant connaître leur adresse physique...

e. Le protocole IGMP (*Internet Group Management Protocol*) : permet de gérer les déclarations d'appartenance à un ou plusieurs groupes auprès des routeurs.

3. La couche Transport

Son rôle est le même que celui de la couche transport du modèle OSI : permettre à des entités paires de soutenir une conversation.

Les protocoles des couches précédentes permettaient d'envoyer des informations d'une machine à une autre. La couche transport permet à des applications tournant sur des machines distantes de communiquer. Le problème consiste à identifier ces applications.

En effet, suivant la machine et son système d'exploitation, l'application pourra être un programme, une tâche, un processus...

De plus, la dénomination de l'application peut varier d'un système à un autre, c'est la raison pour laquelle un système de numéro a été mis en place afin de pouvoir associer un type d'application à un type de données, ces identifiants sont appelés **ports.**

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 :

a. **TCP** : Un protocole fiable **orienté connexion** qui assure le contrôle des erreurs c'est-à-dire qu'il permet à deux machines qui communiquent de contrôler l'état de la transmission. Grâce à ce protocole TCP, les applications peuvent communiquer de façon sûre (grâce au système d'accusés de réception du protocole TCP), indépendamment des couches inférieures. Cela signifie que les routeurs (qui travaillent dans la couche Internet) ont pour seul rôle l'acheminement des données sous forme de datagrammes, sans se préoccuper du contrôle des données, car celui-ci est réalisé par la couche transport (plus particulièrement par le protocole TCP).

b. **UDP (user Datagramme Protocole):** Un protocole non orienté connexion dont le contrôle d'erreur est archaïque.

Il 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 désigne simplement, les numéros de port correspondant aux applications envisagées avec des temps de réponse courts.

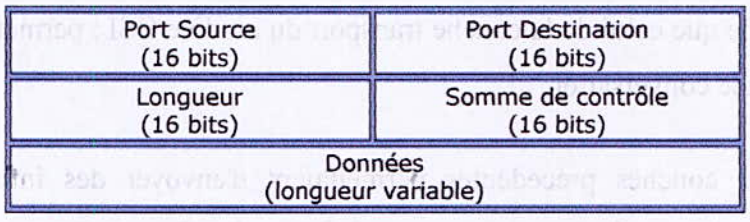


Figure 1.5 : Format d'un message UDP

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.

4. La couche Application

La couche application est la couche située au sommet des couches de protocoles TCP/IP. Celle-ci contient les applications réseaux permettant de communiquer grâce aux couches inférieures.

Les logiciels de cette couche communiquent donc grâce à un des deux protocoles de la couche inférieure (la couche transport) c'est-à-dire TCP ou UDP.

Les applications de cette couche sont de différents types, mais la plupart sont des services réseau, c'est-à-dire des applications fournies à l'utilisateur pour assurer l'interface avec le système d'exploitation. On peut les classer selon les services qu'ils rendent :

- Les services de gestion (transfert) de fichier et d'impression (**FTP**: "File Transfert Protocol ;
- Les services de connexion au réseau et à distance (**TELNET**: protocole permettant de se connecter sur une machine distante (serveur) en tant qu'utilisateur) ;
- La gestion des mails (**SMTP**: "Simple Mail Transport protocol") ;
- Les utilitaires Internet divers;

1.5 L'adressage IP[2]

Le système d'adressage défini avec le protocole réseau de la modélisation TCP/IP est incontournable dans la mise en oeuvre des réseaux actuels

Le rôle fondamental de la couche réseau (niveau 3 du modèle OSI) est de déterminer la *route* que doivent emprunter les paquets. Cette fonction de recherche de chemin nécessite une identification de tous les hôtes connectés au réseau. De la même façon que l'on repère l'adresse postale d'un bâtiment à partir de la ville, la rue et un numéro dans cette rue, on identifie un hôte réseau par une *adresse* qui englobe les mêmes informations.

Le modèle TCP/IP utilise un système particulier d'adressage qui porte le nom de la couche réseau de ce modèle : *l'adressage IP*

Par exemple, *194.153.205.26* est une adresse TCP/IP donnée sous une forme technique. Ce sont ces adresses que connaissent les ordinateurs qui communiquent entre eux.

C'est l'**IANA** (*Internet Assigned Numbers Agency*) qui est chargée d'attribuer ces numéros.

1.5.1 Description d'une adresse IP

Une adresse IP est une adresse 32 bits notée sous forme de 4 nombres entiers séparés par des points. On distingue en fait deux parties dans l'adresse IP:

- une partie des nombres à gauche désigne le réseau (on l'appelle *netID*)
- Les nombres de droite désignent les ordinateurs de ce réseau (on l'appelle *host-ID*)

Prenons un exemple:

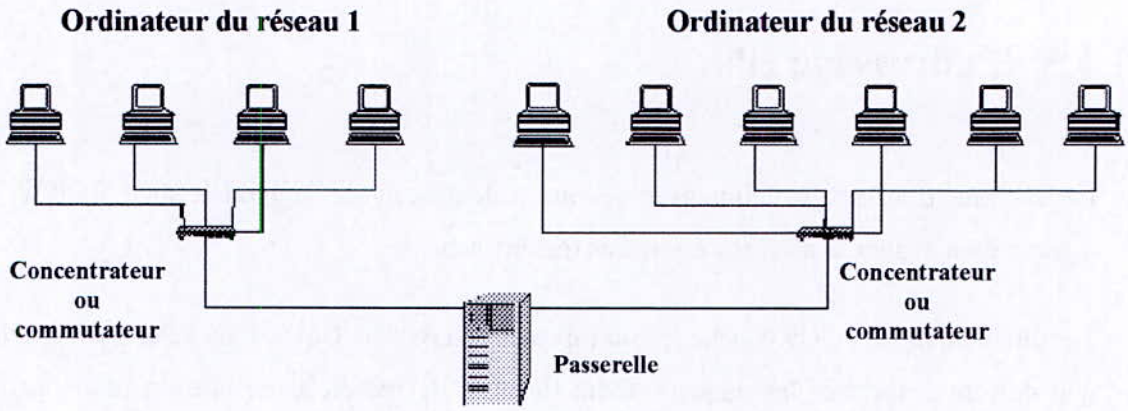


Figure 1.6 Exemple d'un réseau

Notons le réseau de gauche 194.28.12. Il contient alors les ordinateurs suivants:

- 194.28.12.1 à 194.28.12.4

Notons celui de droite 178.12.77. Il comprendra les ordinateurs suivants:

- 178.12.77.1 à 178.12.77.6

Les réseaux sont donc notés 194.28.12 et 178.12.77, puis on numérote en incrémentant, chacun des ordinateurs le constituant.
Imaginons un gros réseau noté *58.24*: on donnera généralement aux ordinateurs reliés à lui les adresses IP allant de *58.24.0.1* à *58.24.255.254*. Il s'agit donc d'attribuer les numéros de telle façon qu'il y ait une organisation dans la hiérarchie des ordinateurs et des serveurs...
Ainsi, plus le nombre de bits réservé au réseau est petit, plus celui-ci peut contenir d'ordinateurs. En effet un réseau noté 102 peut contenir des ordinateurs dont l'adresse IP peut aller de 102.0.0.1 à 102.255.255.254 (256*256*256-2=16777214 possibilités), tandis qu'un réseau noté 194.26 ne pourra contenir que des ordinateurs dont l'adresse IP sera comprise entre 194.26.0.1 et 194.26.255.254 (256*256-2=65534 possibilités), c'est la notion de **classe**.

Adresses particulières :

Lorsque l'on annule la partie host-id, c'est-à-dire lorsque l'on remplace les bits réservés aux machines du réseau, on obtient ce que l'on appelle l'**adresse réseau**.

Ainsi, 194.28.12.0 est une adresse réseau et on ne peut donc pas l'attribuer à un des ordinateurs du réseau

Lorsque l'on annule la partie netid, c'est-à-dire lorsque l'on remplace les bits réservés au réseau, on obtient ce que l'on appelle l'**adresse machine**. Cette adresse représente la machine spécifiée par le host-ID qui se trouve sur le réseau courant.

Lorsque tous les bits de la partie host-id sont à 1, on obtient ce que l'on appelle l'**adresse de diffusion** (en anglais **broadcast**), c'est-à-dire une adresse qui permettra d'envoyer le message à toutes les machines situées sur le réseau spécifié par le *netID*.

Lorsque tous les bits de la partie netid sont à 1, on obtient ce que l'on appelle l'**adresse de diffusion limitée (multicast)**.

L'adresse **127.0.0.1** est appelée **adresse de boucle locale** (en anglais **loopback**), car elle désigne la machine locale (en anglais *localhost*).

1.5.2 Les classes de réseaux

Les adresses IP sont réparties en trois classes, ce qui permet de gérer des réseaux de tailles diverses

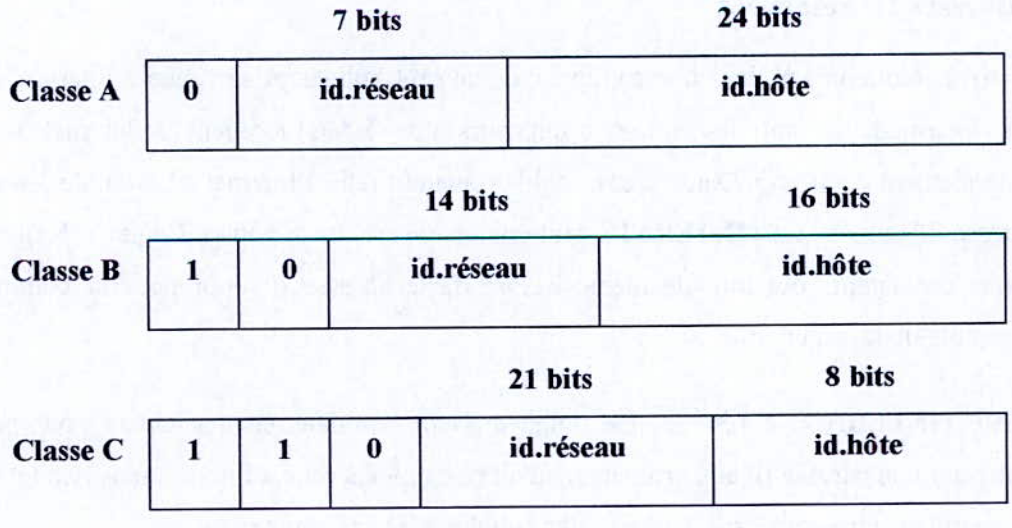


Figure 1.7: Les trois classes d'adresse

Ainsi, les adresses de classe A ont leurs premiers octets compris entre 0 et 127, les adresses de classe B entre 128 et 191 et les adresses de classe C entre 192 et 223

1.5.3 Attribution des adresses IP

Le but de la division des adresses IP en trois classes A, B et C est de faciliter la recherche d'un ordinateur sur le réseau. En effet avec cette notation il est possible de rechercher dans un premier temps le réseau que l'on désire atteindre puis de chercher un ordinateur sur celui-ci. Ainsi l'attribution des adresses IP se fait selon la taille du réseau.

Tableau1.1 : Nombre des réseau et machines dans chaque classe

Classe	Nombre de réseaux possibles	Nombre d'ordinateurs maxi sur chacun
A	126	16777214
B	16384	65534
C	2097152	254

Les adresses de classe A sont réservées aux très grands réseaux, tandis que l'on attribuera les adresses de classe C à des petits réseaux d'entreprise par exemple

Adresses IP réservées

Il arrive fréquemment dans une entreprise qu'un seul ordinateur soit relié à Internet, c'est par son intermédiaire que les autres ordinateurs du réseau accèdent à Internet (on parle généralement de proxy). Dans ce cas, seul l'ordinateur relié à Internet a besoin de réserver une adresse IP auprès de l'**INTERNIC** (Internet Network Information Center). Toutefois, les autres ordinateurs ont tout de même besoin d'une adresse IP pour pouvoir communiquer ensemble de façon interne.

Ainsi, l'**INTERNIC** a réservé une poignée d'adresses dans chaque classe pour permettre d'affecter une adresse IP aux ordinateurs d'un réseau local relié à Internet sans risquer de créer de conflits d'adresses IP sur le réseau. Il s'agit des adresses suivantes:

- 10.0.0.1 à 10.255.255.254
- 172.16.0.1 à 172.31.255.254

- 192.168.0.1 à 192.168.255.254

1.5.4 Masques de sous-réseau

En résumé, on fabrique un masque contenant des 1 aux emplacements des bits que l'on désire conserver, et des 0 pour ceux que l'on veut rendre égaux à zéro. Une fois ce masque créé, il suffit de faire un ET entre la valeur que l'on désire masquer et le masque afin de garder intacte la partie que l'on désire et annuler le reste.

Ainsi, un masque réseau (en anglais *netmask*) se présente sous la forme de 4 octets séparés par des points (comme une adresse IP), il comprend (dans sa notation binaire) des zéros aux niveau des bits de l'adresse IP que l'on veut annuler (et des 1 au niveau de ceux que l'on désire conserver).

Intérêt d'un masque

Il y en a en fait plusieurs. Un d'entre eux est de pouvoir connaître le réseau associé à une adresse IP. En effet, comme nous l'avons vu précédemment, le réseau est déterminé par un certain nombre d'octets de l'adresse IP (1 octet pour les adresses de classe A, 2 pour les adresses de classe B, et 3 octets pour la classe C). De plus, nous avons vu que l'on note un réseau en prenant le nombre d'octets qui le caractérise, puis en complétant avec des 0.

Ainsi, le réseau associé à l'adresse *34.56.123.12* est *34.0.0.0* (puisqu'il s'agit d'une adresse de classe A). Il suffit donc pour connaître l'adresse du réseau associé à l'adresse IP 34.56.123.12 d'appliquer un masque dont le premier octet ne comporte que des 1 (ce qui donne 255), puis des 0 sur les octets suivants (ce qui donne 0..).

Le masque est: 11111111.00000000.00000000.00000000 Le masque associé à l'adresse IP 34.208.123.12 est donc 255.0.0.0. La valeur binaire de 34.208.123.12 est:
00100010.11010000.01111011.00001100
Un ET entre 00100010.11010000.01111011.00001100
ET
11111111.00000000.00000000.00000000
donne
00100010.00000000.00000000.00000000
C'est-à-dire *34.0.0.0*, c'est bien le réseau associé à l'adresse *34.208.123.12*

En généralisant, on obtient les masques suivants pour chaque classe:

- Pour une adresse de **Classe A**, seul le premier octet nous intéresse, on a donc un masque de la forme *11111111.00000000.00000000.00000000*, c'est-à-dire en notation décimale:
  255.0.0.0
- Pour une adresse de **Classe B**, les deux premiers octets nous intéresse, on a donc un masque de la forme *11111111.11111111.00000000.00000000*, c'est-à-dire en notation décimale: **255.255.0.0**
- Pour une adresse de **Classe C** on s'intéresse aux trois premiers octets, on a donc un masque de la forme *11111111.11111111.11111111.00000000*, c'est-à-dire en notation décimale:
  255.255.255.0

1.6 Architecture Client/serveur

1.6.1 Définition d'un Serveur

On appelle logiciel serveur un programme qui offre un service sur le réseau. Le serveur accepte des requêtes, les traite et renvoie le résultat au demandeur. Le terme serveur s'applique à la machine sur lequel s'exécute le logiciel serveur.

1.6.2 Définition d'un Client

On appelle logiciel client un programme qui utilise le service offert par un serveur. Le client émet une requête vers le serveur grâce à son adresse et le port, qui désigne un service particulier du serveur, et reçoit une réponse.

1.6.3 Définition d'une architecture Client/serveur

C'est la description du fonctionnement coopératif entre le serveur et le client. Les services Internet sont conçus selon cette architecture. Ainsi, chaque application est composée de logiciel serveur et logiciel client. A un logiciel serveur, peut correspondre plusieurs logiciels clients développés dans différents environnements: Unix, Mac, PC...; la seule obligation est le respect du protocole entre les deux processus communicants.

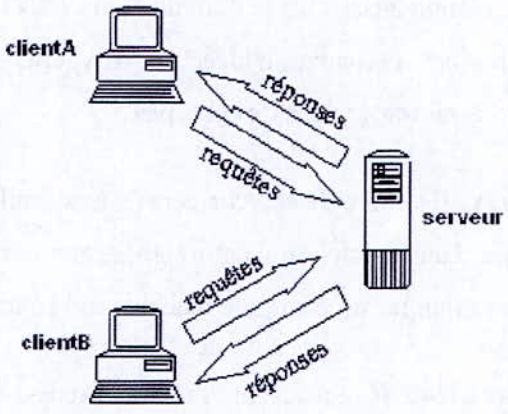


Figure1.8 : Fonctionnement d'un système client / serveur

1.6.4 Les différents Services

Nous citerons ici quelques services mis en œuvre dans un réseau informatique:

Le service d'émulation de terminal: Le service **Telnet** (TErminaL NET-protocol) permet d'utiliser une station en réseau comme un terminal informatique directement raccordé au calculateur. Dans la majorité des cas, il faut disposer d'un compte sur le calculateur en question.

Le courrier électronique: Le service **Mail** (Messagerie) s'appuie sur le protocole **SMTP** (Simple Mail Transfer Protocol). Ce service est le plus utilisé car le plus utile pour les différentes catégories d'utilisateurs. Pour pouvoir recevoir du courrier, il faut disposer d'une BAL (Boîte Aux Lettres ou MailBox) sur un calculateur relié à l'Internet en permanence.

Le transfert de fichiers: Le protocole **FTP** permet outre le transfert, l'accès et la gestion de fichiers à distance. Cette manipulation sur les fichiers doit être autorisée. Des serveurs FTP anonymes permettent d'accéder à une partie de leurs données quelque soit l'utilisateur client.

Les outils de navigation: Internet est un réseau d'informations. Des outils permettent de naviguer dans cette toile d'araignée: **World Wide Web** (accès par hyperliens).

Le forum électronique: Connu aussi sous le nom de **News** (**NNTP** - Network News Transfer Protocol), ce service permet l'échange d'idées et d'expériences. Les différents thèmes intéressant les utilisateurs sont traités dans des groupes.

Serveur d'application(s) : Ce genre de serveur permet à des utilisateurs connectés en réseau d'accéder à tout ou partie d'un logiciel applicatif (programme, interface graphique, données, ...) à partir d'un exemplaire unique situé sur une machine informatique.

Serveur d'annuaire ou LDAP (Lightweight Directory Access Protocol) : Ce protocole est un protocole de consultation d'annuaire sur Internet. Les annuaires permettent de partager des bases d'informations sur le réseau interne ou externe. Ces bases peuvent contenir toute sorte d'information que ce soit des coordonnées de personnes ou des données systèmes.

Serveur de Noms de domaines utilisant le protocole **DNS** (Domain Name Server) : Ce protocole permet de faire la correspondance entre les numéros IP et les noms symboliques des machines (et inversement). (exemple domaine.com correspondant à l'adresse 195.83.163.60).

1.6.5 Le Service DHCP

1.6.5.a Principe du service DHCP

Un serveur DHCP (*Dynamic Host Configuration Protocol*) a pour rôle d'attribuer une configuration réseau à des clients pour une durée déterminée.

Au lieu d'affecter manuellement à chaque hôte (client) une adresse statique, ainsi que tous les paramètres tels que (serveur de noms, passerelle par défaut, nom du réseau), un serveur DHCP alloue à un client, pour une durée déterminée, un accès au réseau,. Le serveur passe en paramètres au client toutes les informations dont il a besoin.

Tous les noeuds critiques du réseau (serveur de nom primaire et secondaire, passerelle par défaut) ont une adresse IP statique ; en effet, si celle-ci variait, ce processus ne serait plus réalisable.

Ce processus est mis en œuvre, par exemple, lorsque on ouvre une session chez un fournisseur d'accès Internet par modem. Le fournisseur d'accès, alloue une adresse IP de son réseau le temps de la liaison. Cette adresse est libérée, donc de nouveau disponible, lors de la fermeture de la session.

1.6.5.b Notion de bail

Pour des raisons d'optimisation des ressources réseau, les adresses IP sont délivrées pour une durée limitée. C'est ce qu'on appelle un bail (lease en anglais). Un client qui voit son bail arriver à terme peut demander au serveur un renouvellement du bail. Si le serveur ne reçoit pas de demandes de renouvellement valide, il rend disponible l'adresse IP. C'est toute la subtilité du DHCP : on peut optimiser l'attribution des adresses IP en jouant sur la durée des baux

Sur un réseau où beaucoup d'ordinateurs se connectent et se déconnectent souvent (réseau d'école ou de locaux commerciaux par exemple), il est intéressant de proposer des baux de courte durée. A l'inverse, sur un réseau constitué en majorité de machines fixes, très peu souvent rebootées, des baux de longues durées suffisent.

1.6.5.c Avantage du service DHCP

1. Le protocole DHCP offre une configuration de réseau TCP/IP fiable et simple, empêche les conflits d'adresses et permet de contrôler l'utilisation des adresses IP de façon centralisée. Ainsi, si un paramètre change au niveau du réseau, comme, par exemple l'adresse de la passerelle par défaut, il suffit de changer la valeur du paramètre au niveau du serveur DHCP, pour que toutes les stations aient une prise en compte du nouveau paramètre dès que le bail sera renouvelé. Dans le cas de l'adressage statique, il faudrait manuellement reconfigurer toutes les machines ;
2. **économie d'adresses** : ce protocole est presque toujours utilisé par les fournisseurs d'accès Internet qui disposent d'un nombre d'adresses limité. Ainsi grâce à DHCP, seules les machines connectées en ligne ont une adresse IP. En effet, imaginons un fournisseur d'accès qui a plus de 1000 clients. Il lui faudrait 5 réseaux de classe C, s'il voulait donner à chaque client une adresse IP particulière. S'il se dit que chaque client utilise en moyenne un temps de connexion de 10 mn par jour, il peut s'en sortir avec une seule classe C, en attribuant, ce que l'on pourrait appeler des "jetons d'accès" en fonction des besoins des clients ;
3. Les postes itinérants sont plus faciles à gérer ;
4. Le changement de plan d'adressage se trouve facilité par le dynamisme d'attribution.

1.6.5.d Inconvénient

Le client utilise des trames de **broadcast (diffusion)** pour rechercher un serveur DHCP sur le réseau, ce qui charge le réseau. Si par exemple, dans une entreprise, plusieurs centaines de personnes ouvrent leur session le matin à 8 h ou l'après midi à 14 h, il peut s'en suivre de graves goulets d'étranglement sur le réseau. L'administrateur devra donc réfléchir sérieusement à l'organisation de son réseau.

1.7 Conclusion

Ce chapitre nous a permis de remarquer la nécessité concrète du service DHCP, compte tenu du fait que les adresses IP à spécifier pour chaque machine, et les paramètres réseau à leur rajouter, sont peu évidents à retenir pour un utilisateur et difficiles à gérer pour un administrateur. Dans le chapitre qui suit nous décrirons plus en détails le fonctionnement du serveur DHCP.

2 Fonctionnement du service DHCP

2.1. Introduction

Les RFC (*Request For Comments*) sont un ensemble de documents qui font référence auprès de la Communauté Internet et qui décrivent, spécifient, standardisent et débattent de la majorité des normes, standards, technologies et protocoles liés à Internet et aux réseaux en général. Le Fonctionnement du DHCP est décrit en détails dans les RFCs : 1541, 2131, 2132 et la description qui suit, ne fait que les résumer.

2.2 Fonctionnement du service DHCP

DHCP fonctionne sur le modèle client-serveur : un serveur, qui détient la politique d'attribution des configurations IP, envoie une configuration donnée pour une durée donnée à un client donné (typiquement, une machine qui vient de démarrer). Le serveur va servir de base pour toutes les requêtes DHCP (il les reçoit et y répond), aussi doit-il avoir une configuration IP fixe.

2.2.1 Structure des messages DHCP

Le protocole DHCP est dérivé du protocole BOOTP[3] qui lui, permettait aux hôtes n'ayant pas de disque dur, de configurer automatiquement leurs paramètres réseau afin de profiter de celui-ci.

La figure 2.1 donne le format d'un message DHCP et le tableau 2.1 décrit chacun des ces champs. Les nombres entre parenthèses indiquent la taille de chaque champ en octets. Les noms pour les champs donnés dans la figure 2.1 seront utilisés au travers de ce document pour faire référence aux champs des messages DHCP

octet 1	octet 2	octet 3	octet 4
op (1)	htype (1)	hlen (1)	hops (1)
xid (4)			
secs (2)		flags (2)	
ciaddr (4)			
yiaddr (4)			
siaddr (4)			
giaddr (4)			
chaddr (16)			
sname (64)			
file (128)			
options (variable)			

Figure 2.1: Format d'un message DHCP

Tableau 2.1: Description des champs dans un message DHCP

CHAMP	OCTETS	DESCRIPTION
'op'	1	Vaut 1 pour BOOTREQUEST (requête client), 2 pour BOOTREPLY (réponse serveur)
'htype'	1	Type d'adresse matérielle ; par ex., '1' = Ethernet 10Mb.
'hlen'	1	Longueur de l'adresse matérielle (par ex. '6' for Ethernet 10Mb).
'hops'	1	Mis à zéro par le client, utilisé de manière optionnelle par les agents de relais quand on démarre via un agent de relais
'xid'	4	Identifiant de transaction, un nombre aléatoire choisi par le client, utilisé par le client et le serveur pour associer les messages et les réponses entre un client et un serveur
'secs'	2	Rempli par le client, les secondes s'écoulent depuis le processus d'acquisition ou de renouvellement d'adresse du client
'flags'	2	Drapeaux ; dont le bit le plus fort est mis a zéro dans le cas d'un diffusion du message client-serveur ou serveur-client..
'ciaddr'	4	Adresse IP des clients, rempli seulement si le client est dans un état AFFECTÉ, RENOUVELLEMENT ou REAFFECTATION
'yiaddr'	4	'votre' (client) adresse IP.
'siaddr'	4	Adresse IP du prochain serveur à utiliser pour le processus de démarrage; retournée par le serveur dans DHCPOFFER et DHCPACK.
'giaddr'	4	Adresse IP de l'agent de relais, utilisée pour démarrer via un agent de relais.
'chaddr'	16	Adresse matérielle des clients.
'sname'	64	Nom d'hôte du serveur optionnel, chaîne de caractères terminée par un caractère nul.

'fichier'	128	Nom du fichier de démarrage, chaîne terminée par un caractère nul ; nom "generic" ou nul dans le DHCPDISCOVER, nom du répertoire explicite dans DHCPOFFER.
'Options'	var	Champ de paramètres optionnels. (RFC 2132)

Le champ *'options'* limité à 64 octets par exemple pour la première version de Bootp est maintenant de longueur variable. Un client DHCP doit s'attendre à recevoir des messages DHCP avec un champ *'options'* d'au moins 312 octets. Ce qui implique qu'un client DHCP doit s'attendre à la réception d'un message jusqu'à 576 octets, taille minimum d'un datagramme qu'une machine IP doit savoir accepter.

Le passage de paramètres (nom de la machine, serveur de nom de domaine...) se fait par l'intermédiaires d'options. Les options sont documentées dans la RFC 2132. Elles portent toutes un numéro qui les identifie. Par exemple, l'option 15 est celle qui permet de donner au client le nom de domaine du réseau. Il est bien entendu possible d'envoyer plusieurs options dans le même message DHCP ; dans tous les cas, que l'on ne transmette qu'une seule option utile ou plusieurs, on doit toujours finir la zone d'options par une option 255 (*end*). Le format des options est le suivant :

octet 1	**octet 2**	**données...**
Code de l'option	longeur champ de données	...

Figure 2.2 : Le format des options

Le numéro de l'option n'est codé que sur 1 octet, donc il ne peut y avoir que 256 options possibles. L'octet 2 code la longueur du champ de données qui suit. . Le client DHCP a la possibilité d'imposer au serveur DHCP une taille maximum pour le champ d'options (option 57). La conséquence d'une telle limitation est que le serveur peut manquer de place pour envoyer toutes les options souhaitées. Pour répondre à ce problème, le serveur est autorisé à utiliser les champs *'sname'* et *'fichier'* pour finir son envoi. Le client est averti de cet usage par l'option 52 dans la zone d'options.

2.2.2 Le protocole Client-Serveur

Le message DHCP a la structure donnée dans le tableau 2.1 et la figure 2.1. Les 4 premiers octets du champ d'options doivent être initialisés respectivement avec les valeurs 99, 130, 83 et 99 (valeurs en décimal), cette séquence est appelée le *"magic cookie"* (gâteau magique en français).

L'option *"type du message DHCP"* doit être incluse dans chaque message. Cette option définit le "type" du message DHCP. Des options supplémentaires peuvent être permises, requises ou non permises ; cela dépend du type de message DHCP.

Au travers de ce document, les messages DHCP qui contiennent une option *''Type du message DHCP''* seront liés au type du message. Par ex : un message DHCP avec un *''type du message DHCP''* option type 1 seront considérés comme des messages *DHCPDISCOVER*.

Tableau 2.2: messages DHCP

nom	Description
DHCPDISCOVER (1)	Pour localiser les serveurs DHCP disponibles et demander une première configuration
DHCPOFFER (2)	Réponse du serveur à un message DHCPDISCOVER, qui contient les premiers paramètres
DHCPREQUEST (3)	Requête diverse du client pour par exemple prolonger son bail
DHCPDECLINE (4)	Le client annonce au serveur que l'adresse est déjà utilisée
DHCPPACK (5)	Réponse du serveur qui contient des paramètres et l'adresse IP du client
DHCPNAK (6)	Réponse du serveur pour signaler au client que son bail est arrivé a terme ou si le client a une mauvaise configuration réseau
DHCPRELEASE (7)	Le client libère son adresse IP
DHCPINFORM (8)	Le client demande des paramètres locaux, il a déjà une adresse IP

2.2.2.a L'interaction client - serveur - allocation d'une adresse réseau

Le résumé suivant du protocole d'échange entre clients et serveurs se réfère aux messages DHCP décrits dans le tableau 2, La valeur entre parenthèses correspondant à la valeur que doit avoir l'option *''type de message''*. Le diagramme temporel de la figure 2.2.3 montre les relations temporelles dans une interaction typique entre un client et un serveur. Si le client connaît déjà son adresse, certaines étapes peuvent être omises, cette interaction réduite est décrite dans la section suivante :

1. Le client diffuse un message *DHCPDICOVER* sur son réseau local physique. Le message *DHCPDISCOVER* peut inclure des options qui suggèrent des valeurs pour les adresses réseau et la durée du bail. Les agents de relais BOOTP peuvent passer le message sur des serveurs DHCP qui ne se trouvent pas sur le même sous réseau physique.

Remarque : Un agent de relais BOOTP est un routeur capable de reconnaître, traiter et relayer les message de protocole BOOTP et DHCP.

2. Chaque serveur peut répondre avec un message *DHCPOFFER* qui inclut une adresse réseau valide dans le champ *'yiaddr'* (et d'autres paramètres de configuration des options DHCP). Le serveur transmet un message *DHCPOFFER* au client, en utilisant l'agent de relais BOOTP si nécessaire.

3. Le client reçoit un ou plusieurs messages *DHCPOFFER* d'un ou plusieurs serveurs. Le client peut choisir d'attendre des réponses multiples. Le client choisit un serveur pour ses paramètres de configuration. Basé sur la configuration présente dans le *DHCPOFFER*, le client diffuse un message *DHCPREQUEST* qui doit inclure l'option *'identifiant serveur'* indiquant quel serveur il a sélectionné et qui peut inclure d'autres options spécifiant les valeurs de configuration désirées. L'option *'adresse IP demandée'* doit être réglée sur la même valeur que *'yiaddr'* du message *DHCPOFFER* provenant du serveur. Le client clôture à la fin d'un délai d'attente et retransmet le message *DHCPDISCOVER* si il ne reçoit pas de messages *DHCPOFFER*.

4. Les serveurs reçoivent les diffusions *DHCPREQUEST* des clients. Les serveurs qui ne sont pas sélectionnés par le message *DHCPREQUEST* interprètent ce message comme une notification que le client décline leur offre. Le serveur sélectionné dans le message *DHCPREQUEST* initie une transaction avec le client dans sa mémoire permanente et répond avec un message *DHCPACK* qui contient la configuration pour le client demandeur. La combinaison entre l'option *'identifiant client'* ou *'chaddr'* et l'adresse réseau assignée constitue un identifiant unique pour le bail du client. Elle est utilisée à la fois par le client et le serveur pour identifier un bail auquel il sera fait référence dans tous les messages DHCP. Aucun paramètre de configuration contenu dans le message *DHCPACK* ne devrait produire de conflit avec ceux du précédent message *DHCPOFFER* auquel le client a répondu. Le *'yiaddr'* du *DHCPACK* est initialisé avec l'adresse réseau sélectionnée.

5. Le client reçoit un message *DHCPACK* avec les paramètres de configuration. Le client devrait faire une vérification finale sur les paramètres (en envoyant des requêtes ARP), et noter la durée du bail spécifié dans le *DHCPACK*. A ce moment, le client est configuré. Si le client détecte que l'adresse est déjà utilisée (par ex : via l'utilisation de ARP) le client doit envoyer un *DHCPDECLINE* au serveur et relancer le processus de configuration. Le client devrait attendre un minimum de dix secondes avant de relancer la configuration pour éviter un trafic réseau excessif dans le cas d'un bouclage.

6. Le client peut choisir de renoncer au bail sur une adresse réseau en envoyant un *DHCPRELEASE* au serveur. Le client identifie le bail qu'il libère avec son *'identifiant client'* ou *'chaddr'* et l'adresse réseau dans le message *DHCPRELEASE*. Si le client utilise un *'identifiant client'* quand il obtient le bail il doit utiliser le même *'identifiant client'* dans le message *DHCPRELEASE*.

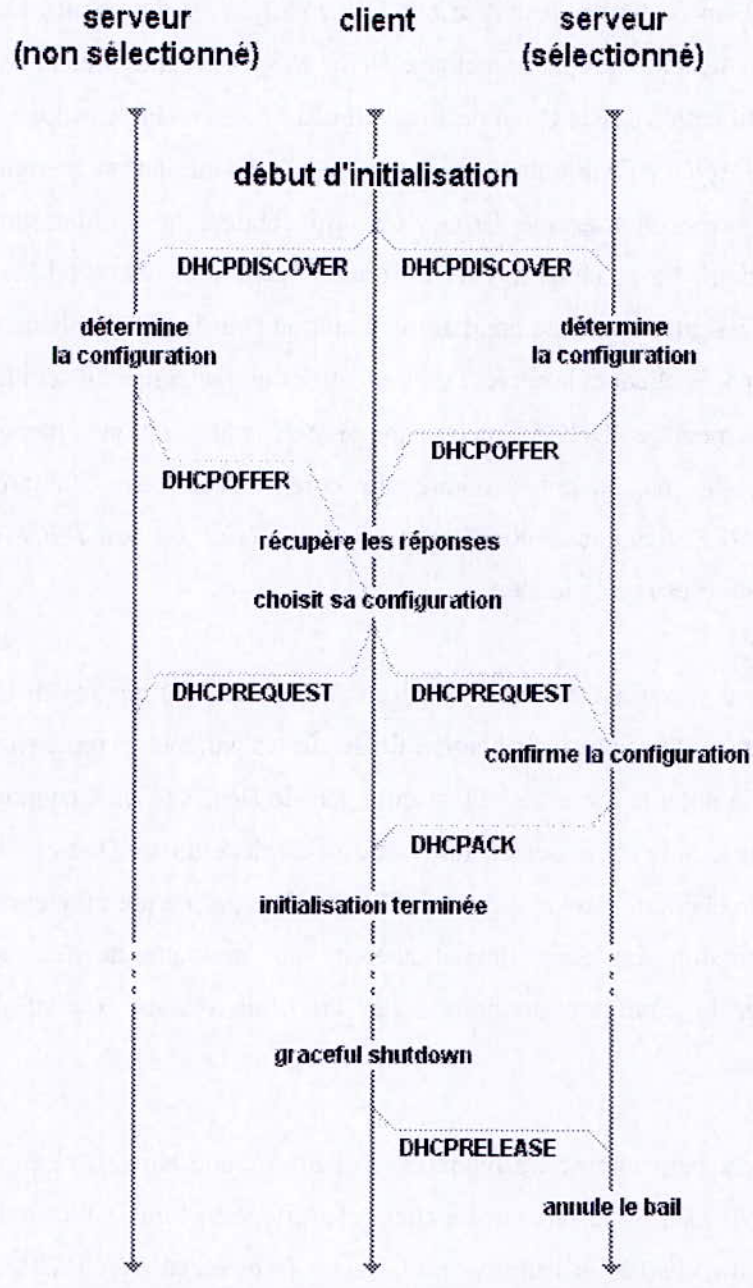


Figure 2.3 : Diagramme temporel des messages échangés entre les clients et les serveurs DHCP quand on alloue une nouvelle adresse

2.2.2.b Interaction client serveur - Réutilisation d'une adresse réseau anciennement attribuée

Si un client se souvient d'une adresse réseau qui lui a été attribuée par le passé et souhaite la réutiliser, il peut choisir d'omettre les étapes décrites précédemment. Le diagramme temporel

de la figure 2.4 montre la relation temporelle dans une interaction classique client-serveur pour un client réutilisant une adresse réseau qui lui avait déjà été attribuée dans le passé :

1. Le client diffuse un message *DHCPREQUEST* sur son sous réseau local. Le message inclut l'adresse réseau du client dans l'option *'adresse IP demandée'*. Comme le client n'a pas encore reçu son adresse réseau, il ne doit pas remplir le champ *'ciaddr'*. Les agents de relais BOOTP transmettent le message au serveur DHCP qui n'est pas sur le même sous réseau. Si le client utilise un *'identifiant client'* pour obtenir son adresse, le client doit utiliser le même *'identifiant client'* dans le message *DHCPREQUEST*.

2. Les serveurs qui connaissent les paramètres de configuration du client répondent avec un *DHCPACK* au client. Les serveurs ne devraient pas vérifier que l'adresse réseau du client est déjà utilisée; Pour des raisons d'optimisation des ressources réseau, les adresses IP sont délivrées pour une durée limitée. Si pour une raison ou une autre la requête du client est invalide, le serveur devrait répondre pas un message *DHCPNAK*.

3. Le client reçoit le message *DHCPACK* avec les paramètres de configuration. Le client effectue une vérification finale sur les paramètres (ex : en envoyant des requêtes ARP), et note la durée du bail spécifiée. Si le client détecte que l'adresse IP dans le message *DHCPACK* est déjà utilisée, le client doit envoyer un message *DHCPDECLINE* au serveur et relancer le processus de configuration en faisant la requête d'une nouvelle adresse réseau (*DHCPDISCOVER*). De même, si le client reçoit le message *DHCPNAK*, il doit aussi relancer un processus de configuration décrit dans la section précédente. Si le client ne reçoit ni un *DHCPAK* ni un *DAHCNAK*, il doit retransmettre un *DHCPREQUEST* en accord avec un algorithme de retransmission qui lui est propre. En effet, Le client devrait choisir de retransmettre le *DHCPREQUEST* un nombre suffisant de fois pour espérer contacter le serveur sans faire en sorte qu'il (le client, et par extension, l'utilisateur du client) attende trop longtemps avant d'abandonner, par exemple : un client

pourrai retransmettre le *DHCPREQUEST* 4 fois sur un totale de 60 secondes, avant de relancer la procédure d'initialisation. Bien entendu, si le client ne reçoit aucun message du serveur, il pourra choisir d'utiliser l'adresse réseau allouée précédemment avec les mêmes paramètres réseaux et le même bail

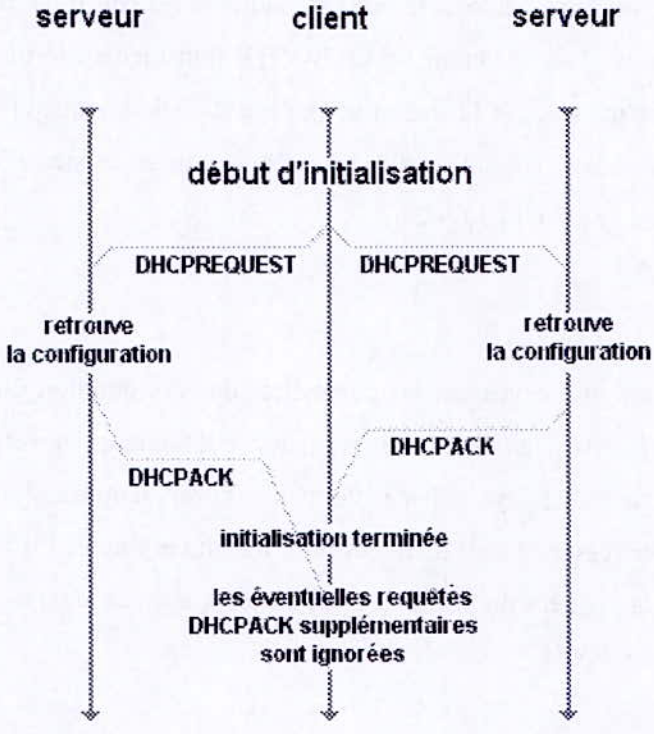


Figure 2.4: Diagramme temporel des messages échangés entre le client DHCP et les serveurs quand il réutilise une adresse ayant été précédemment attribuée.

4. Le client peut choisir de renoncer à son bail sur une adresse réseau en envoyant un message *DHCPRELEASE* au serveur. Le client identifie le bail dont il veut se défaire avec l'identifiant client ou 'chaddr' et l'adresse réseau dans le message *DHCPREALEASE*.

2.2.2.c Interprétation et représentation des valeur de temps

Un client acquiert un bail pour une adresse réseau pour une période fixe (qui peut être infinie). Dans ce protocole, l'unité de temps est la seconde. La valeur temps 0xffffffff est réservée pour représenter l'infini. Comme un client et un serveur n'ont pas forcément une horloge synchronisée, et pour être interprété correctement par les horloges des clients locaux, le temps est représenté dans les messages par des données relatives. Le temps relatif, représenté en

secondes dans un mot de 32 bits non signé, donne une plage de temps relatif de 0 à environ 100 ans, ce qui est suffisant pour les durées mesurées dans DHCP. Le serveur précise le bail au client en envoyant dans le champ options l'option 51 (*bail offert*), le client peut aussi envoyer cette option au serveur, mais dans ce cas, le serveur interprétera cette option comme étant une demande de bail.

2.3 Spécification du protocole client-serveur

Dans cette section on s'appuie sur le fait que le serveur DHCP possède un bloc d'adresses réseau à partir desquelles il peut satisfaire les requêtes d'adresse. Chaque serveur maintient également, dans un espace de stockage local, une base de données des adresses allouées et des baux.

Cette section représente à nos yeux, la partie la plus important car elle montre les différences de base entre les différents messages DHCP du point de vu de leur structure. Nous nous sommes appuyé sur cette section pour définir notre model du serveur DHCP décrit dans le chapitre suivant.

2.3.1 Construction et envoie des message DHCP

Les clients et serveurs DHCP construisent tous deux des messages DHCP en remplissant les champs de la partie fixe des messages et ajoutant des données marquées dans la zone option de longueur variable. La zone des options inclut d'abord un *'cookie magique'* de 4 octets (qui à été décrit dans la section 2.2.2) suivi par les options. La dernière option doit toujours être *'end'*.

DHCP utilise le protocole de transport UDP. Les messages DHCP d'un client vers un serveur sont envoyés au port (67) du serveur DHCP et les messages d'un serveur vers un client sont envoyé au port (68) du client DHCP. Un serveur avec de multiples adresses réseau (par ex : une machine multi-sites) peut utiliser n'importe laquelle de ses adresses dans un message DHCP sortant.

2.3.2 Comportement du serveur DHCP

Un serveur DHCP traite les messages DHCP provenant d'un client selon l'état courant de l'affectation pour ce client. Un serveur DHCP peut recevoir les messages suivants d'un client :

- *DHCPDISCOVER*
- *DHCPREQUEST*
- *DHCPDECLINE*
- *DHCPRELEASE*
- *DHCPINFORM*

Le tableau 2.3 donne l'utilisation faite par un serveur des champs et des options dans un message DHCP. Le reste de cette section décrit l'action du serveur DHCP pour chaque message entrant.

Tableau 2.3: Champs et options utilisés par les serveurs DHCP

Champ	**DHCPOFFER**	**DHCPACK**	**DHCPNAK**
'op'	BOOTREPLY	BOOTREPLY	BOOTREPLY
'htype'	(Du RFC 1700 "Assigned Numbers")		
'hlen'	(longueur de l'Adresse matérielle en octets)		
'hops'	0	0	0
'xid'	'xid' du message DHCPDISCOVER client	'xid' du message DHCPREQUEST client	'xid' du message DHCPREQUEST client
'secs'	0	0	0
'ciaddr'	0	'ciaddr' du DHCPREQUEST ou 0	0
'yiaddr'	adresse IP offerte au client	adresse IP assignée au client	0
'siaddr'	adresse IP du serveur d'amorçage suivant	adresse IP du serveur d'amorçage suivant	0
'flags'	drapeaux du message client DHCPDISCOVER	drapeaux du message client DHCPREQUEST	drapeaux du message client DHCPREQUEST
'giaddr'	'giaddr' du message client DHCPDISCOVER	'giaddr' du message client DHCPREQUEST	'giaddr' du message client DHCPREQUEST
'chaddr'	'chaddr' du message client DHCPDISCOVER	'chaddr' du message client DHCPREQUEST	'chaddr' du message client DHCPREQUEST
'sname'	nom d'hôte serveur ou options	nom d'hôte serveur ou options	(inutilisé)

'fichier'	nom du fichier démarrage client ou options	nom du fichier démarrage client ou options	(inutilisé)
'options'	Options	options	

Option	**DHCPOFFER**	**DHCPACK**	**DHCPNAK**
adresse IP demandée	NE DOIT PAS	NE DOIT PAS	NE DOIT PAS
Durée du bail de l'adresse IP	DOIT	DOIT (DHCPREQUEST) NE DOIT PAS (DHCPINFORM)	NE DOIT PAS
Utilisation des champs *'fichier'/'sname'*	PEUT	PEUT	NE DOIT PAS
Type du message DHCP	DHCPOFFER	DHCPACK	DHCPNAK
Liste des Paramètres requis	NE DOIT PAS	NE DOIT PAS	NE DOIT PAS
Message	DEVRAIT	DEVRAIT	DEVRAIT
Identifiant client	NE DOIT PAS	NE DOIT PAS	PEUT
Identifiant de classe fournisseur	PEUT	PEUT	PEUT
Identifiant serveur	DOIT	DOIT	DOIT
Taille maximum des messages	NE DOIT PAS	NE DOIT PAS	NE DOIT PAS
Tout le reste	PEUT	PEUT	NE DOIT PAS

1. le message *DHCPDISCOVER*

Quand un serveur reçoit un message *DHCPDISCOVER* d'un client, le serveur choisit une adresse réseau pour le client demandeur. Si aucune adresse n'est disponible, le serveur peut choisir de rapporter le problème à l'administrateur système. Si une adresse est disponible, le serveur peut donner l'adresse IP que le client a demandé dans l'option *'adresse IP demandé'* (code 50), si celle si est valide, ou bien il peut donner au client un nouvelle adresse IP. De même, pour le bail, le serveur peut satisfaire la demande du client concernant le temps d'attribution de l'adresse IP ou bien, il peut lui imposé un bail définit selon la politique de gestion du réseau.

2. le message *DHCPREQUEST*

Un message *DHCPREQUEST* peut venir d'un client répondant à un message *DHCPOFFER* du serveur (*DHCPREQUEST* généré durant l'état *SELECTION*), d'un client vérifiant une adresse IP précédemment allouée (*DHCPREQUEST* généré pendant l'état *INIT-REBOOT*) ou d'un client qui rallonge le bail sur une adresse réseau (*DHCPREQUEST* généré pendant l'état *RENOUVELLEMENT* ou *REAFFECTATION*). Si le message *DHCPREQUEST* contient une option *'identifiant serveur'*, le message est la réponse à un *DHCPOFFER*. Sinon, le message est une requête pour vérifier ou étendre un bail existant. Si le client utilise un *'identifiant client'* dans un message *DHCPREQUEST*, il doit utiliser le même *'identifiant client'* dans tous les messages suivants. Si le client inclut une liste de paramètres requis dans un message *DHCPDISCOVER* il doit inclure cette liste dans tous les messages suivants. Notons que Le *DHCPREQUEST* d'un client en *REAFFECTATION* est destiné à satisfaire les sites qui possèdent plusieurs serveurs DHCP et un mécanisme pour maintenir une certaine cohésion entre les différents baux contrôlés par les divers serveurs. Un serveur DHCP peut étendre le bail d'un client seulement s'il dispose de l'autorisation locale de le faire.

Le tableau 2.4 détaille les différences entre les messages des clients selon les différents états :

Tableau 2.4: Les messages clients selon les différents états.

	INIT-REBOOT	**SELECTION**	**RENOUVELLEMENT**	**REAFFECTATION**
diff./adressé	diffusé	diffusé	adressé	Diffusé
IP serveur	DOIT PAS	DOIT	DOIT PAS	DOIT PAS
IP demandée	DOIT	DOIT	DOIT PAS	DOIT PAS
ciaddr	zéro	Zéro	adresse IP	adresse IP

3. le message *DHCPDELINE*

Si un serveur reçoit un message *DHCPDECLINE*, le client a découvert par d'autres moyens que l'adresse réseau suggérée est déjà utilisée. Le serveur doit marquer l'adresse réseau comme étant indisponible et devrait informer l'administrateur du réseau local d'un éventuel problème de configuration.

4. Le message *DHCPRELEASE*

Quand il reçoit un message *DHCPRELEASE*, le serveur marque l'adresse réseau comme non allouée. Le serveur devrait avoir un enregistrement des paramètres d'initialisation du client pour une réutilisation ultérieure de l'adresse par ce même client.

5. Le message *DHCPINFORM*

Le serveur répond à un message *DHCPINFORM* en envoyant un message *DHCPACK* directement à l'adresse donnée dans le champ *'ciaddr'* du message *DHCPINFORM*. Le serveur ne doit pas envoyer un temps d'expiration du bail au client et ne devrait pas remplir le champ *'yiaddr'*.

2.3.3 Les comportement des clients DHCP

Un client peut recevoir les messages suivants d'un serveur :

- *DHCPOFFER*
- *DHCPACK*
- *DHCPNAK*

Le tableau 2.5 donne l'utilisation, par un client, des champs et les options dans un message DHCP.

Tableau 2.5: Champs et options utilisés par les clients DHCP

Champ	**DHCPDISCOVER DHCPINFORM**	**DHCPREQUEST**	**DHCPDECLINE, DHCPRELEASE**
'op'	BOOTREQUEST	BOOTREQUEST	BOOTREQUEST
'htype'	(Du RFC 1700 "Assigned Numbers")		
'hlen'	(longueur de l'Adresse matérielle en octets)		
'hops'	0	0	0
'xid'	choisi par le client	'xid' du message DHCPOFFER serveur	choisi par le client
'secs'	0 ou nb secondes depuis départ processus DHCP	0 ou nb secondes depuis départ processus DHCP	0

'flags'	drapeau 'BROADCAST' levé si le client demande une réponse diffusée	drapeau 'BROADCAST' levé si le client demande une réponse diffusée	0
'ciaddr'	0 (DHCPDISCOVER) adresse réseau du client(DHCPINFORM)	0 ou adresse réseau du client(BOUND/RENEW/REBIND)	0 (DHCPDECLINE) adresse réseau du client(DHCPRELEASE)
'yiaddr'	0	0	0
'siaddr'	0	0	0
'giaddr'	0	0	0
'chaddr'	adresse matérielle du client	adresse matérielle du client	adresse matérielle du client
'sname'	options, si indiqué dans option 'sname/fichier' sinon inutilisé	options, si indiqué dans option 'sname/fichier' sinon inutilisé	(inutilisé)
'fichier'	options, si indiqué dans option 'sname/fichier' sinon inutilisé	options, si indiqué dans option 'sname/fichier' sinon inutilisé	(inutilisé)
'options'	Options	options	(inutilisé)

Option	**DHCPDISCOVER DHCPINFORM**	**DHCPREQUEST**	**DHCPDECLINE, DHCPRELEASE**
adresse IP demandée	PEUT (DISCOVER) NE DOIT PAS (INFORM)	DOIT (dans SELECTION ou INIT-REBOOT) NE DOIT PAS (dans AFFECTATION ou RENOUVELLEMENT)	DOIT (DHCPDECLINE) NE DOIT PAS (DHCPRELEASE)
Durée du bail IP	PEUT (DISCOVER) NE DOIT PAS (INFORM)	PEUT	NE DOIT PAS
Utilisation des champs 'fichier'/'sname'	PEUT DHCPDISCOVER/ DHCPINFORM	PEUT DHCPREQUEST	PEUT DHCPDECLINE/ DHCPRELEASE
Identifiant client	PEUT	PEUT	PEUT

Identifiant classe fournisseur	PEUT	PEUT	NE DOIT PAS
Identifiant serveur	NE DOIT PAS	DOIT (après SELECTION) NE DOIT PAS (après INIT-REBOOT, AFFECTÉ, RENOUVELLEMENT ou REAFFECTATION)	DOIT
Liste des Paramètres requis	PEUT	PEUT	NE DOIT PAS
Taille maximum des messages	PEUT	PEUT	NE DOIT PAS
Message	NE DEVRAIT PAS	NE DEVRAIT PAS	DEVRAIT
Site spécifique	PEUT	PEUT	NE DOIT PAS
Autres	PEUT	PEUT	NE DOIT PAS

Ainsi donc, le modèle du serveur DHCP que nous allons proposer dans le chapitre suivant, traite que les champs ayants des valeurs différentes pour chaque message du client. Nous exposerons ces différences une peu plus tard.

2.4 Cas de gestion de plusieurs sous-réseaux et fonctionnement des agents relais

Un agent relais les messages DHCP/BOOTP diffusés sur l'une des interfaces physiques qui lui sont connecté, telle une carte réseau, vers d'autres sous réseaux à distance auxquels il est connecté par d'autres interfaces physiques. La figure suivante montre comment le Client C du Sous-réseau 2 obtient un bail d'adresse DHCP auprès du Serveur 1 du Sous-réseau 1.

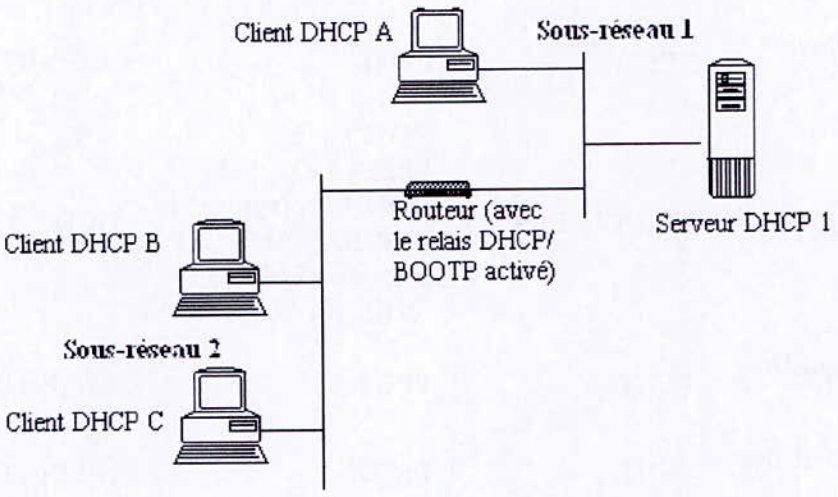


Figure 2.5 : Exemple de réseau avec agents relais

1. Le Client C DHCP diffuse un message de découverte DHCP (*DHCPDISCOVER*) sur le Sous-réseau 2, en tant que datagramme UDP (User Datagram Protocol) qui utilise le port de serveur UDP connu 67 (numéro de port réservé et partagé pour les communications des serveurs BOOTP et DHCP).
2. L'agent relais, ici un routeur activé en tant que relais DHCP/BOOTP, examine le champ d'adresse IP de la passerelle ('*giaddr*'), qui figure dans l'en-tête du message DHCP/BOOTP. Si l'adresse IP de ce champ est 0.0.0.0, l'agent place dans ce champ son adresse IP et transmet le message au Sous-réseau 1 où se trouve le serveur DHCP.
3. Lorsque le Serveur DHCP 1 du Sous-réseau 1 reçoit le message, il examine le champ d'adresse IP de la passerelle pour offrir la bonne adresse.
4. Si le Serveur DHCP 1 dispose de plusieurs étendues DHCP, l'adresse qui figure dans le champ d'adresse IP de la passerelle '*giaddre*' identifie l'étendue DHCP d'où il peut proposer un bail d'adresse. Par exemple, si le champ d'adresse IP de la passerelle '*giaddre*' contient l'adresse IP 10.0.0.2, le serveur DHCP vérifie dans ses étendues d'adresses disponibles s'il trouve une plage d'adresses d'étendue qui correspond au réseau IP de classe A contenant l'adresse de passerelle en tant qu'hôte. Dans ce cas, le serveur DHCP vérifie s'il trouve des adresses d'étendue situées entre 10.0.0.1 et 10.0.0.254. Lorsque le serveur DHCP trouve une étendue qui correspond, il sélectionne une adresse disponible dans cette étendue et l'utilise dans une réponse d'offre de bail d'adresse IP au client.

5. Lorsque le Serveur DHCP 1 reçoit le message *DHCPDISCOVER*, il traite et envoie une offre de bail d'adresse IP (*DHCPOFFER*) directement à l'agent relais identifié dans le champ d'adresse IP de la passerelle '*giaddr*'.
6. le routeur relaie ensuite l'offre de bail d'adresse (*DHCPOFFER*) jusqu'au client DHCP. L'adresse IP du client est toujours inconnue et doit donc être diffusée sur le sous-réseau local. De la même façon, un message de demande (*DHCPREQUEST*) est relayé du client au serveur, et un message d'accusé de réception (*DHCPACK*) est relayé du serveur au client.

2.5 Mise en œuvre d'un serveur DHCP

Dans cette section nous allons installer un serveur DHCP, en utilisant l'environnement LINUX et plus précisément la distribution SUSE 9.1. Par la suite, avec l'aide d'un scanner de réseaux nous analyserons les messages DHCP entre clients et serveur DHCP.

2.5.1 Installation du serveur DHCP

Sur SUSE, ça se fait très simplement en installant les paquetage dhcp-common et dhcp-server. Dans la version 9.1 de SUSE, nous disposons de la version 3.0 du serveur. Il y a un seul fichier de configuration : /etc/dhcp.conf, qu'on peut configurer avec un éditeur de texte, ou à travers une interface graphique (exemple : Webmin).

2.5.2 Configuration du serveur DHCP

Comme nous l'avons vu plut haut, un serveur DHCP, en plus de fournir la configuration IP de base (Adresse et masque), peut aussi transmettre un nombre plus ou moins grand de paramètres supplémentaires (adresse d'un DNS, adresse de la passerelle par défaut, ...).

Notre configuration de base est la suivante :

- Un seul réseau, avec des IP choisies dans la classe B privée 172.16.0.0, donc avec un masque 255.255.0.0 ;
- Une passerelle par défaut unique pour tous nos hôtes du réseau, dans l'exemple, ça sera 172.16.0.1 ;
- Enfin, nous disposons d'un DNS, également unique pour tout les hôtes du réseau et ayant l'adresse 172.16.0.12 ;

- Sur la totalité de la classe B disponible, nous avons réserver les adresse comprises entre 172.16.0.100 et 172.16.0.250 pour les client du réseau. Cette plage constitue la réserve d'adresse que le DHCP pourra fournir aux client ;
- La duré du bail sera limité pour tout les clients a 1 heure (soit 3600 secondes) .

2.5.3 Analyse des trames

Un «**sniffer**» (appelé *analyseur réseau* en français) est un dispositif permettant d'"écouter" le trafic d'un réseau, c'est-à-dire de capturer les informations qui y circulent. C'est un outil d'administrateur, qui est capable du meilleur comme du pire. Il en existe un tout à fait convenable et gratuit, aussi bien en version Linux que Windows, c'est ***Ethereal*** [4]. Nous l'avons installée dans un des clients DHCP. La figure suivante représente, l'interface graphique du scanner, la partie surlignée en verre indique le filtre du scanner, cela pour afficher que les messages en relation avec le processus DHCP (messages DHCP et requêtes ARP).

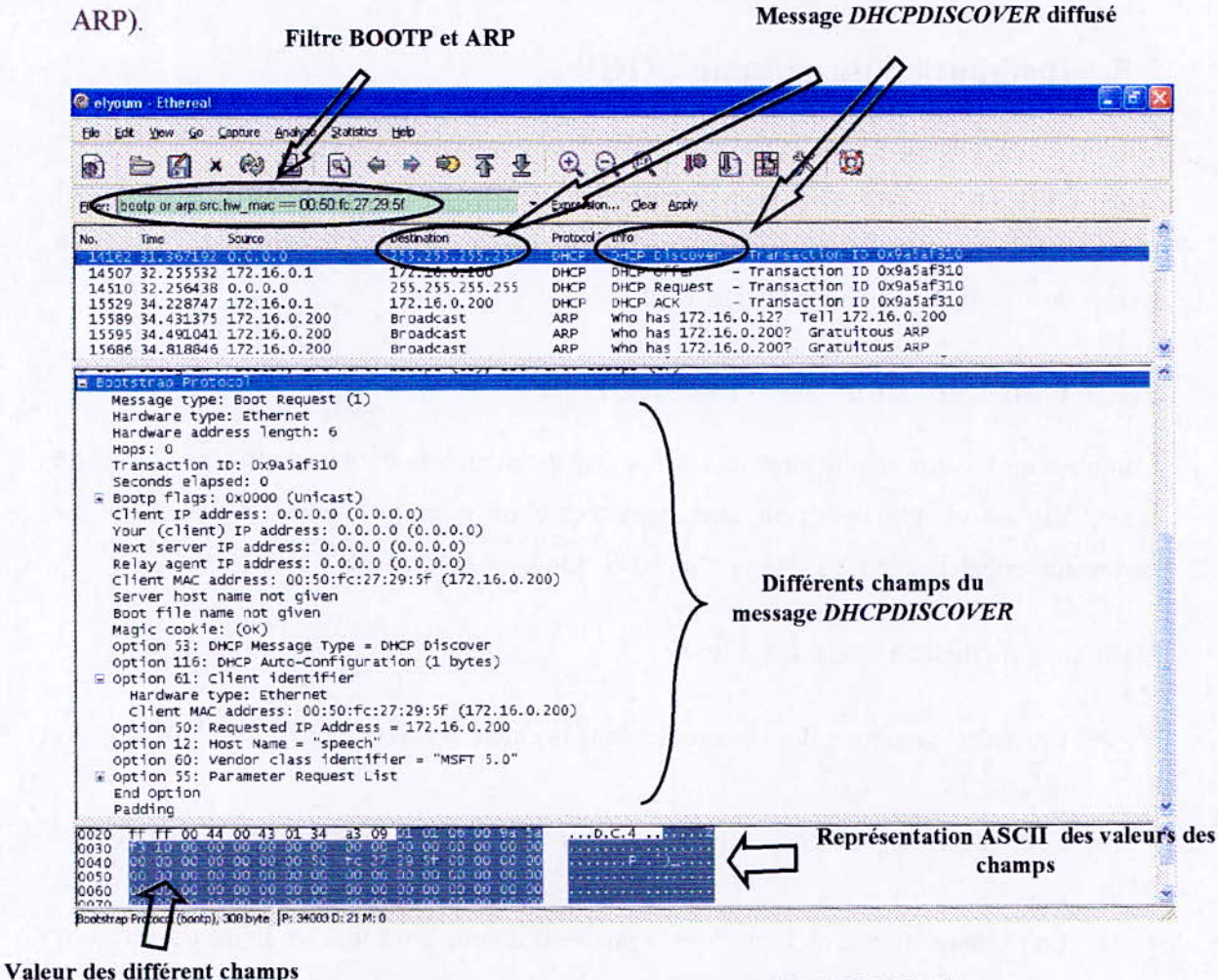


Figure 2.6 : Ethereal en œuvre, trame *DHCPDISCOVER*

On peut partager la figure du dessus, en trois parties :

1. La premier partie (celle du haut), représente l'en-tête des trames, en peut lire et cela par ordre chronologique le message *DHCPDISCOVER* pour la demande d'une adresse IP, *DHCPOFFER* qui est l'offre du serveur DHCP, *DHCPREQUEST* pour accepter l'offre du serveur (état *SELECTION*), *DHCPACK* pour 'dire' au client que la configuration qui veut avoir est correcte. En fin, le client envoie des requêtes ARP, pour voir si les paramètres réseau offerts par le serveur sont corrects ou non.

2. La deuxième partie (celle du milieu), nous montre ce que chaque champs du message *DHCPDICOVER* contient comme information, ainsi on peut voir que, par exemple, le champs '*op*' (code oprération) est positionné sur un *BOOTREQUEST*, que le champs '*flags*' est mis a zéro, ce qui veut dire que les message a bien était diffuser, on peut aussi remarquer les différents options émises par le client, a savoir, l'option '*type de message DHCP*' (code 53), l'option '*identifiant client*' (code 61), qui est l'adresse MAC (matérielle) du client, l'option 50 (*adresse IP demandé*) qui est l'adresse que souhaiterai avoir le client.

3. La troisième partie (celle du bas) représente une chaîne de caractère hexadécimaux , qui forme le message DHCP, on peut par exemple remarquer que cette chaîne de caractères commence par la valeur 01 qui représente la valeur du champs '*op*' (qui est bien de taille égale à 1 octet) et que l'octet suivant (01) correspond au champs '*htype*' et qui vaut 1 en décimale pour dire que l'adresse matérielle est du type Ethernet, ainsi de suite on peut connaître la valeur hexadécimale de chaque champs y compris les champs d'options. Notons, que ces chaîne de caractères sont de grandes importances, car c'est ce que nous avons mis a l'entrée de notre modèle du serveur DHCP, lors de nos simulations.

La figue suivante est la même qui la précédente sauf que, dans ce cas il s'agit du message *DHCPOFFER* émis par le serveur. On peut remarquer que notre serveur offre bien un bail de 1 heure (option 53), représenter par la chaîne d'octet suivante 33 04 00 00 0E 10, ou le premier octet correspond au code de l'option (53 en décimale), l'octet suivant (4 en

décimale) veut dire que le bail (en secondes) est codé sur 4 octets, enfin 00 00 0E 10 est la valeur en hexadécimal du bail (3600 secondes).

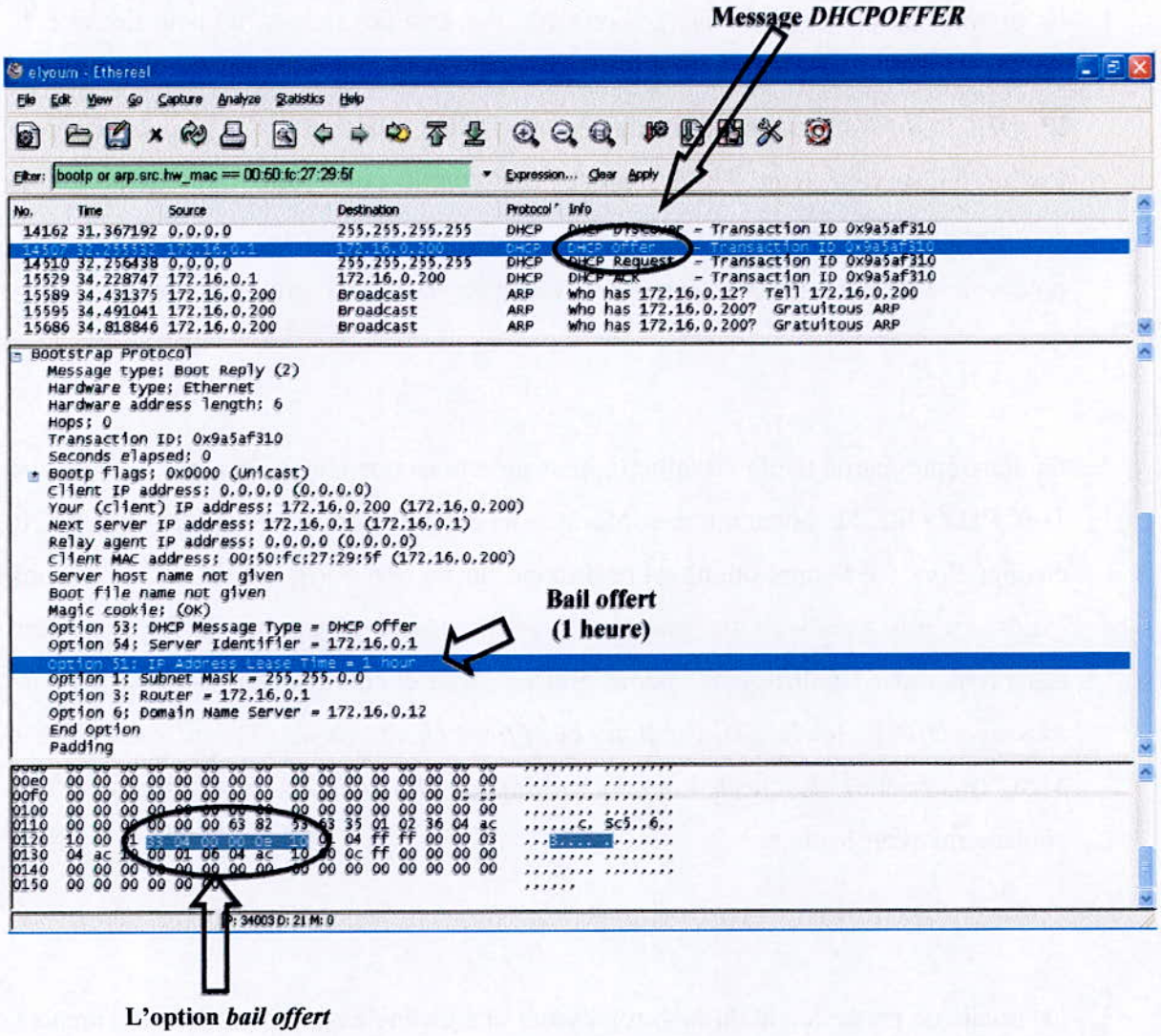


Figure 2.7 : Ethereal en œuvre, trame *DHCPOFFER*

2.6Conclusion

Malgré la simplicité apparente de la tâche d'un serveur DHCP, à savoir attribuer des adresses IP et des paramètres réseaux aux machines clientes, le protocole DHCP que nous avons présenté dans ce chapitre, indique que le serveur DHCP devra en plus attribuer une configuration réseau, garder une certaine cohésion à cette procédure, pour garantir une bonne utilisation du réseau tout en tenant compte de sa sécurité.

3 Développement

3.1 Introduction

Dans ce chapitre nous exposerons la méthodologie de développement suivie pour aboutir à la réalisation du service implémenté sur carte. De prime abord nous expliquerons l'approche globale du Soc, ensuite nous développerons pourquoi il serait préférable d'élever le niveau d'abstraction pour décrire notre système. Et enfin pourquoi utiliser le langage formel SDL pour modéliser le protocole DHCP.

3.2 Approche de conception

Trois approches sont envisageables pour le passage des spécifications abstraites vers l'implémentation sur circuit:

1) A l'aide d'un microprocesseur.
2) Passant par un microprocesseur intégré sur un FPGA .
3) A l'aide de langages de description matériel (HDL) tels que VHDL, Verilog ou autres.

Comme le système que nous avons à décrire se trouve être assez complexe pour être modélisé à l'aide de langages de description matériel, nous avons décidé d'élever le niveau d'abstraction, en utilisant un langage de spécification système et de surcroît formel. Mais les concepts que manipule ce genre de langages ne sont pas aisément transposables sur ceux des langages de description de matériels, et donc un moyen de passage direct de ces spécifications vers un code VHDL par exemple, n'est pas encore possible (du moins à notre connaissance). De ce fait, nous avons choisi d'utiliser la deuxième approche pour la synthèse des spécifications système, autrement dit le passage par un processeur intégré utilisant le code C généré à partir des spécifications.

Les différentes étapes de la conception du Soc une fois la modélisation terminée et le code C généré sont les suivantes :

1- La création du système matériel (hardware) embarqué c'est-à-dire la plate-forme matérielle consistant en un ou plusieurs processeurs, bus et périphériques connectés aux processeurs. Avec la possibilité de créer et d'importer tous les IP (blocs fonctionnels réutilisables), désirés et catalogués.
2- La deuxième étape consiste en la création du système embarqué logiciel (software), c'est-à-dire une application logicielle à exécuter sur le processeur, impliquant l'ajout des librairies, la compilation des applications décrites en langage C.
3- Après avoir décrit le système processeur et écrit les applications logicielles, on peut charger le système sur la carte FPGA grâce aux outils d'implémentation, qui permettent la génération Des HDL Netlist pour la description des composants et leurs connections, et Enfin la génération du fichier de configuration Bitstream.

La figure 3.1 qui suit détaille ces étapes :

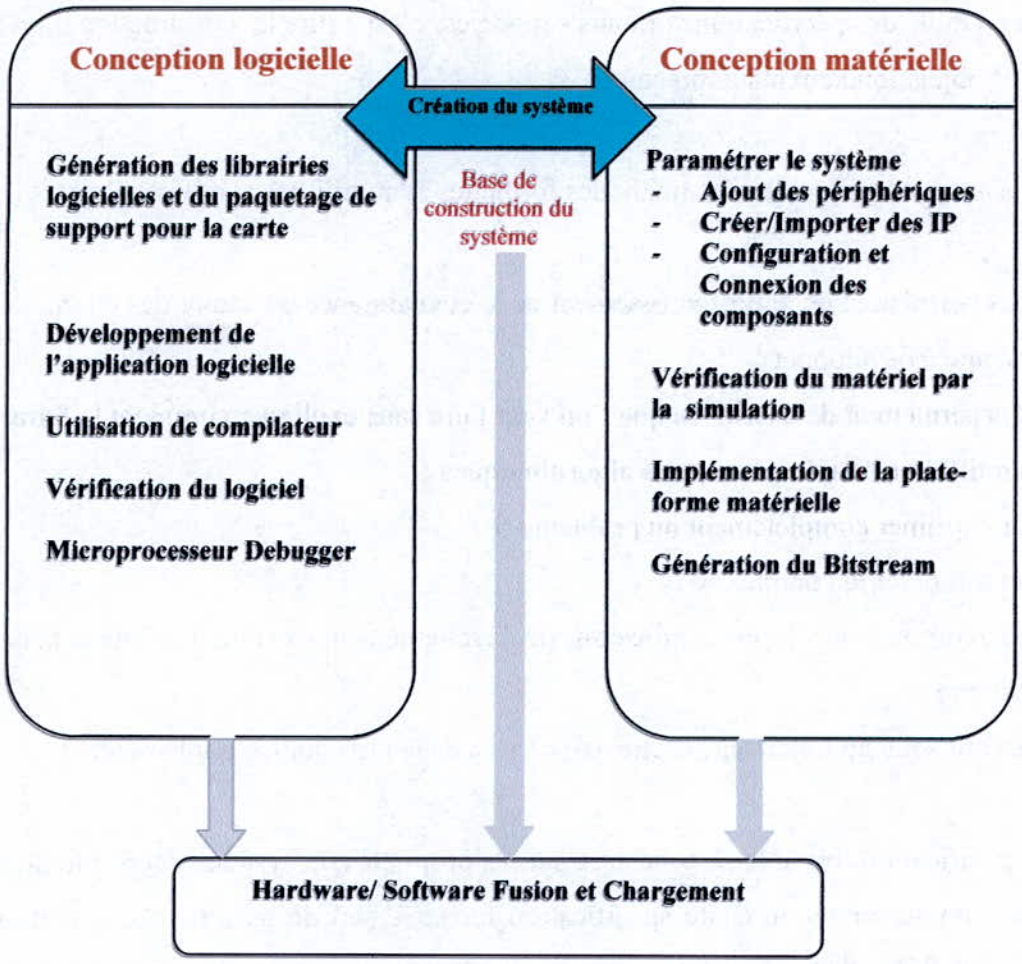


Figure3.1 : Flot de conception Soc

3.3 Nécessité d'une modélisation formelle

Une description formelle est une description claire et parfaitement précise. Elle permet d'obtenir des spécifications système prouvées et des programmes prouvés conformes aux exigences fonctionnels.

Les langages formels reposent sur des fondements mathématiques qui permettent d'effectuer des vérifications formelles sur les systèmes décrits. Différentes approches ont été développées, parmi elles, les automates d'états finis et les types de données abstraites [5].

Il existe deux classes de méthodes formelles :

La première traite de spécifications orientées propriétés, qui sont des descriptions données dans un langage permettant d'énoncer les propriétés attendues d'un système.

Et la seconde de spécifications orientées modèles, c'est à dire la construction d'un système à partir d'objets fondamentaux préétablis.

Les raisons pour lesquelles les méthodes formelles sont utilisées sont les suivantes :

- Elles permettent un approfondissement de la connaissance du cahier des charges du système à développer ;
- Elles permettent de décrire ce que l'on veut faire sans expliquer comment le faire ;
- Pour différer l'étude des aspects algorithmiques ;
- Pour exprimer complètement un problème ;
- Pour diminuer la complexité ;
- Pour avoir un contrôle sur le processus de développement (qualité, fiabilité et la découverte d'erreurs) ;
- Elles ont souvent l'avantage d'être associées à des outils qui les implémentent.

Une spécification formelle se concentre sur les propriétés du système décrit plutôt que sur les détails d'implémentation. Cette spécification formelle sert de base pour une réalisation, elle doit donc faire abstraction des détails d'implémentation afin de retarder les décisions concernant la réalisation et de permettre de n'exclure aucune implémentation valide.

Nous pouvons citer comme exemples de langages de description formelle : Le langage Z, le langage B, ESTEREL, LUSTRE, SDL (Specification and Description Language), UML (Unified Modeling Language), LOTOS (Language Of Temporal Ordering Specification), ESTELLE…etc

3.4 Cas de SDL

Un langage spécifique est choisi en fonction du domaine d'application, de son pouvoir d'expression, et de la disponibilité de méthodes et d'outils autour de ce langage.

Dans notre cas, les langages les mieux adaptés sont ceux dédiés à la spécification de protocoles de communication et aux systèmes distribués, c'est-à-dire : LOTOS, ESTELLE

SDL [5] et UML. Aussi, SDL et UML présentent un avantage sur LOTOS et ESTELLE, celui la richesse d'expression ainsi que la facilité de compréhension et d'apprentissage.

L'UML se situe en réalité, à un niveau d'abstraction plus élevé que SDL et ses principes de modélisation sont applicables à n'importe quel concept. En contrepartie l'UML n'est pas assez précis pour décrire de manière détaillée les applications. De ce fait, on l'utilise souvent dans les phases en amont d'un développement pour les opérations d'analyse et de spécifications système, mais rarement durant les opérations de spécifications détaillées et de conception. Et par conséquent, un autre langage est rapidement nécessaire pour enrichir le modèle décrit (avec du C ou SDL) [6].

Pour toutes ces raisons, SDL se présente comme étant le mieux adapté et le plus pratique, pour nos spécifications système et la modélisation du protocole DHCP.
Il présente également, plusieurs spécificités avantageuses telles que le fait que ce soit un langage normalisé, possédant une riche grammaire et sans ambiguïté. SDL permet des spécifications textuelles mais également graphiques, et ses diagrammes sont faciles à comprendre.
Une autre caractéristique d'SDL est qu'il est un langage Orienté Objet, supportant des encapsulations et des polymorphismes, étendant le concept de classe de données Orientées Objet pour des applications techniques et des objets actifs (exemples : systems, blocks... etc).

La précision et la formalité de ce langage permettent à des outils supportés par SDL, la compilation vers des langages de niveau inférieur comme C/C ++. Cela signifie que le système SDL peut être traduit en une application exécutable.

Donc, Pendant le processus de modélisation et après son achèvement et parce que SDL a une sémantique complète, la description SDL peut être rapidement vérifiée et mise au point en utilisant les outils puissants disponibles aujourd'hui que sont les compilateurs et les simulateurs. Ce qui permet la correction des erreurs à un niveau très avancé.

3.5 Présentation du langage SDL

Le langage SDL (Specification and Description Language)[7],[8], est dédié à la modélisation des systèmes temps réel, distribués et de télécommunication et est standardisé par l'ITU

(International Union of Telecommunication)[9]. Un système décrit en SDL est composé d'un ensemble de processus concurrents communicant à travers des signaux. SDL supporte différents concepts permettant la description de systèmes tels que la structure, le comportement et la communication. Il existe deux formats SDL, un graphique et l'autre textuel, bien que différente, ces deux manière de décrire un système SDL ont une même base sémantique commun.

3.5.1 Structure

Dans une description SDL, l'entité la plus haute de la hiérarchie est appelée système (*system*). La structure statique d'un système est décrite en utilisant une hiérarchie de blocs (*block*). Un bloc peut contenir d'autres blocs, résultant en une structure hiérarchique ou bien un ensemble de processus (*process*) dans le cas d'un bloc feuille.

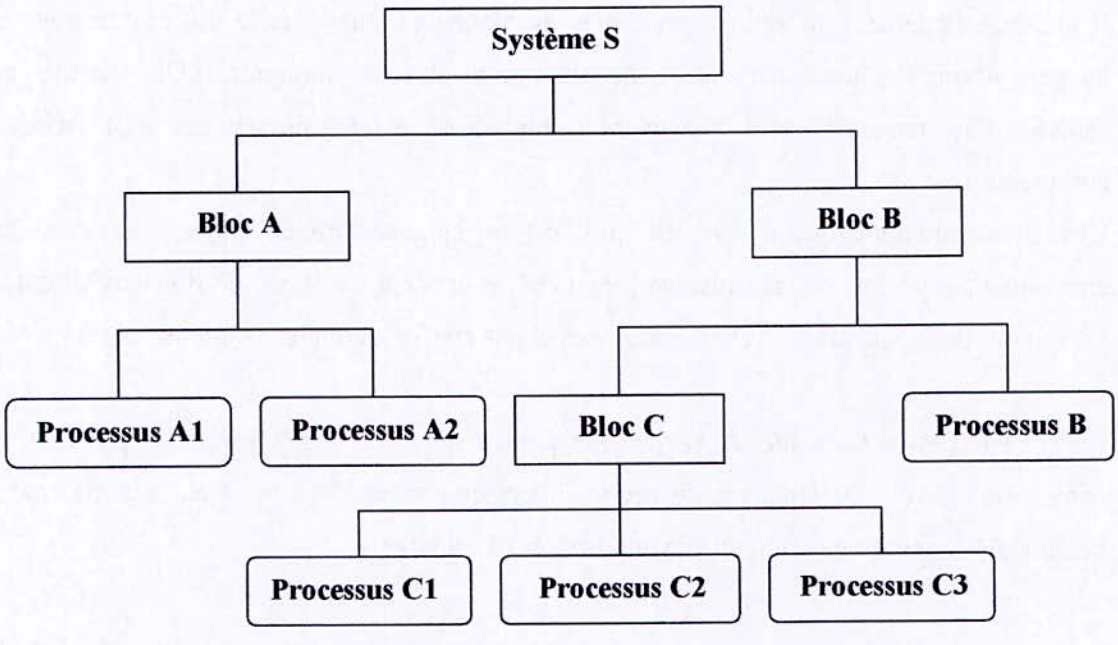


Figure 3.2 : Exemple de structure SDL

- Une instance de système contient un ensemble d'instances de blocs.
- Une instance de bloc peut contenir d'autres instances de blocs ou un ensemble d'instances de processus.
- Une instance de processus peut contenir un automate d'états finis ou un ensemble de services exécutés séquentiellement.
- Un service est un automate d'états finis.

- Une procédure est un automate d'états finis.

Les interfaces de communication des différents processus sont interconnectées entre elles à l'aide de canaux de communication (figure 3.2). Les différents processus d'un même bloc sont connectés entre eux et jusqu'à la frontière du bloc par des routes. Les blocs sont connectés entre eux par des canaux. Deux instances dans différentes parties d'un système sont connectées entre elles à travers les interfaces des blocs qui les englobent : deux processus appartenant à des blocs différents sont reliés à leur blocs respectifs par des routes et les blocs sont reliés entre eux par des canaux. Les canaux et les routes sont des vecteurs de signaux utilisés par les processus pour communiquer. Les signaux échangés par les processus suivent un chemin composé de routes et de canaux du processus émetteur au processus receveur. Sur le système représenté en figure 3.3.2, le processus *A1* communique avec les frontières du bloc *A* à travers la route *chenvA* . Cette route est connectée au canal chEnvA. Sur la figure 3.3.1, les deux blocs *A* et *B*, communiquent à travers le canal *chAB* et à la frontière à travers *chEnvA* et *chEncB*.

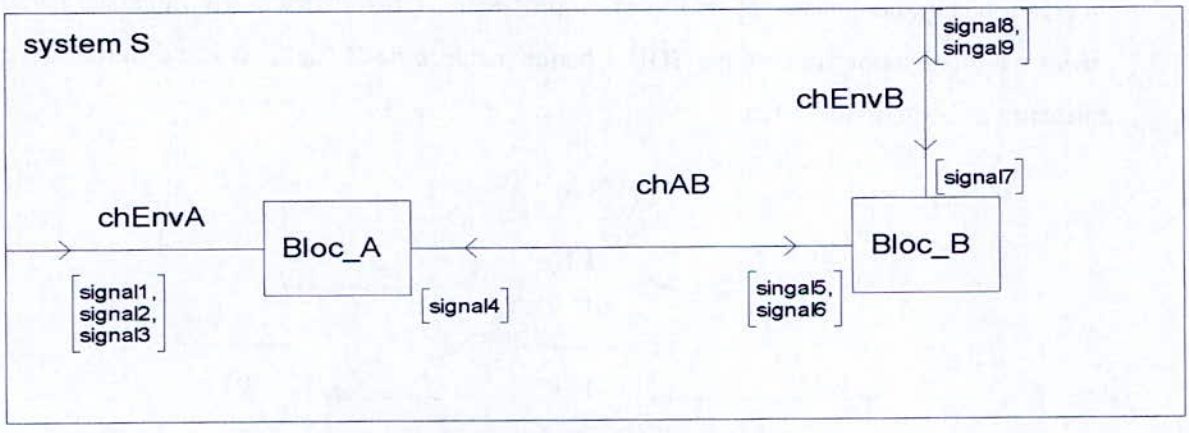


Figure 3.3.1 : Exemple d'un Système SDL

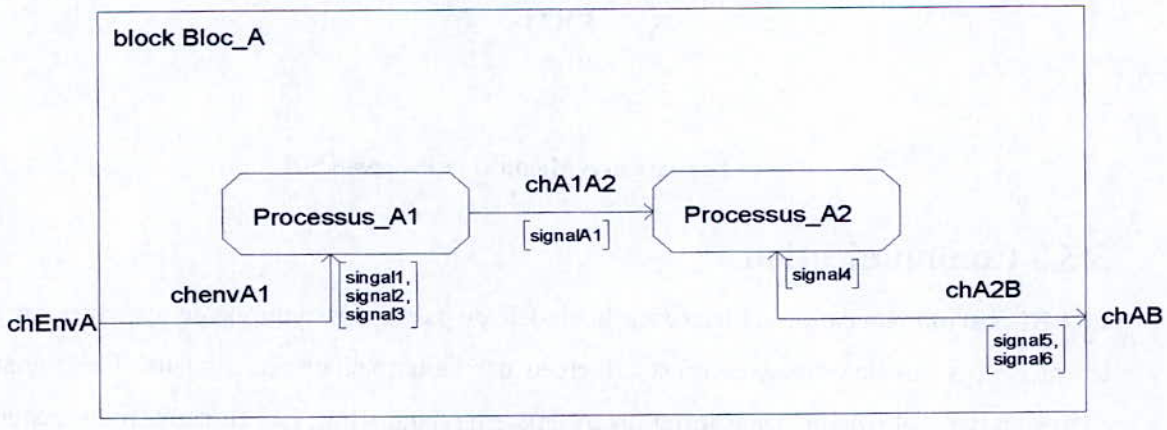


Figure 3.3.2 : Exemple d'un bloc SDL

Figure 3.3 : Exemple d'une Description SDL graphique

3.5.2 Création et destruction de processus

Les processus SDL sont soit créés statiquement à l'initialisation du système soit dynamiquement lors de l'exécution. Pour chaque processus il est possible de définir des paramètres transmis au processus lors de la création de l'instance ainsi que le nombre minimal et maximal d'instances qui peuvent être créés. Lorsque le nombre minimal d'instances possibles d'un processus est supérieur à zéro, celles ci sont crées à l'initialisation du système.
Un processus qui vient d'être créé se retrouve dans l'état initial *start*. Il peut alors s'exécuter. Un processus peut se terminer avec l'instruction *stop* et devient inactif. Durant l'exécution d'un système SDL, il est possible de créer dynamiquement de nouvelles instances de processus en utilisant l'instruction *create*. Chaque instance de processus possède une adresse unique, appelée Pid (*process identifier*), qui l'identifie parmi les autres processus. Chaque processus peut accéder à son Pid à travers la variable *self*. Cette adresse peut être utilisée pour communiquer avec une instance particulière d'un processus. La figure 3.4 montre un exemple de création statique d'instances multiples du processus. Trois instances du processus *P1* sont créées à l'initialisation du système SDL. Chaque instance de *P1* aura un *Pid* différent, ce qui permettra à *P2* de les identifier.

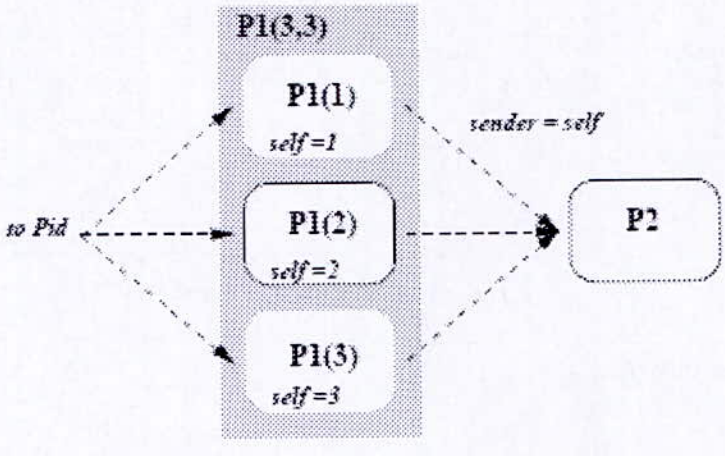


Figure 3.4 : Instances Multiples de Processus SDL

3.5.3 Communication

En SDL, la communication est basée sur le modèle de passage asynchrone de messages entre les processus. La communication est effectuée par l'intermédiaire de signaux. Les signaux SDL sont typés et peuvent transporter un ensemble de paramètres. Les signaux sont transférés

entre les processus par les routes et les canaux. Les routes sont directement connectées aux processus et finissent à la frontière des blocs où elles sont connectées aux canaux. les canaux connectent les blocs entre eux. A la frontière d'un bloc, un canal peut être connecté à plusieurs autres canaux ou routes. Si les processus sont contenus dans le même bloc alors les routes suffisent à acheminer les signaux. Si les processus sont situés dans des blocs différents, les signaux doivent emprunter des canaux. Les principales différences entre les routes et les canaux sont :

- une communication à travers une route s'effectue en temps nulle, alors que une communication à travers un canal subit un retard non déterministe. Aucune hypothèse ne peut être faite concernant le délai et l'ordre d'arrivée de signaux utilisant deux chemins différents ne peut pas être prédit.
- un canal peut effectuer une opération de routage sur un signal. Il peut router un signal à travers différents canaux (routes) connectés à la frontière du bloc en fonction du processus destinataire.
- les routes acheminent les signaux aux processus destinataires. Une route ne peut pas être connectée à plus d'un canal.

Les canaux et les routes peuvent être mono ou bidirectionnels. Si plusieurs signaux sont transférés sur un même canal ou route, leur ordre est préservé. Lors de leur acheminement par les canaux et les routes, les signaux ne sont pas autorisés à se doubler. Malgré tout, aucun ordre d'arrivée dans une file d'attente ne peut être prédit pour des signaux ayant suivi des canaux et des routes différents.

3.5.4 Comportement

Le comportement d'un système est décrit comme un ensemble de processus concurrents communicants (figure 3.5). Un processus est décrit comme un automate d'états finis qui communique avec les autres processus de façon asynchrone à travers des signaux. Chaque processus possède en entrée une file d'attente infinie dans laquelle les signaux sont stockés à leur arrivée. Les signaux sont extraits de la file d'attente par le processus dans leur ordre d'arrivée. En d'autres termes, les signaux sont bufférisés dans un ordre first-in-first-out (FIFO). Ce modèle de communication asynchrone faiblement couplé semble être un bon modèle pour la description de systèmes distribués.

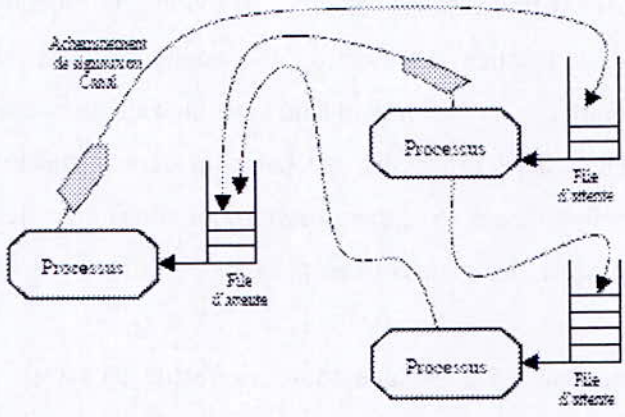


Figure 3..5 : Modèle de spécification SDL

Etats et transitions

Chaque processus est composé d'un ensemble d'états (*state*) et de transitions entre deux états (*nextstate*). Dans un état donné de l'automate, un processus est en attente sur un ensemble de signaux (*input*). L'arrivée d'un signal attendu dans la file d'attente valide une transition et le processus peut alors exécuter un ensemble d'opérations (*task*) telle que la manipulation de variables, des appels de procédure (*call*) et l'émission de signaux (*output*). Le signal reçu détermine la transition à exécuter. Lorsqu'un signal a initié une transition il est retiré de la file d'attente. Si aucune transition ne correspond au signal en tête de la file alors celui-ci est simplement consommé sans changement d'état de l'automate (transition implicite). Les transitions implicites ont pour but d'éviter qu'un processus ne reste infiniment bloqué dans le même état. Une transition mentionne le nom du signal et un ensemble de variables dans lequel seront placés les paramètres du signal. La variable prédéfinie *sender* recevra implicitement l'adresse du processus émetteur du dernier signal extrait de la file d'attente. Une transition peut être validée quel que soit le signal extrait de la file d'attente en utilisant *input* *. Dans ce cas l'astérisque remplace tous les signaux susceptibles d'être reçus par le processus. Un signal peut être retenu dans la file d'attente et son traitement peut être reporté dans un autre état à l'aide de l'instruction *save*. Cela permet d'affecter des priorités dans le traitement des signaux.

La *condition d'autorisation* permet d'ajouter une condition booléenne supplémentaire au déclenchement d'une transition. Une transition validée ne sera exécutée que si la condition d'autorisation est vraie. Sinon le signal sera sauvegardé et l'automate restera dans le même état. Les signaux continus permettent de définir des transitions gardées par des conditions

booléennes. Lorsque la condition est vraie la transition est exécutée. Un signal continu ne peut déclencher une transition que lorsque la file d'attente est vide. Si plusieurs signaux continus sont associés à un état, l'ordre d'évaluation des priorités peut être fixé en associant une priorité à chacun d'eux.

La figure 3.6 représente un exemple de spécification graphique de processus SDL. Dans l'état *st1* le processus est en attente sur les signaux *s2* et *s3* et le signal *s1* est sauvé. Le signal *s2* a deux paramètres, le signal *s3* un et le signal *s1* aucun. La figure 3.6.2 représente une transition implicite. Un signal extrait de la file d'attente dans un état donné (*st1*) et qui ne déclenche aucune transition est consommé sans changement d'état de l'automate. La figure 3.6.3 représente la sauvegarde du signal *s1*. Dans l'état *st2* (figure 3.6.4), la transition gardée par le signal *s4* est associée à une condition d'autorisation *z=7*, *z* étant une variable définie dans le processus. La transition ne pourra être exécutée que si cette condition est remplie. Cet état comporte aussi un signal continu gardé par l'équation booléene *b1=false*.

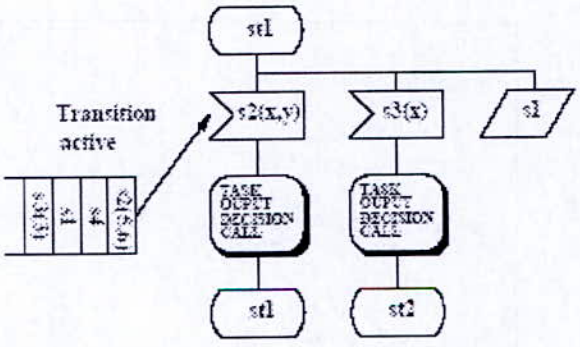


Figure 3.6.1: Illustration d'automate d'états finis

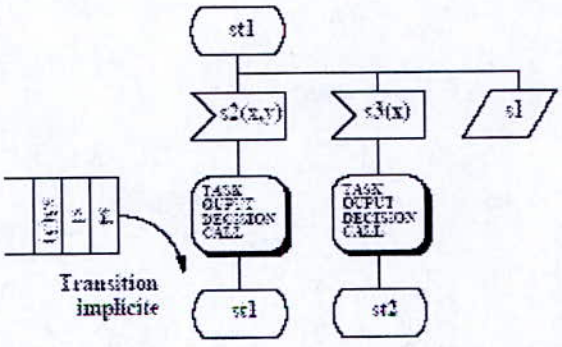


Figure 3.6.2: Illustration de transition implicite

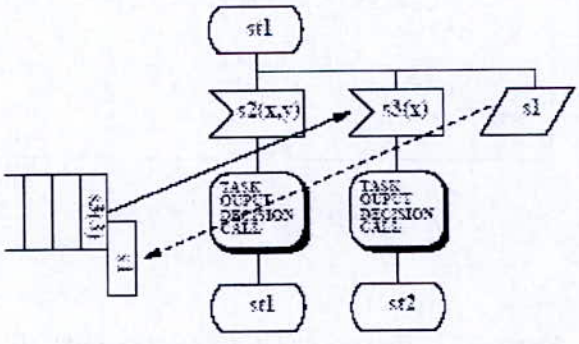


Figure 3.6.3: Illustration de sauvegarde d'un signal

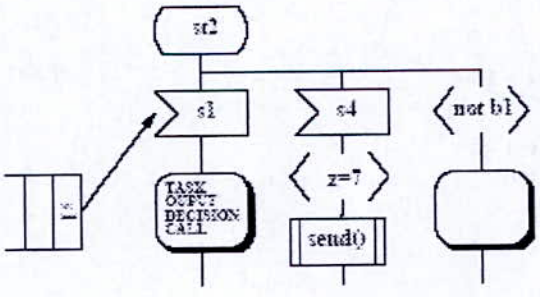


Figure 3.6.4: Illustration de signaux continus et conditions

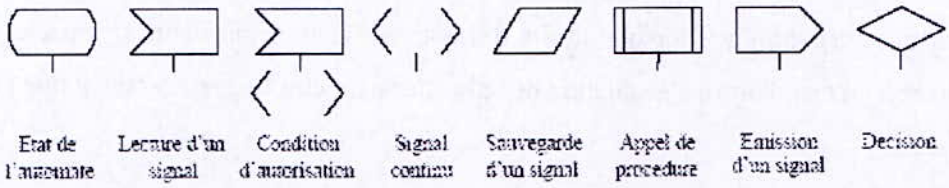


Figure 3.6 : Modèle d'exécution SDL

La figure 3.7.1 représente une spécification graphique d'un processus SDL et la figure 3.7.2 représente la description textuelle correspondante. L'état *start* est l'état d'initialisation. *input* représente la condition de garde d'une transition. Cette transition sera exécutée lorsque le signal correspondant sera extrait de la file d'attente. *task* représente une opération à effectuer lorsque la transition est exécutée et *output* émet un signal avec ses paramètres éventuels.

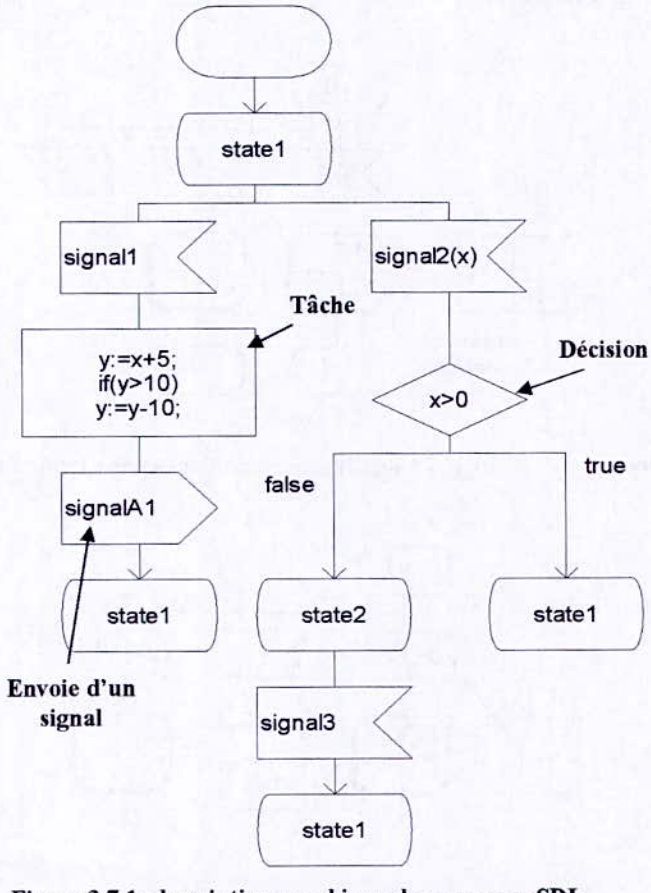


Figure 3.7.1: description graphique du processus SDL

```
start;
    nextstate state1;

state state1;
  input signal1;
    task {
      y := x + 5;
      if (y > 10)
        y := y - 10;
    };
    output signalA1;
    nextstate state1;
  input signal2(x);
    decision x > 0;
    (false) :
      nextstate state2;
    (true) :
      nextstate state1;
    enddecision;
endstate;
```

Figure 3.7.2 : Description textuelle du processus SDL

Figure 3.7 : Spécification du processus SDL

Types de variable

En SDL les variables sont propres à un processus et ne peuvent pas être modifiées par d'autres processus. La synchronisation interprocessus est effectuée à l'aide de signaux uniquement. Les variables prédéfinis en SDL sont représentées dans le tableau suivant :

Tableau 3.1: Type de donnée SDL

Type	Littéraux	Opérateurs
Integer	-∞,.. -1,0,.., +∞	+, -, *, /, =, <, >, float, fix
Real	-∞,..12.2,...+∞	+, -, *, /, =, <, >
Booléen	true, false	not, and, or, xor
Time	as real	+, -, =, <,>
Duration	as real	+, -, *, /, =,<, >
Character	ex : 'a' ,'-', '?' ,'2' ,'P'	=, <, >, num, chr
charstring	ex: 'abcdef 0', '$^&'	mkstring, length, first, last, //, subtring
Bit	0 ou 1	not, or, and, xor, =>
Bit_string	ex: 00001111001	mkstring, length, first, last, //, substring, bitstr, hexstr, not, and, xor, =>
Octet	ex: 02, AC, 4F, FF	not, and,or, xor, =, <, >, =>, i2o, o2i, mod, rem, shiftl, shiftr, +,-, *, /, bitstr, hexstr
Octet_string	ex: 005A7DFF	mkstring, length, first, last, //, substring, bit_string, octet_string, bitstr, hexstr

Bien entendu, SDL permet la définition de nouveaux types de données et de nouveaux opérateurs. L'implémentation de ces opérateurs s'effectue à l'aide du langage C, ce qui a comme avantage de bénéficier de la puissance de celui ci.

3.5.5 Procédures

SDL permet la définition de procédures dans un processus. Une procédure peut définir ses propres variables locales mais ne peut pas les exporter. Une procédure utilise la file d'attente

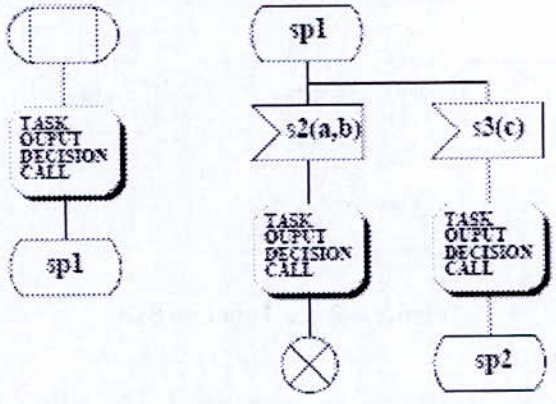


Figure 3.8 : Illustration d'une procédure en SDL

du processus où elle est définie pour déclencher des transitions. La figure 3.8 représente la procédure *send* appelée dans la figure 3.6.4.

Services

Le comportement d'un processus peut être spécifié comme une ensemble de services s'exécutant séquentiellement. Chaque service est défini séparément comme un automate d'états finis mais un seul service s'exécute à un moment donné. Lorsqu'un service atteint un nouvel état, le service capable de consommer le signal en tête de la file d'attente devient actif et s'exécute. Les différentes transitions des services sont donc entrelacées et il n'y a aucun risque d'interférence entre les différents services. Les différents services d'un processus partagent la file d'attente, les variables et les expressions *self* et *sender* du processus dans lequel ils sont définis.

Timers

En SDL, l'expression *NOW* contient la valeur courante de la variable globale de type *timer*, ainsi un processus peur armet un timer et recevoir un signal de timeout lorsque celui ci est arrivé à expiration. Le signal du timeout arrivera dans la file d'attente du processus et pourra être utilisé pour valider une transition.

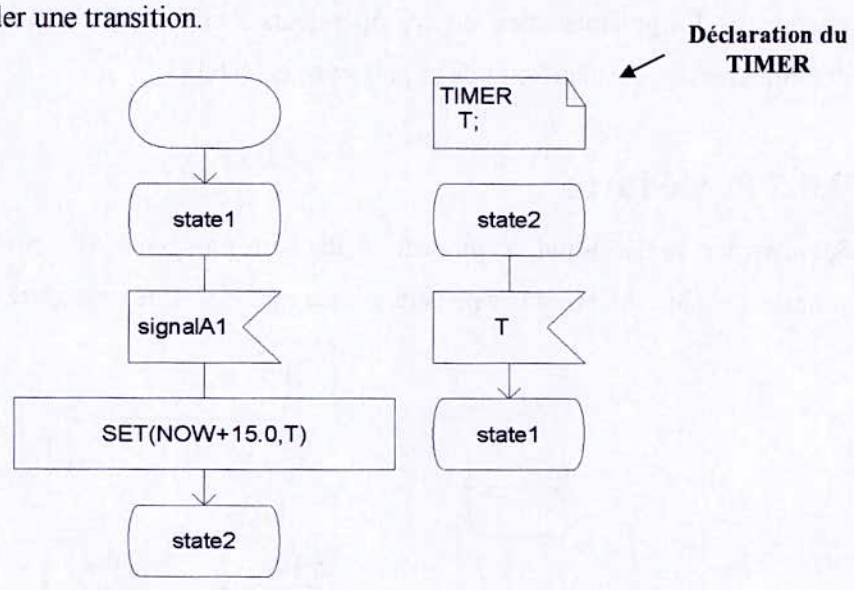


Figure 3.9: Le Timer en SDL

Dans le figure de dessus, on peut remarque que le timer T, a 15 unité de temps pour expirer, ainsi lors de l'expiration du timer T, le processus recevra le signal timeout T.

3.5.6 SDL, langage orienté OBJET

Pour une réutilisation facile de définition de type de données et de structures dans différents systèmes, celles ci peuvent êtres mises dans des *package* comme c'est montré dans la figure 3.10. Ces packages, peuvent être importés de n'importe quel système (figure 3.11).

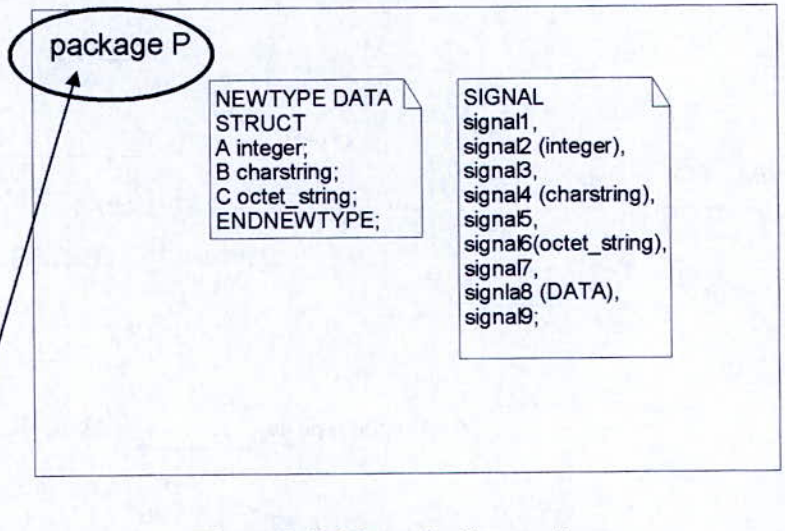


Figure 3.10: Exemple d'un package

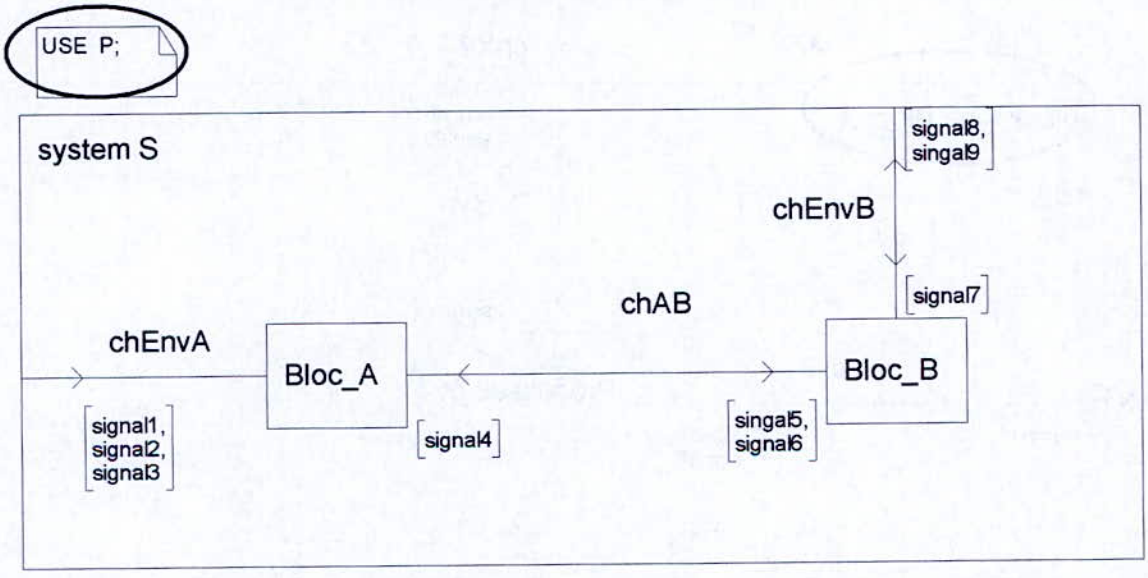


Figure 3.11: Exemple d'utilisation d'un package

Aussi, SDL permet la réutilisation de structure (systèmes, blocks, processus, services).En effet, chaque structure peut être typée, on peut définir des types de systèmes, des types de

blocs, des types de processus et des types de services. La figure 3.12 est un exemple d'un bloc typé.

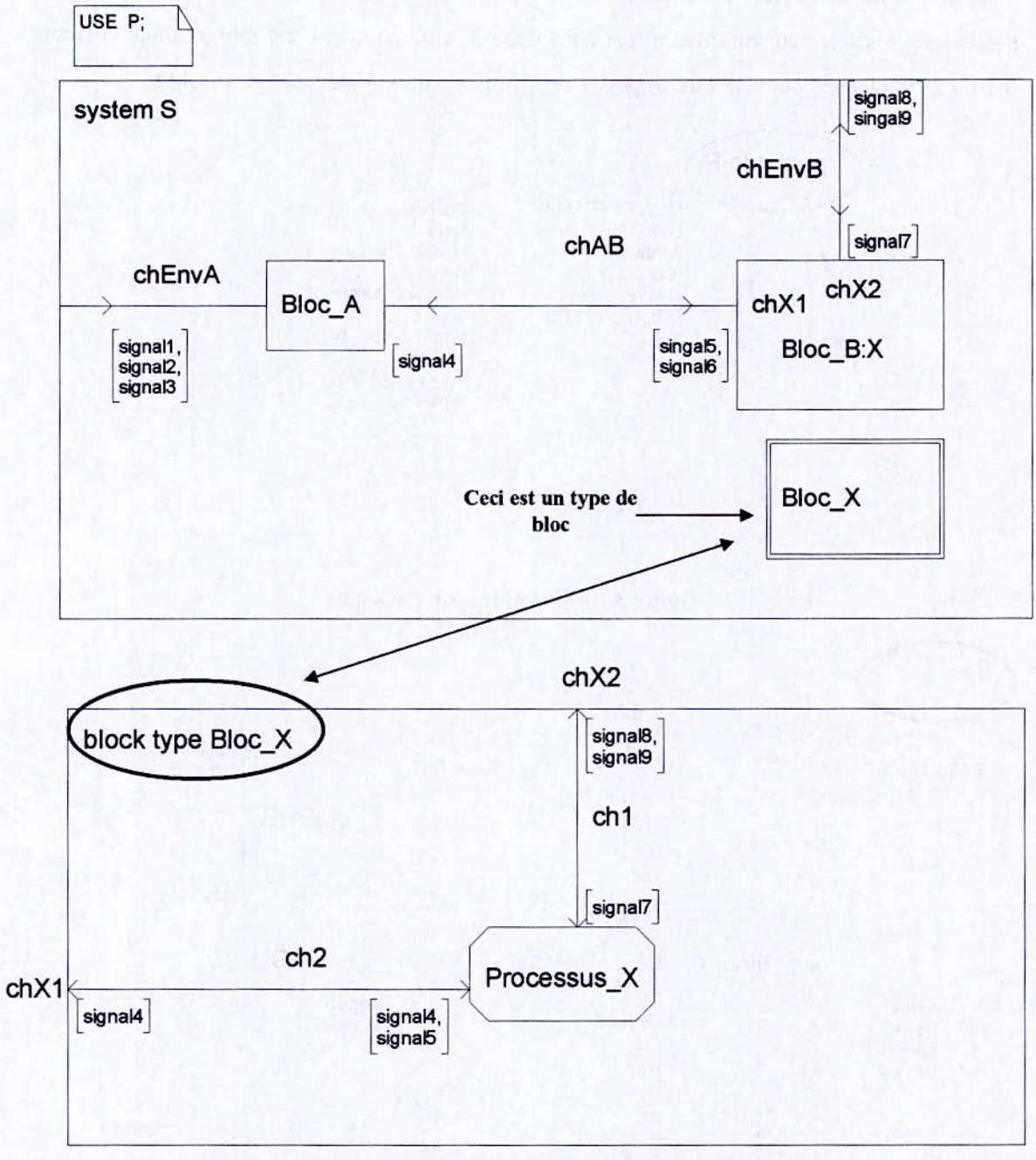


Figure 3.12: Exemple d'utilisation d'un type de bloc

3.6 Conclusion

Nous avons tenté dans ce chapitre de montrer que le langage formel SDL, se trouve être tout à fait approprié pour la spécification de notre système, autrement dit la modélisation du service DHCP.

Maintenant que la méthodologie de notre démarche a été définie, nous passons à la description en détails de l'environnement et des outils utilisés lors des différentes étapes, à partir de la modélisation jusqu'à la validation de notre description.

4 Modélisation SDL du service DHCP

4.1 Introduction

Dans ce chapitre nous allons présenter le modèle du service DHCP proposé.
Nous présentons, sous une forme généralisée, le comportement de notre système tout en donnant plus de détails dans la partie simulation en décrivant quelques cas précis.
Nous terminerons par la validation de notre modèle à travers la création d'un exécutable prêt à l'emploi qui sera testé en utilisant deux ordinateurs, l'un contenant l'exécutable en question.
Mais auparavant, il nous faut présenter l'environnement de développement TauSDL utilisé pour mener à bien ce travail.

4.2 Présentation de l'environnement de développement TauSDL

On se propose dans cette section de faire un petite présentation de l'environnement de développement TauSDL 4.4 développé par la société Telelogic [10], avec lequel nous avons pu modéliser, simuler et surtout valider notre modélisation du service DHCP.

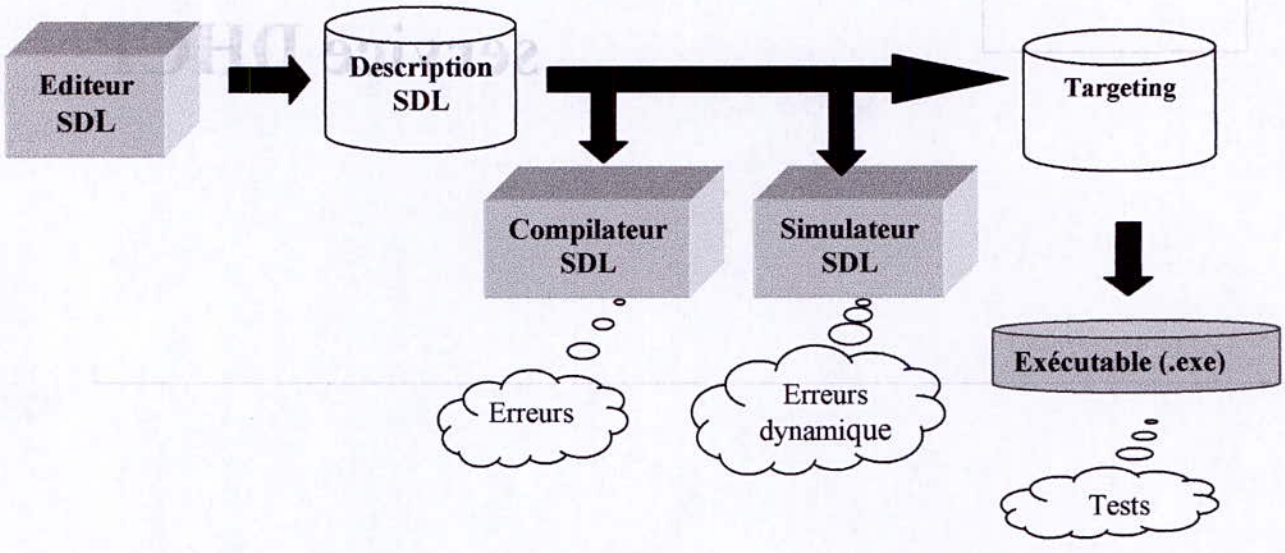


Figure 4.1 : Le flot de conception dans TauSDL

La figure 4.1 expose clairement la suite logique de la conception assistée par TauSDL. Ainsi, l'approche de création de tout projet se résume aux étapes suivantes :

1. On commence par décrire le programme en utilisant l'éditeur SDL,
2. Ensuite, avec le compilateur SDL, on prendra connaissance des erreurs de conception,
3. Puis, il y a le processus de validation simulée, en d'autres termes, le programme va simuler tous les cas possibles et va détecter les erreurs liées à l'utilisation du système,
4. L'avant dernière étape, est le ***targeting*** (ciblage en français). Elle consiste à cibler l'environnement dans le quel, on veut que notre programme tourne.
5. Enfin, lorsque toutes les étapes précédentes se sont bien déroulées, le code est alors généré, l'application est ainsi créée, un fichier exécutable est prêt à être utilisé

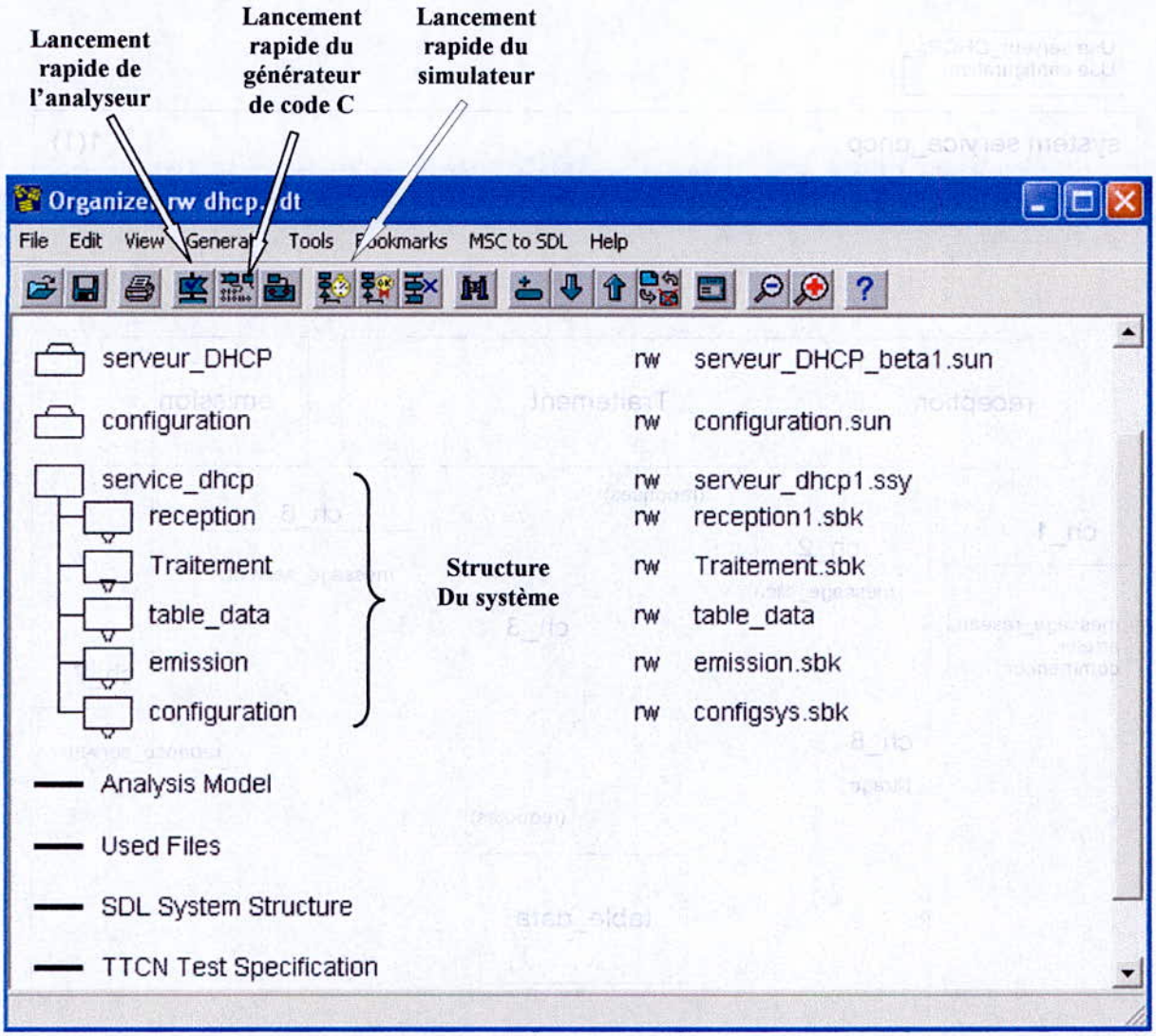


Figure 4.2: Fenêtre principale de TauSDL (OS Windows)

4.3 Modèle SDL du service DHCP

Le modèle du service DHCP proposé, se restreindra à la gestion d'un seul sous réseau, de ce fait, aucun agent de relais ne sera utilisé. Aussi ce service offrira un seul bail aux clients, à l'exception de ceux déclarés au niveau de la configuration qui peuvent avoir un bail infini.

La figure 4.3 est la description système du service DHCP. Comme présenté dans cette figure, notre système comporte cinq (05) blocs, chacun jouant un rôle bien déterminé et dont trois (03) communiquant avec l'environnement extérieur. Nous avons également défini des signaux (figure 4.4 et 4.5) qui peuvent être émis de et vers leurs blocs respectifs. En quelque sorte, ces signaux représentes un protocole de communication entres les cinq blocs.

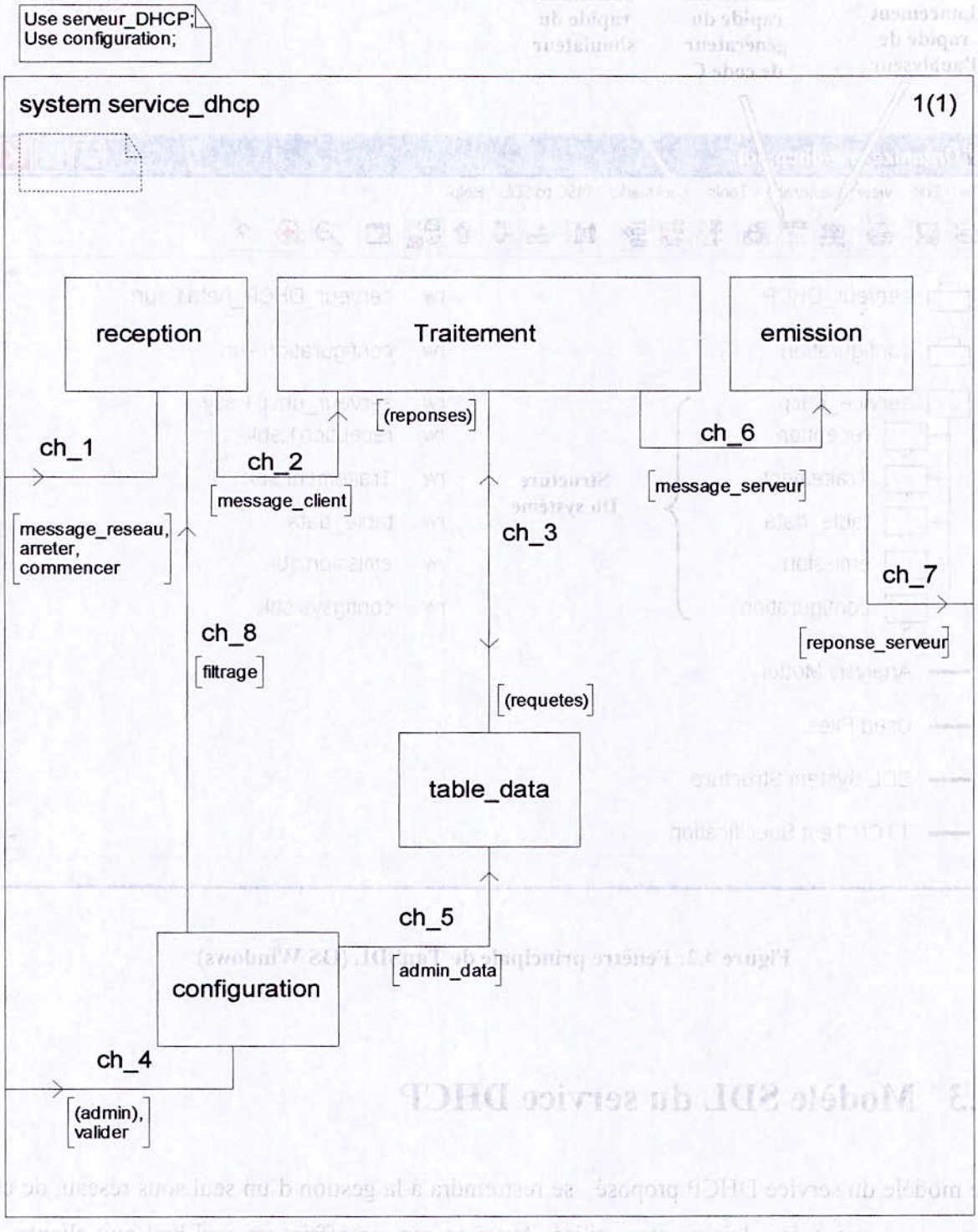


Figure 4.3 : Système SDL du service DHCP

Maintenant, nous allons présenter le rôle de chaque bloc en commençant pas ceux qui communiquent avec l'environnement extérieur, à savoir, le bloc *reception*, le bloc *emission* et le bloc *configuration*, en suite nous parlerons du bloc *table_data* et du bloc *Traitement* qui est le cœur de notre système.

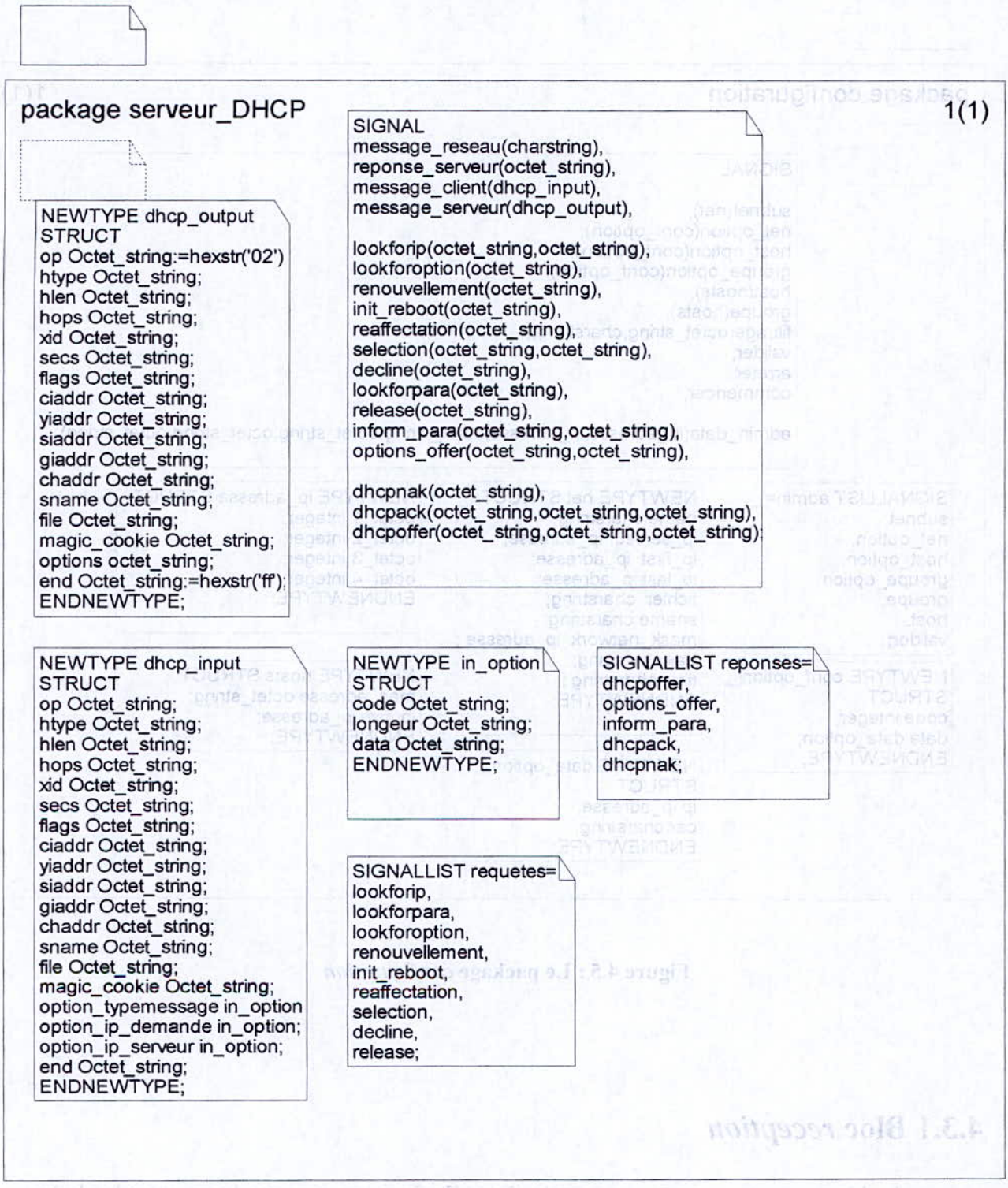


Figure 4.4 : Le package *serveur_DHCP*

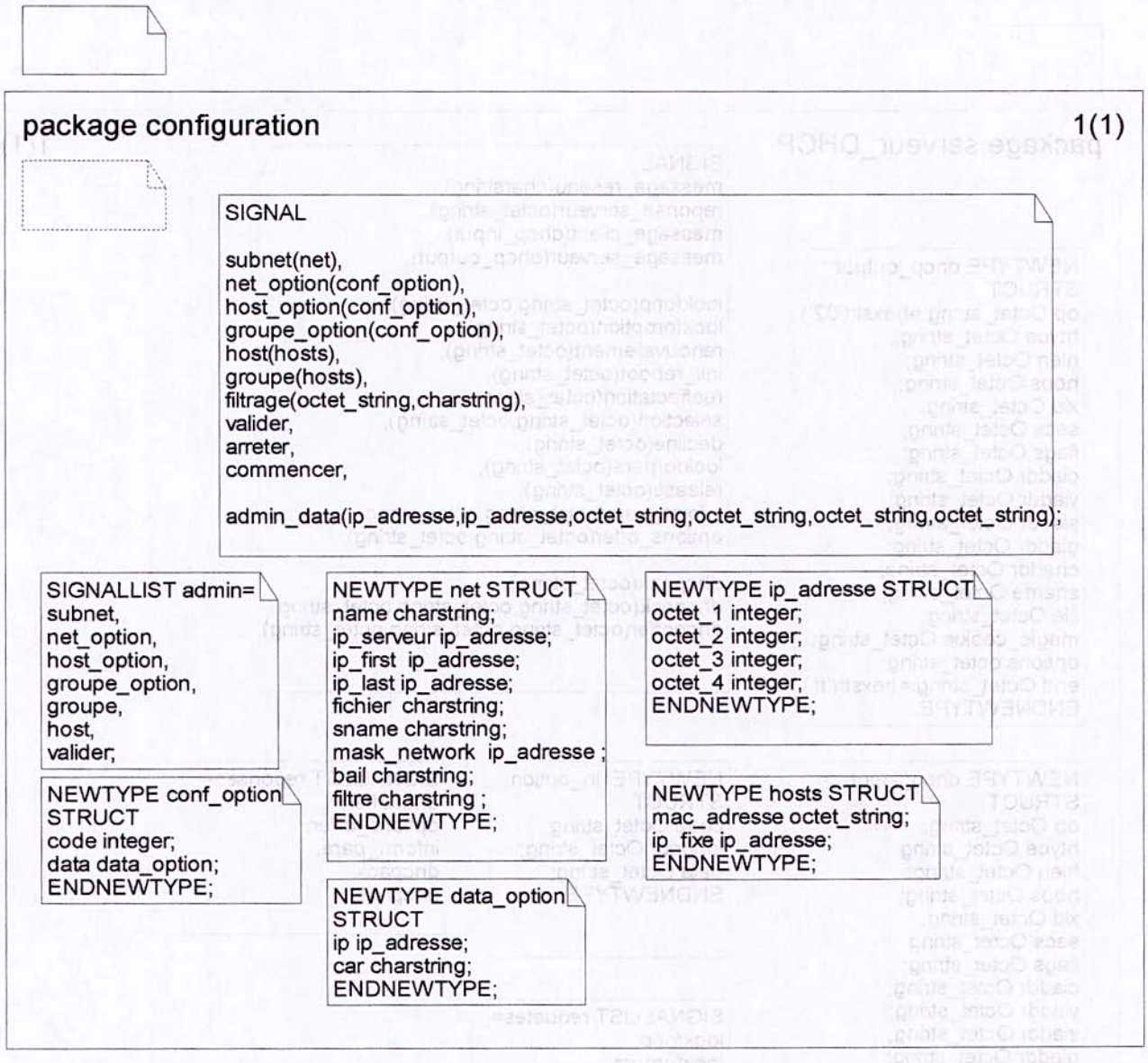


Figure 4.5 : Le package *configuration*

4.3.1 Bloc *reception*

Le bloc *reception* a pour principale tâche, de faire la correspondance entre la chaîne de caractères reçue à travers le signal *message_serveur,* et les différents champs constituant le message DHCP. En effet, dans le package *serveur_dhcp* nous avons défini un type de donnée appelé *dhcp_input* et dont la structure est la même que celle du message DHCP présenté dans le chapitre 2. Aussi, nous avons défini trois options que ce bloc doit localiser, ces options sont :

- l'option *type de message* (code 53) ;
- l'option *adresse IP demandée* (code 50) ;
- l'option *identifiant serveur* (code 54)

Ces options doivent être identifiées car on aura besoin de les traiter pour, entre autre, identifier le type de message. Bien entendu, on ne localisera pas le reste des options, car il est inutile de les traiter.

Après décomposition de la chaîne de caractères reçue, la nouvelle variable créée, de type *dhcp_input* sera transmise au bloc *Traitement* grâce au signal *message_client.*
Le bloc *reception* peut aussi recevoir le signal *commencer* et le signal *arreter* pour lancer ou arrêter notre système, ainsi que le signal *filtrage* provenant du bloc *configuration*, avec comme paramètre une liste d'adresses matérielles (adresse MAC) de clients déclarés lors de la configuration du serveur et l'activation ou non du filtrage.

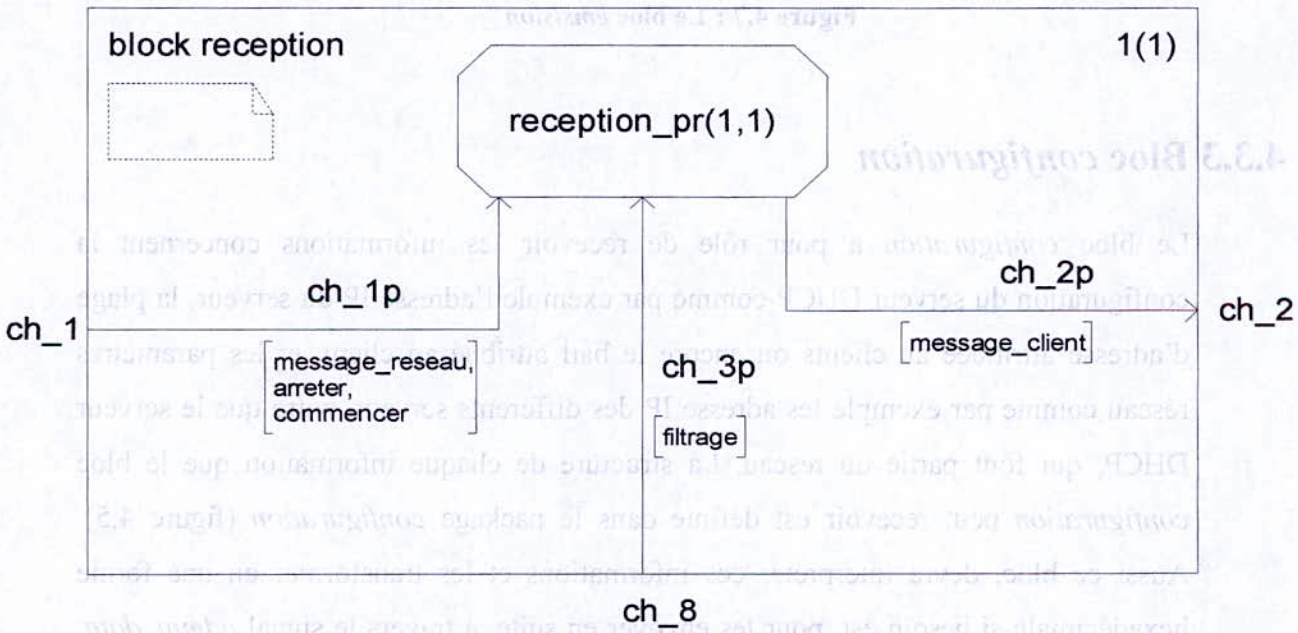


Figure 4.6 : Le bloc *reception*

4.3.2 Bloc *emission*

Nous passons maintenant à la présentation du bloc qui se trouve à l'autre extrémité de la chaîne de traitement, car il joue le même rôle que le bloc *reception* sauf que la fonction s'inverse. En effet, ce bloc, après réception du signal *message_serveur* avec comme paramètre une variable de type *dhcp_output* (voir package *serveur_dhcp*) du bloc *Traitement,* aura pour tâche de construire une chaîne de caractères à partir de

différents champs constituants le paramètre reçu. Cette opération se fait par concaténation.

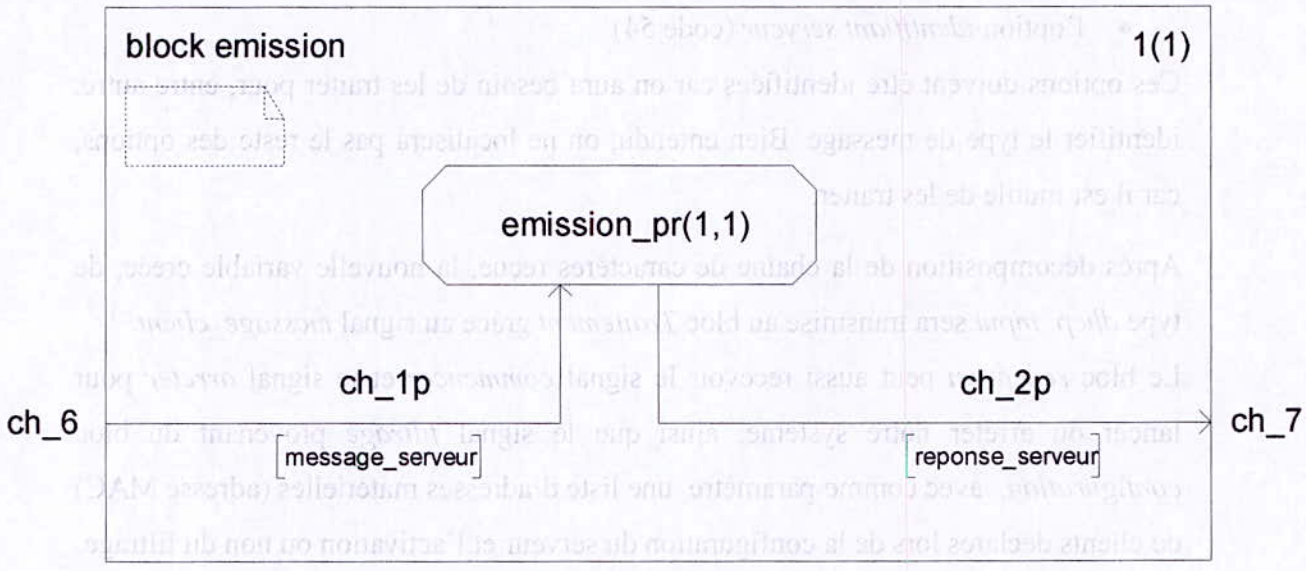


Figure 4.7 : Le bloc *emsision*

4.3.3 Bloc *configuration*

Le bloc *configuration* a pour rôle de recevoir les informations concernent la configuration du serveur DHCP comme par exemple l'adresse IP du serveur, la plage d'adresse attribuée au clients ou encore le bail attribué au client, et les paramètres réseau comme par exemple les adresse IP des différents serveur, autre que le serveur DHCP, qui font partie du réseau. La structure de chaque information que le bloc *configuration* peut recevoir est définie dans le package *configuration* (figure 4.5). Aussi ce bloc, devra interpréter ces informations et les transformer en une forme hexadécimale si besoin est, pour les envoyer en suite, à travers le signal *admin_data*, au bloc *table_data* qui joue le rôle de gestionnaire de données dans notre système.

On peut aussi déclarer des clients qui auront une configuration réseau particulière, comme par exemple une adresse IP fixe, ou des groupes de clients qui auront la même configuration réseau. Il est aussi possible d'indiquer si le serveur DHCP devra répondre ou non aux requêtes des clients non déclarés lors de la configuration (activation de la procédure de filtrage).

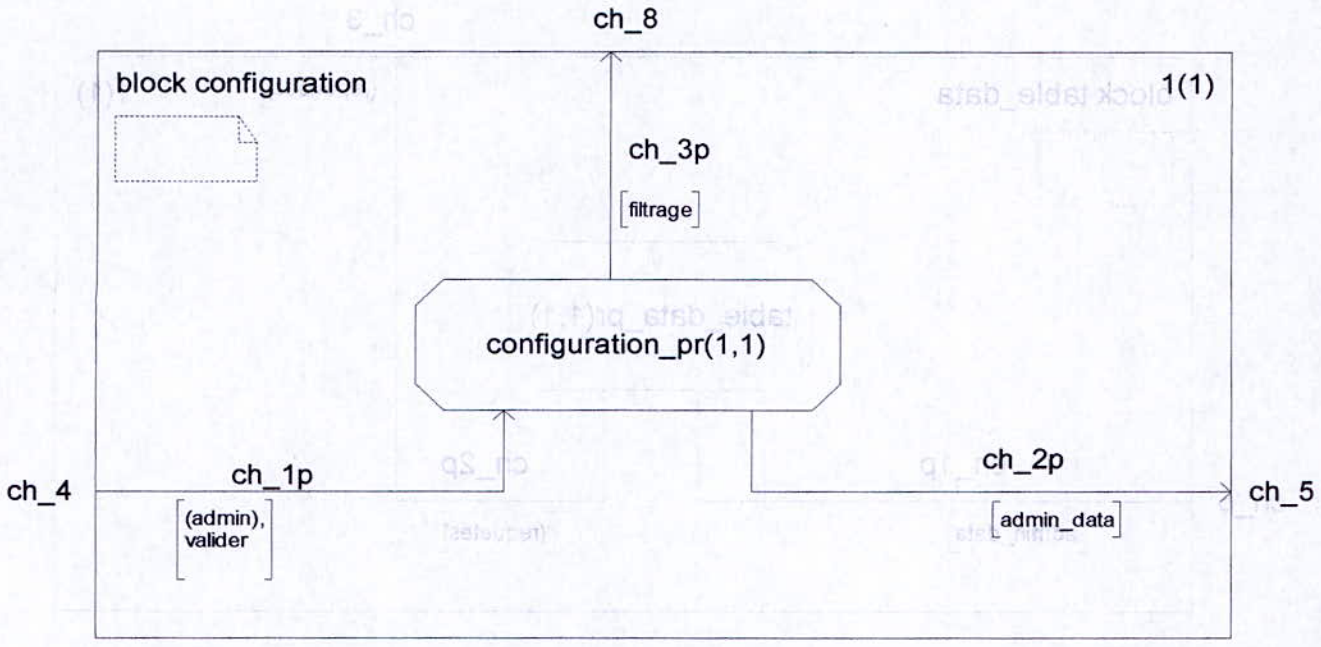


Figure 4.8 : Le bloc *configuration*

4.3.4 Bloc *table_data*

Comme nous l'avons déjà mentionné, ce bloc assure la gestion des informations qui garantissent le bon fonctionnement de notre service, à savoir, les informations concernent la configuration réseau, les informations concernent le configuration du service DHCP, éventuellement une liste de clients ou de groupes de clients déclarés dans la partie configuration, et surtout une trace de toutes les adresses offertes au clients. Aussi, ce bloc peut recevoir un ensemble de *requetes,* du bloc *Traitement,* pour par exemple, offrir une adresse IP à un client ou alors lui attribué des paramètres de configuration ou bien encore pour signaler que le client a une mauvaise configuration réseau.

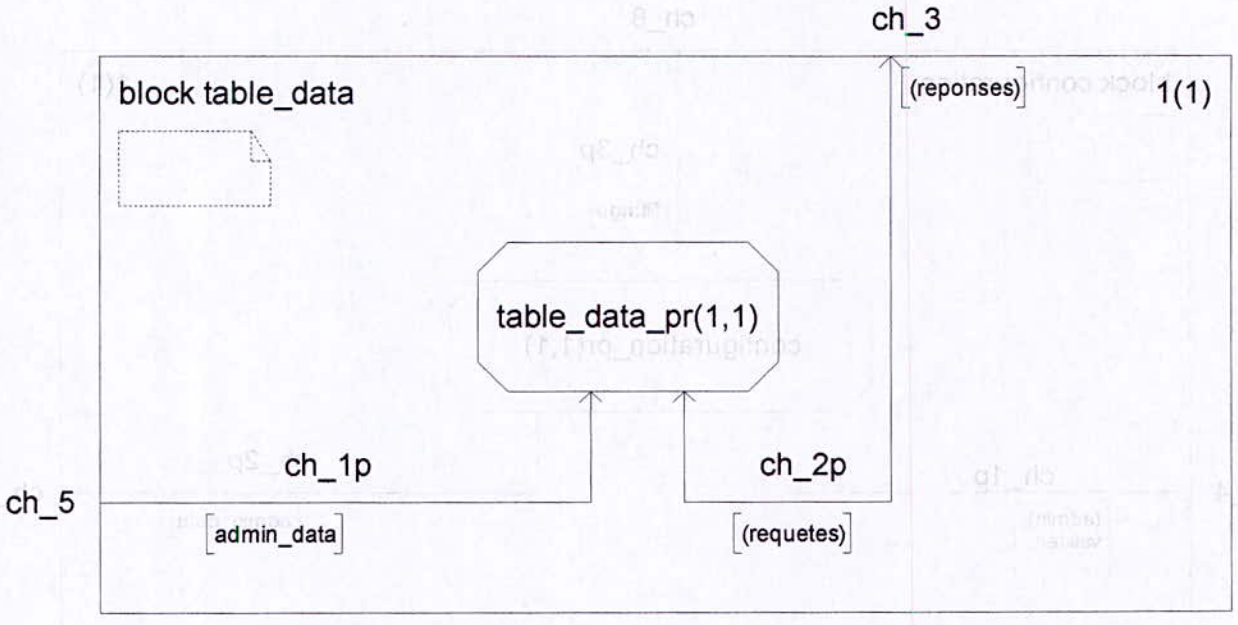


Figure 4.9 : Le bloc *table_data*

4.3.5 Bloc *Traitement*

Contrairement à tous les autres blocs, le bloc *Traitement* contient plus d'un processus, comme le montre la figure 4.10. En effet, ce bloc en contient cinq (05), chacun jouant un rôle bien déterminé, ces processus sont listés comme suite :

- **Le processus *typedemessage*** : qui a pour rôle de déterminer le type de message reçu, en traitant le champ option *type de message* (code 53), du message reçu. Selon les cas, ce processus va envoyer un signal correspondant au type de message reçu, au processus qui est chargé de le traiter ensuite. Par exemple, si le message reçu d'un éventuel client, est un message *DHCPDISCOVER* alors, dès que le processus *typedemessage* traitera l'option *type de message*, il va envoyer le signal *dhcpdiscover* au processus *DHCPdiscover* avec le même paramètre que celui du signal reçu *message_client,* afin qu'il continue de traiter la requête du client.

- **Le processus *DHCPdiscover*** : lorsque le processus *DHCPdiscover* est sollicité, cela veut dire que le client souhaite avoir une configuration réseau. A cet effet, ce processus devra envoyer deux requêtes (*lookfoip* et *looforoption*)

au bloc *table_data*, pour que celui-ci lui revoie comme réponse, à travers les signaux *dhcpoffer* et *options_offer* , l'adresse IP offerte au client et le reste de sa configuration réseau.

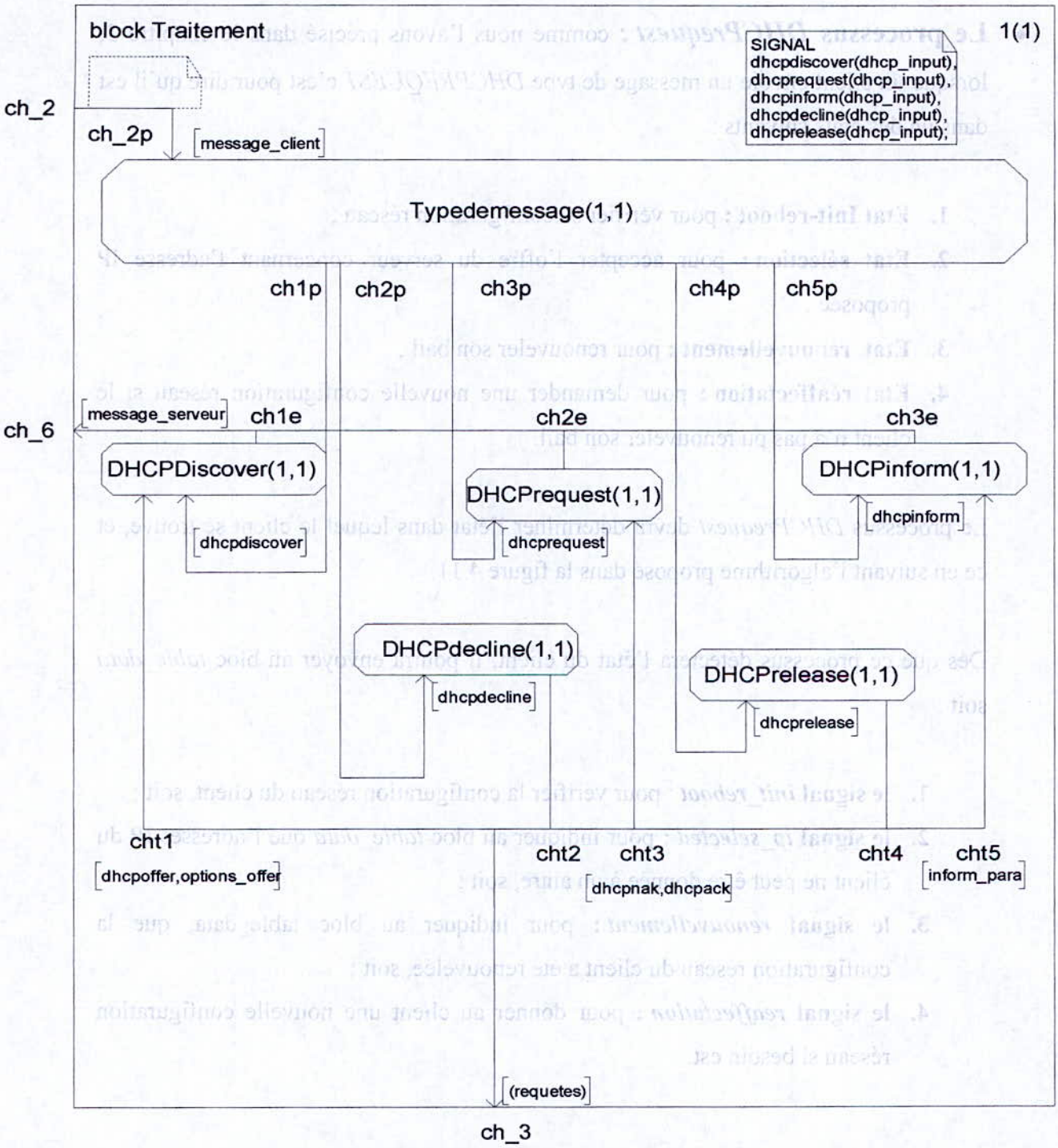


Figure 4.10 : Le bloc *Traitement*

- **Le processus *DHCPdecline*** : lorsque ce processus reçoit le signal *dhcpdecline* du processus *typedemessage,* cela signifie que le client décline l'offre que le serveur

DHCP a précédemment proposée. Le processus *DHCPdecline* émettra le signal *decline* vers bloque *table_data* qui devra mettre l'adresse IP du client comme étant indisponible.

- **Le processus *DHCPrequest* :** comme nous l'avons précisé dans le chapitre 2, lorsque un client envoie un message de type *DHCPREQUEST* c'est pour dire qu'il est dans un des états suivants :

  1. **Etat Init-reboot :** pour vérifier sa configuration réseau ;
  2. **Etat sélection :** pour accepter l'offre du serveur concernant l'adresse IP proposée ;
  3. **Etat renouvellement :** pour renouveler son bail ;
  4. **Etat réaffectation :** pour demander une nouvelle configuration réseau si le client n'a pas pu renouveler son bail.

  Le processus *DHCPrequest* devra déterminer l'état dans lequel le client se trouve, et ce en suivant l'algorithme proposé dans la figure 4.11.

  Dés que ce processus détectera l'état du client, il pourra envoyer au bloc *table_data* soit :

  1. **le signal *init_reboot*** : pour vérifier la configuration réseau du client, soit ;
  2. **le signal *ip_selected* :** pour indiquer au bloc *table_data* que l'adresses IP du client ne peut être donnée à un autre, soit ;
  3. **le signal *renouvellement* :** pour indiquer au bloc table_data, que la configuration réseau du client a été renouvelée, soit ;
  4. **le signal *reaffectation* :** pour donner au client une nouvelle configuration réseau si besoin est.

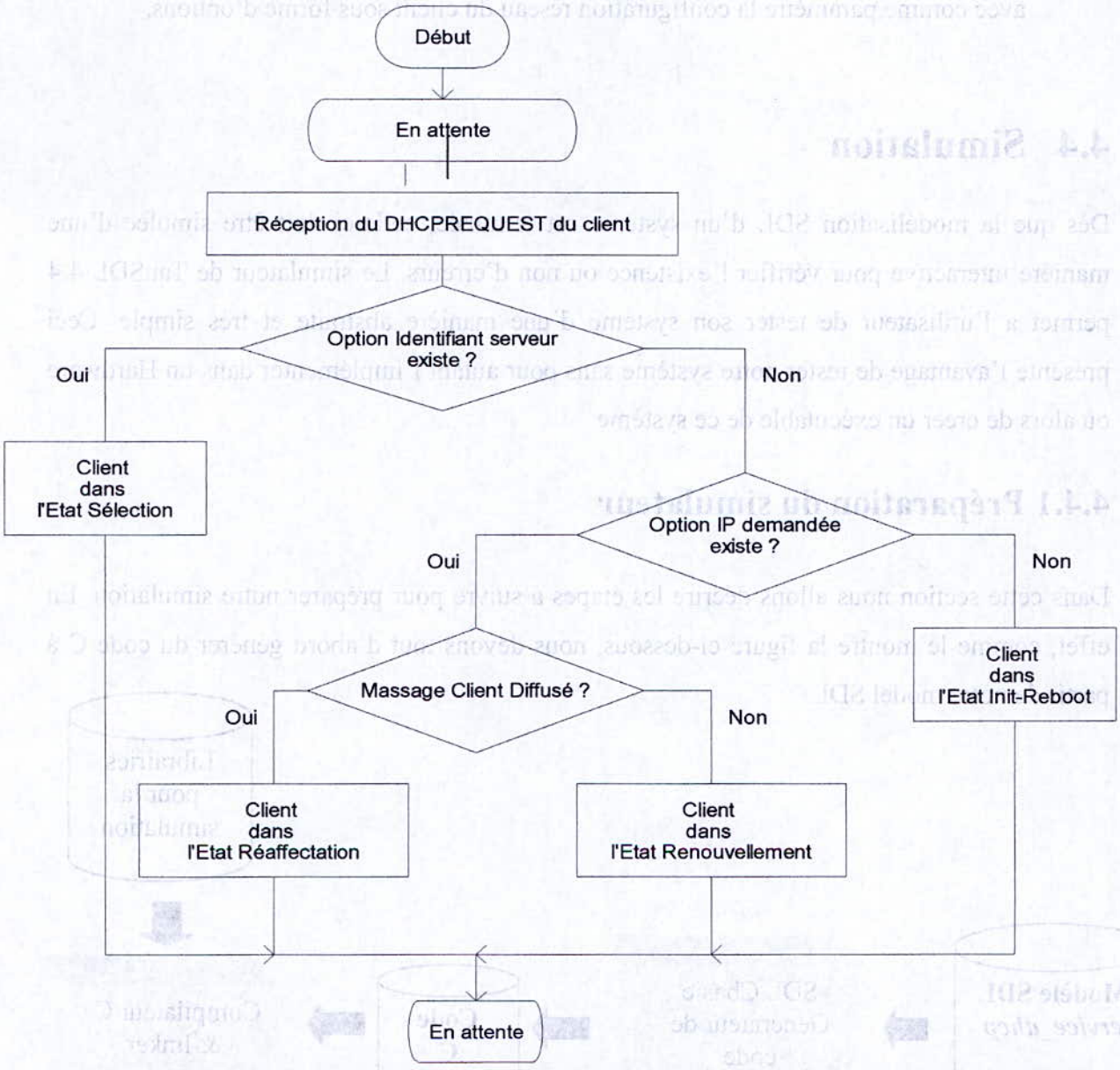


Figure 4.11 : Algorithme pour déterminer l'état du client dans le message DHCPREQUEST

- **Le processus *DHCPrelease* :** Dés que ce processus reçoit le signal *dhcprelease* du processus *typedemessage*, il transmettra au bloc *table_data* le signal *release* pour lui indiquer que le client libère son adresse IP. Ainsi le serveur pourra donner cette adresse à l'autre client si besoin est.

- **Le processus *DHCPinform* :** Le processus *DHCPinform* a pour tâche de demander au bloc *table_data* la configuration réseau que le client souhaiterait connaître. Et ce à travers le signal *inform* émis du processus *DHCPinform* vers le bloc

table_data. Le bloc *table_data* répond à ce processus grâce au signal *inform_para* avec comme paramètre la configuration réseau du client sous forme d'options.

4.4 Simulation

Dès que la modélisation SDL d'un système est terminée, celle-ci doit être simulée d'une manière interactive pour vérifier l'existence ou non d'erreurs. Le simulateur de TauSDL 4.4 permet à l'utilisateur de tester son système d'une manière abstraite et très simple. Ceci présente l'avantage de tester notre système sans pour autant l'implémenter dans un Hardware ou alors de créer un exécutable de ce système.

4.4.1 Préparation du simulateur

Dans cette section nous allons décrire les étapes à suivre pour préparer notre simulation. En effet, comme le montre la figure ci-dessous, nous devons tout d'abord générer du code C à partir de notre model SDL.

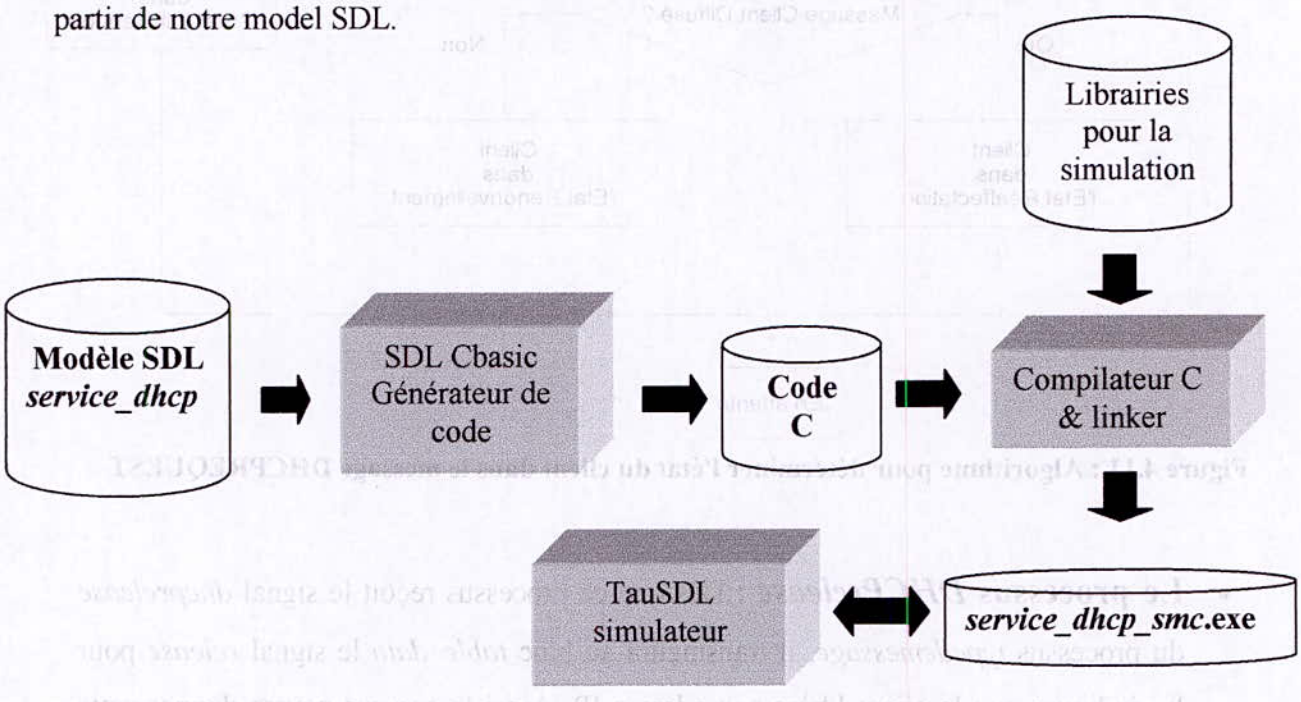


Figure 4.12 : Schéma de préparation de la simulation

Ensuite, le code C générer devra être compilé pour créer un exécutable avec lequel le simulateur peut interagir. Ces opérations se font à travers de l'instruction ***make*** de TauSDL.

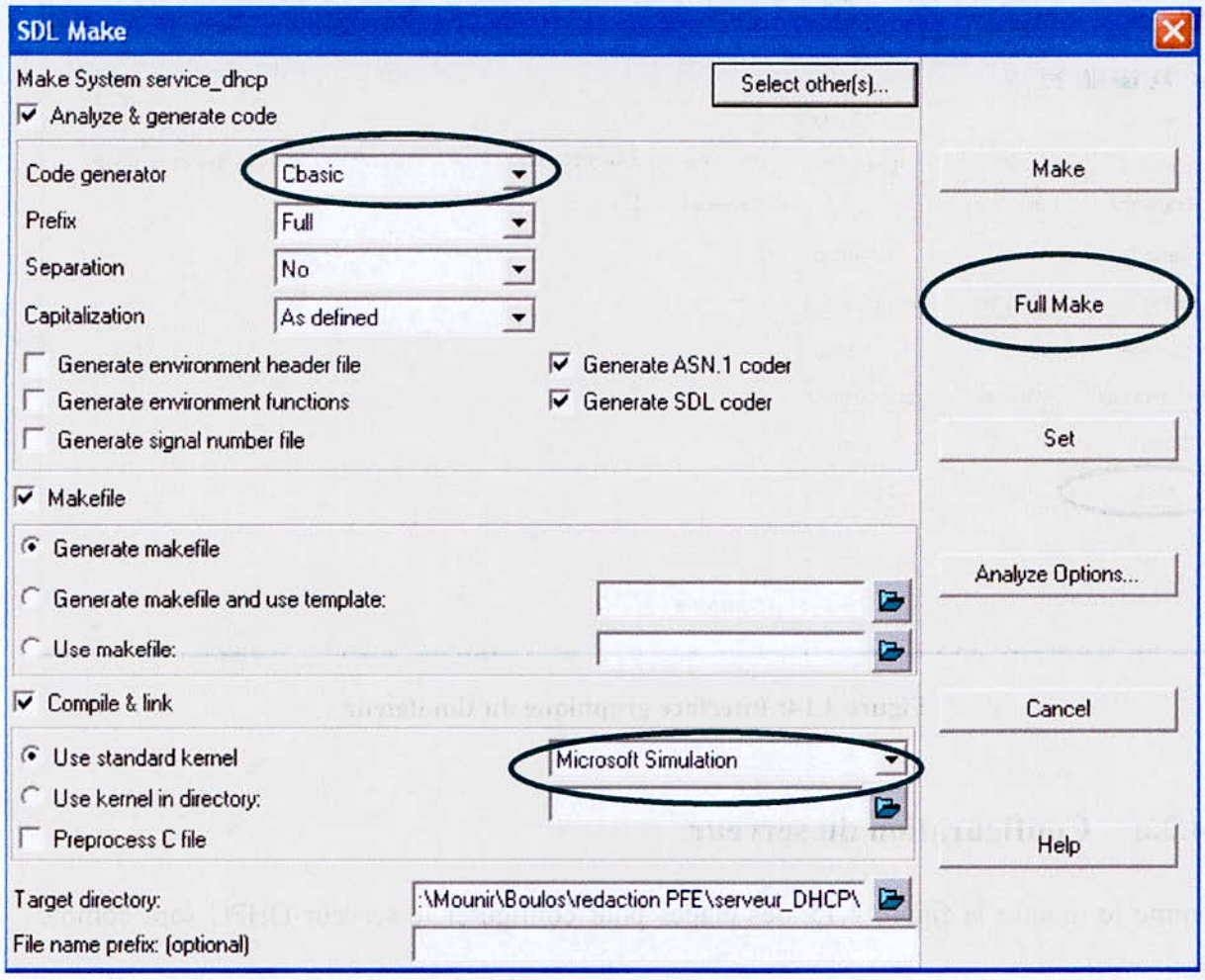


Figure 4.13 : La fenêtre *make* de TauSDL

4.4.2 Lancement du simulateur

Après cette préparation, nous pourrons lancer la simulation. Le simulateur de l'environnement de développement TauSDL nous permet de voir d'une manière graphique et très intuitive le comportement de notre système en utilisant l'éditeur MSC (message sequence chart).

Comme nous l'avons énoncé en introduction, nous allons simuler quelque cas précis en donnant, cette fois-ci, plus de détails, que durant la phase de présentation du modèle. La simulation que nous allons faire concerne les points suivants :

- La configuration du serveur DHCP ;
- L'attribution à un client d'une configuration réseau (adresse IP et paramètres réseau) ;
- La déclaration d'un client au niveau de la configuration ;
- Tester la procédure de filtrage des clients.

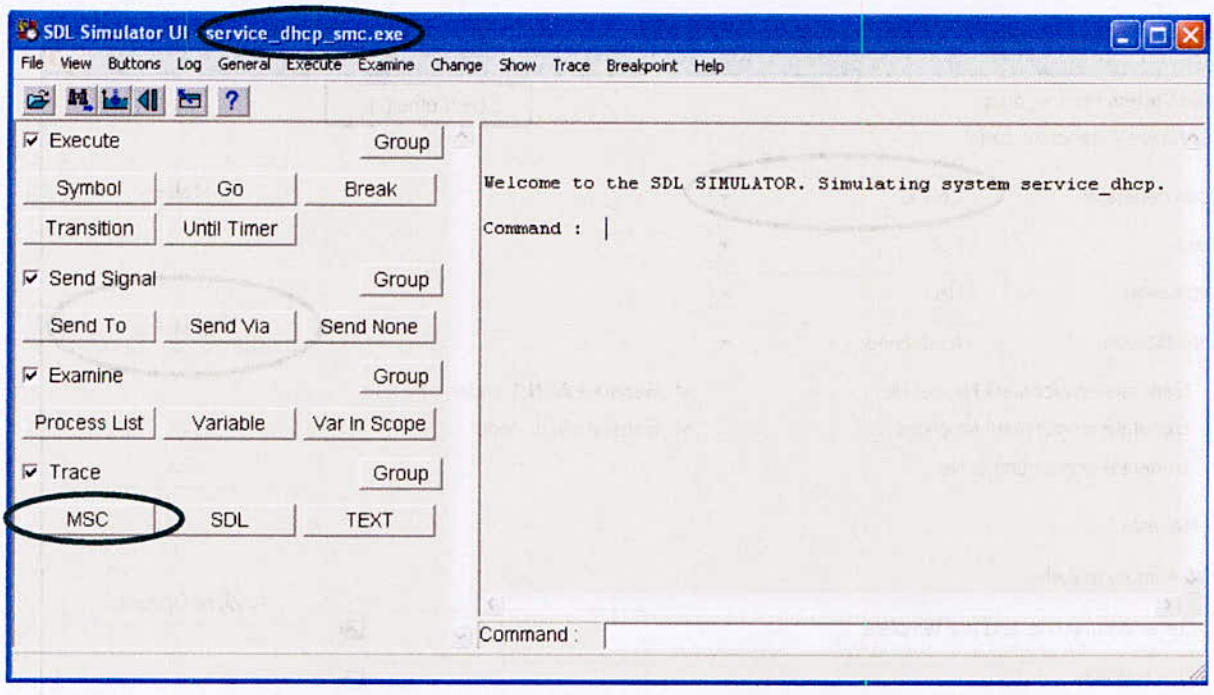


Figure 4.14: Interface graphique du simulateur

4.4.2.a Configuration du serveur

Comme le montre la figure 4.15 les étapes pour configurer le serveur DHPC sont comme suit:

1. Après avoir envoyé le signal *commencer* pour lancer le serveur, on débute la configuration de notre serveur par l'envoie au bloc *configuration* du signal *subnet* avec les paramètre indiqués sur la figure.
2. Le processus *configuration_pr* étant dans l'état *waintforoption,* on indique les options concernant le configuration réseau grâce au signal *net_option*, cette opération s'arrête lorsque on envoie le même signal avec le paramètre nul.
3. On termine par valider l'opération en envoyant le signal *valider.*

On peut remarquer, qu'après la validation de la configuration du serveur, le processus *configuration_pr* envoie à travers le signal *admin_data* :

- La plage d'adresse attribuée au client ;
- L'adresse IP du serveur DHCP sous une forme Hexadécimale ;
- Toutes les options, en hexadécimale, qui doivent être mises dans le champs '*option*' du message DHCP par le serveur DHCP ;

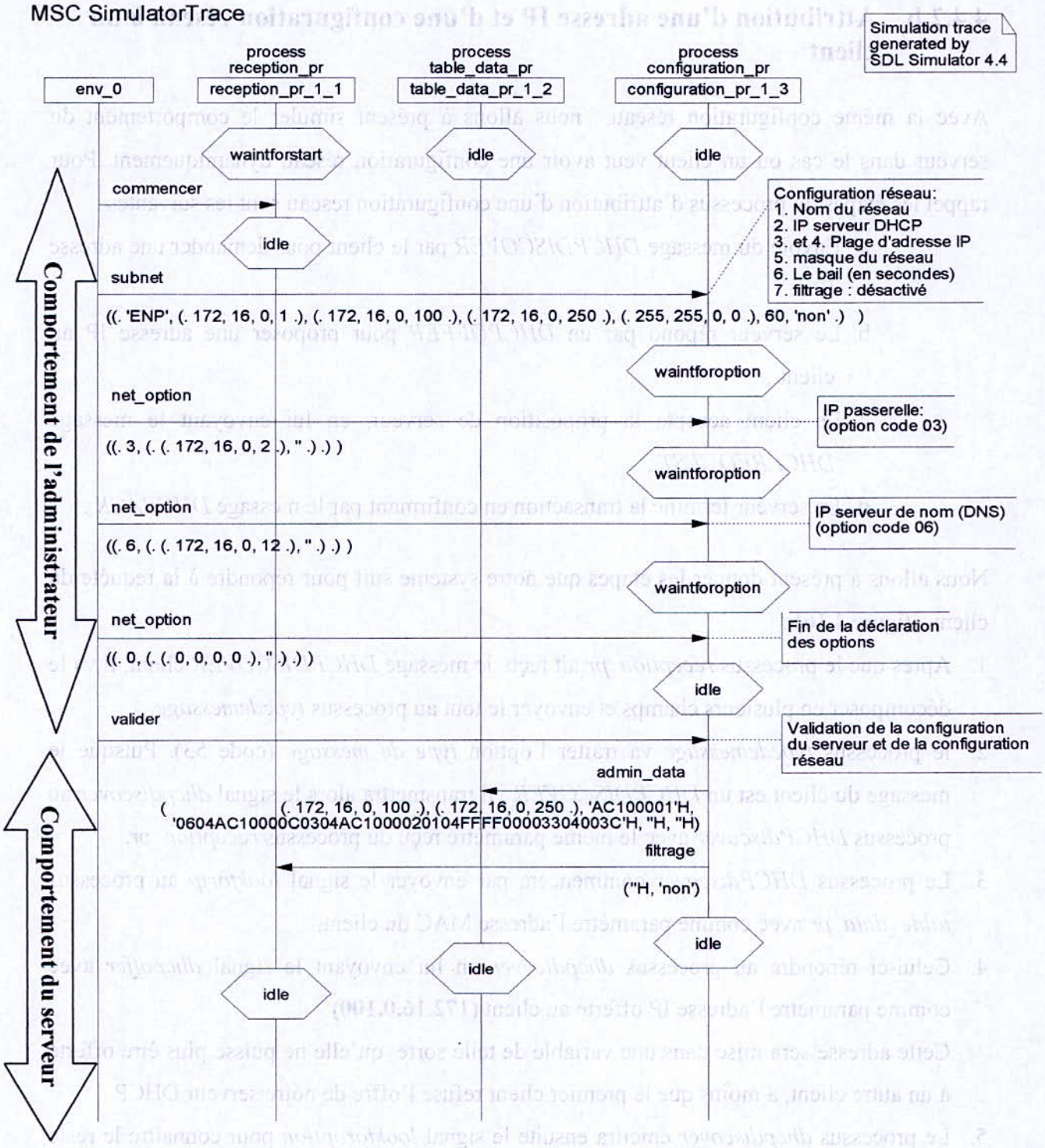


Figure 4.15 : Configuration du serveur

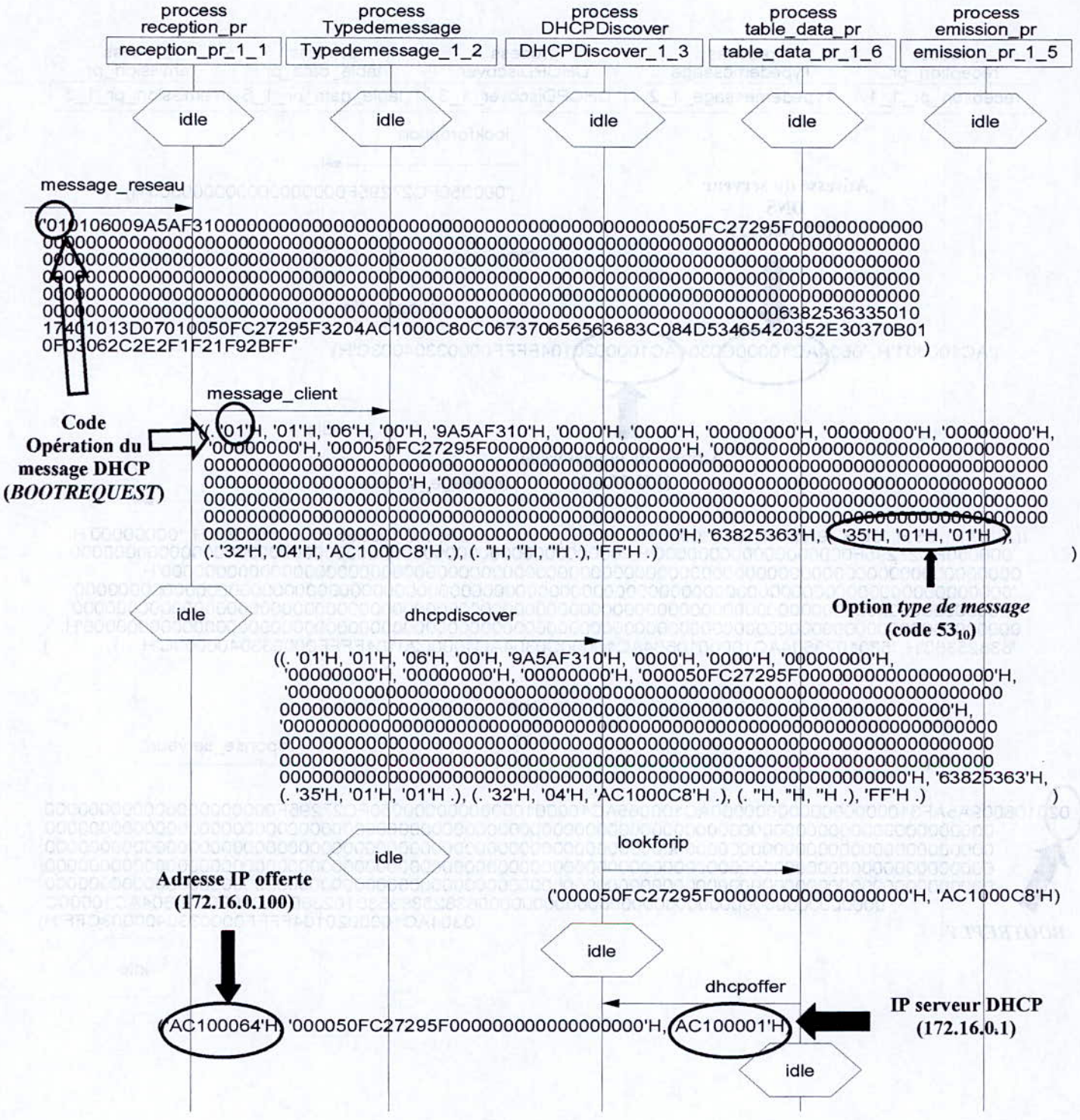
4.4.2.b Attribution d'une adresse IP et d'une configuration réseau à un client

Avec la même configuration réseau, nous allons à présent simuler le comportement du serveur dans le cas ou un client veut avoir une configuration réseau dynamiquement. Pour rappel les étapes du processus d'attribution d'une configuration réseau sont les suivantes :

a. Envoie du message *DHCPDISCOVER* par le client pour demander une adresse IP ;
b. Le serveur répond par un *DHCPOFFER* pour proposer une adresse IP au client ;
c. Le client accepte la proposition de serveur, en lui envoyant le message *DHCPREQUEST* ;
d. Le serveur termine la transaction en confirmant par le message *DHCPACK* .

Nous allons à présent donner les étapes que notre système suit pour répondre à la requête du client (figure 4.16) :

1. Après que le processus *reception_pr* ait reçu le message *DHCPDISCOVER* client, il va le décomposer en plusieurs champs et envoyer le tout au processus *typedemessage*.
2. le processus *typedemessage* va traiter l'option *type de message* (code 53). Puisque le message du client est un *DHCPDISCOVER* ; il transmettra alors le signal *dhcpdiscover* au processus *DHCPdiscover* avec le même paramètre reçu du processus *reception_pr*.
3. Le processus *DHCPdiscover* commencera par envoyer le signal *lookforip* au processus *table_data_pr* avec comme paramètre l'adresse MAC du client.
4. Celui-ci répondra au processus *dhcpdicover* en lui envoyant le signal *dhcpoffer* avec comme paramètre l'adresse IP offerte au client **(172.16.0.100)**.
   Cette adresse sera mise dans une variable de telle sorte qu'elle ne puisse plus être offerte à un autre client, a moins que le premier client refuse l'offre de notre serveur DHCP.
5. Le processus *dhcpdiscover* émettra ensuite le signal *lookforoption* pour connaître le reste de la configuration réseau que le client devra avoir.
6. Le processus *table_data_pr* répondra en envoyant le signal *option_offer* .
7. Le processus *dhcpdiscover* transmettra alors le signal *message_serveur*, avec comme paramètres la nouvelle valeur des champs du message DHCP au processus *emission_pr*.
8. le processus *emission_pr* formera une chaîne de caractères à partir des différents champs reçus à travers le signal *message_serveur* et l'envoie au client.


process reception_pr
reception_pr_1_1
process Typedemessage
Typedemessage_1_2
process DHCPDiscover
DHCPDiscover_1_3
process table_data_pr
table_data_pr_1_6
process emission_pr
emission_pr_1_5
idle
message_reseau
message_client
Code Opération du message DHCP (BOOTREQUEST)
Option type de message (code 53$_{10}$)
dhcpdiscover
lookforip
('000050FC27295F00000000000000000'H, 'AC1000C8'H)
Adresse IP offerte (172.16.0.100)
dhcpoffer
IP serveur DHCP (172.16.0.1)
'AC100064'H
'000050FC27295F00000000000000000'H,
'AC100001'H

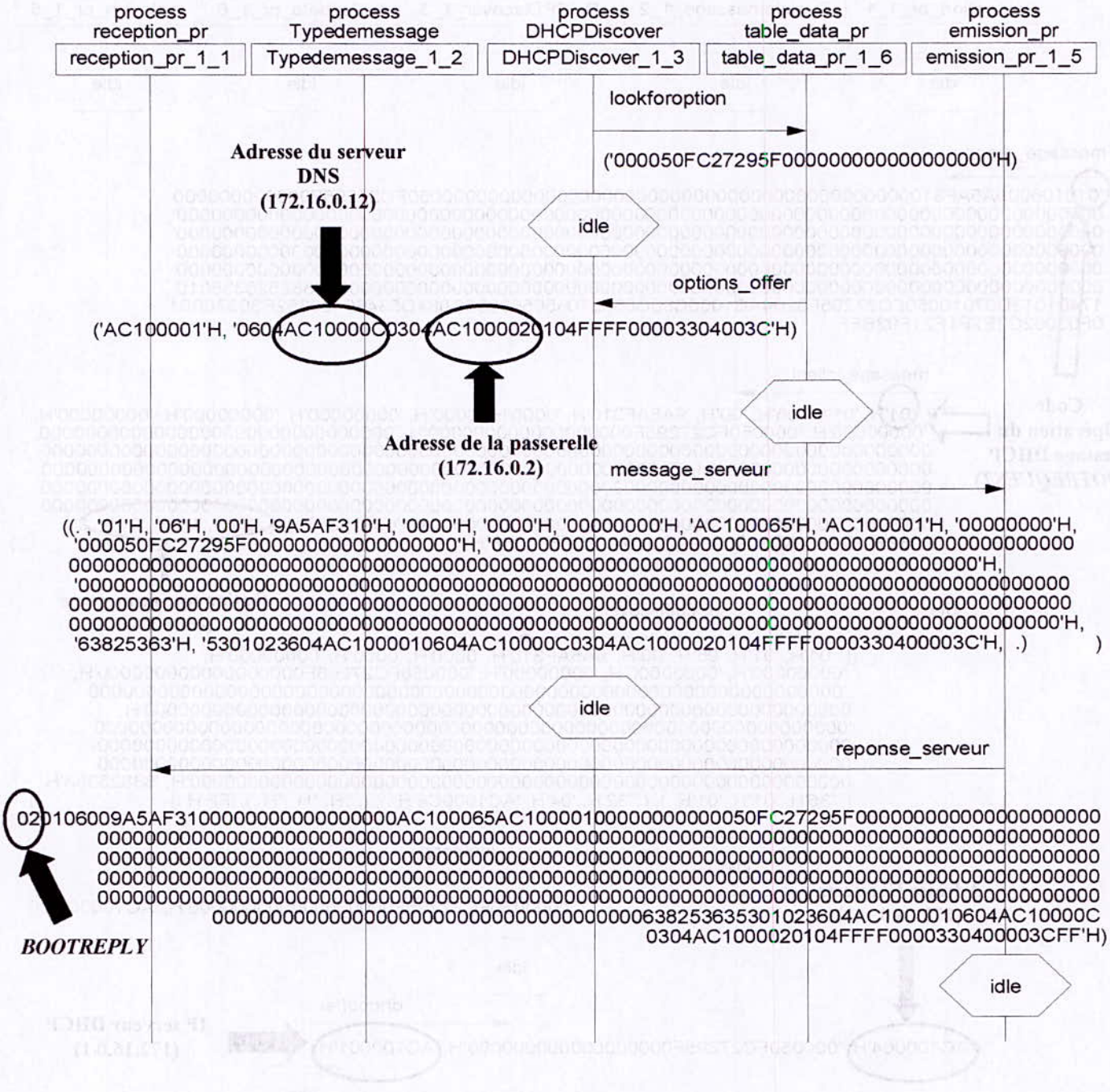
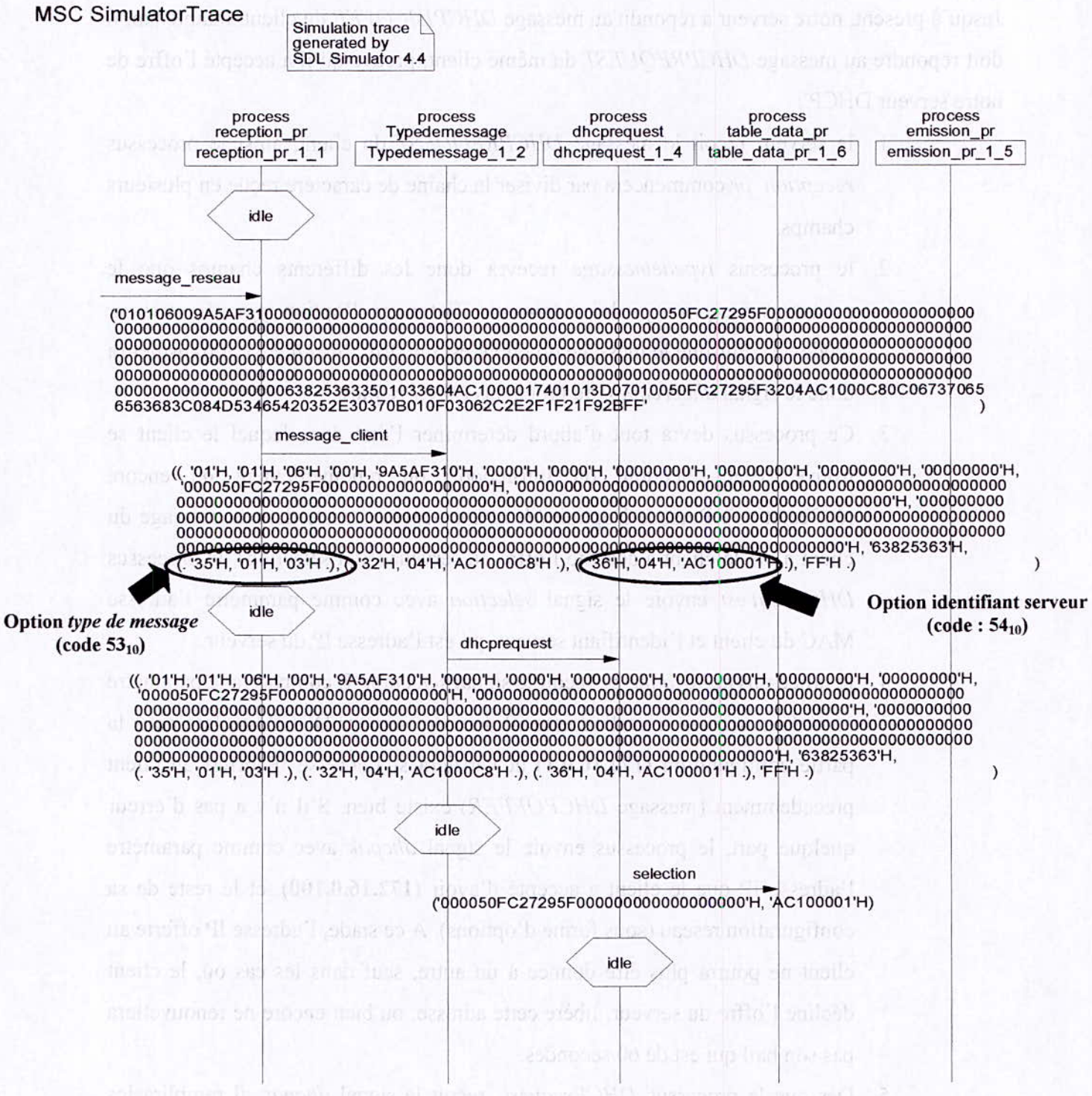


Figure 4.16 : Traitement du message *DHCPDISCOVER*

Jusqu'à présent, notre serveur à répondu au message *DHCPDICOVER* du client, maintenant il doit répondre au message *DHCPREQUEST* du même client, preuve qu'il a accepté l'offre de notre serveur DHCP :

1. le serveur reçoit le message *DHCPREQUEST* du client, ainsi le processus *reception_pr* commencera par diviser la chaîne de caractère reçue en plusieurs champs.
2. le processus *typedemessage* recevra donc les différents champs que le processus *reception_pr* lui a transmis. Il traitera l'option *type de message* (code 53). Puisque le message du client est un *DHCPREQUEST* il transmettra donc le signal *dhcprequest* au processus *DHCPrequest.*
3. Ce processus devra tout d'abord déterminer l'état dans lequel le client se trouve (Etat sélection, ou Init-reboot, ou renouvellement ou encore réaffectation). Puisque l'option *identifiant serveur* existe dans le message du client alors le client a accepté l'offre du serveur (Etat sélection). le processus *DHCPrequest* envoie le signal *selection* avec comme paramètre l'adresse MAC du client et l'identifiant serveur qui est l'adresse IP du serveur.
4. le processus *table_data_pr* reçoit le signal *selection.* Il comparera le paramètre *identifiant serveur* avec l'adresse IP de notre serveur DHCP, déclaré dans la partie configuration **(172.16.0.1)** et vérifiera si l'adresse IP offert au client précédemment (message *DHCPOFFER*) existe bien. S'il n'y a pas d'erreur quelque part, le processus envoie le signal *dhcpak* avec comme paramètre l'adresse IP que le client a accepté d'avoir **(172.16.0.100)**, et le reste de sa configuration réseau (sous forme d'options). A ce stade, l'adresse IP offerte au client ne pourra plus être donnée à un autre, sauf dans les cas où, le client décline l'offre du serveur, libère cette adresse, ou bien encore ne renouvellera pas son bail qui est de 60 secondes.
5. Dès que le processus *DHCPrequest* reçoit le signal *dhcpak,* il remplira les champs constituant le message DHCP à envoyer au client, et les envoie au processus *emission.*
6. Le processus *emission,* construira alors une chaîne de caractères à partir des valeurs des différents champs reçus du processus *DHCPrequest.* Dés que cette opération de concaténation est terminée le serveur répondra au client à travers le signal *reponse_serveur* .


MSC SimulatorTrace
Simulation trace generated by SDL Simulator 4.4
process reception_pr
process Typedemessage
process dhcprequest
process table_data_pr
process emission_pr
reception_pr_1_1
Typedemessage_1_2
dhcprequest_1_4
table_data_pr_1_6
emission_pr_1_5
idle
message_reseau
('010106009A5AF310000000000000000000000000000000000000000050FC27295F0000000000000000000000
message_client
((. '01'H, '01'H, '06'H, '00'H, '9A5AF310'H, '0000'H, '0000'H, '00000000'H, '00000000'H, '00000000'H, '00000000'H,
'000050FC27295F0000000000000000'H, '00000000000000000000000000000000000000000000000000000000
'0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000'H, '00000000
'00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
'00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
'00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000'H, '63825363'H,
(. '35'H, '01'H, '03'H .), (. '32'H, '04'H, 'AC1000C8'H .), (. '36'H, '04'H, 'AC100001'H .), 'FF'H .)
Option type de message (code 53₁₀)
Option identifiant serveur (code : 54₁₀)
dhcprequest
((. '01'H, '01'H, '06'H, '00'H, '9A5AF310'H, '0000'H, '0000'H, '00000000'H, '00000000'H, '00000000'H,
'000050FC27295F0000000000000000'H, '00000000000000000000000000000000000000000000000000000000
'0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000'H, '00000000
'00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
'00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000'H, '63825363'H,
(. '35'H, '01'H, '03'H .), (. '32'H, '04'H, 'AC1000C8'H .), (. '36'H, '04'H, 'AC100001'H .), 'FF'H .)
selection
('000050FC27295F0000000000000000'H, 'AC100001'H)

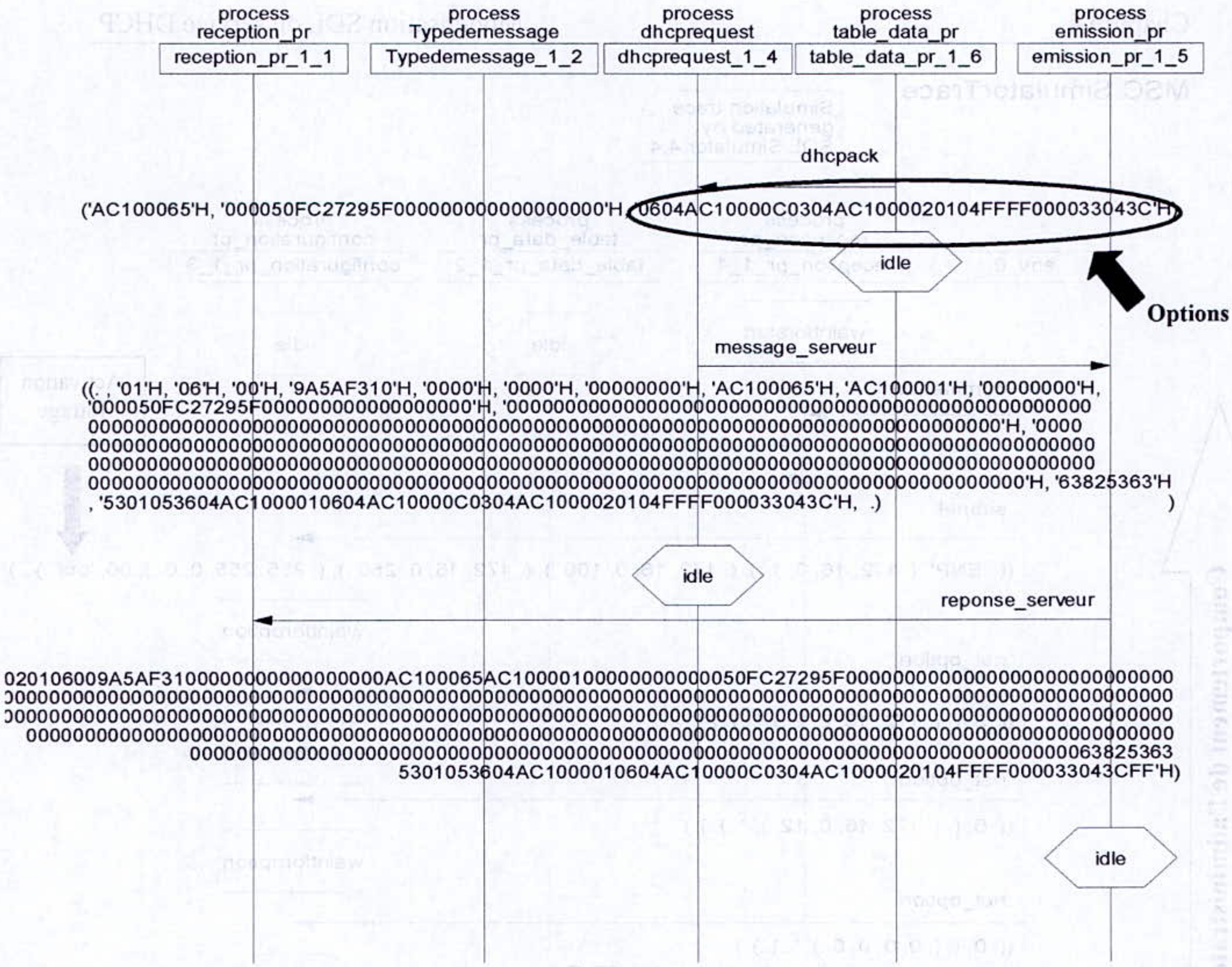


Figure 4.4.17: Traitement du message *DHCPREQUEST*

4.4.2.c Déclaration d'un client au niveau configuration

La figure 4.18 illustre bien les étapes à suivre pour déclarer un client (hôte) appartenant au réseau :

1. Après avoir terminer la configuration du réseau (avec activation du filtrage), on commence par déclarer le client en donnant son nom, son adresse MAC et si besoin, l'adresse IP (fixe) qu'on veut lui attribuer.
2. On peut remarquer qu'après validation de la configuration, le processus *configuration_pr* envoie avec le signal filtrage au processus *recepetion_pr,* une liste des adresses MAC des clients déclarés dans la partie configuration, dans notre cas c'est une seule adresse MAC.

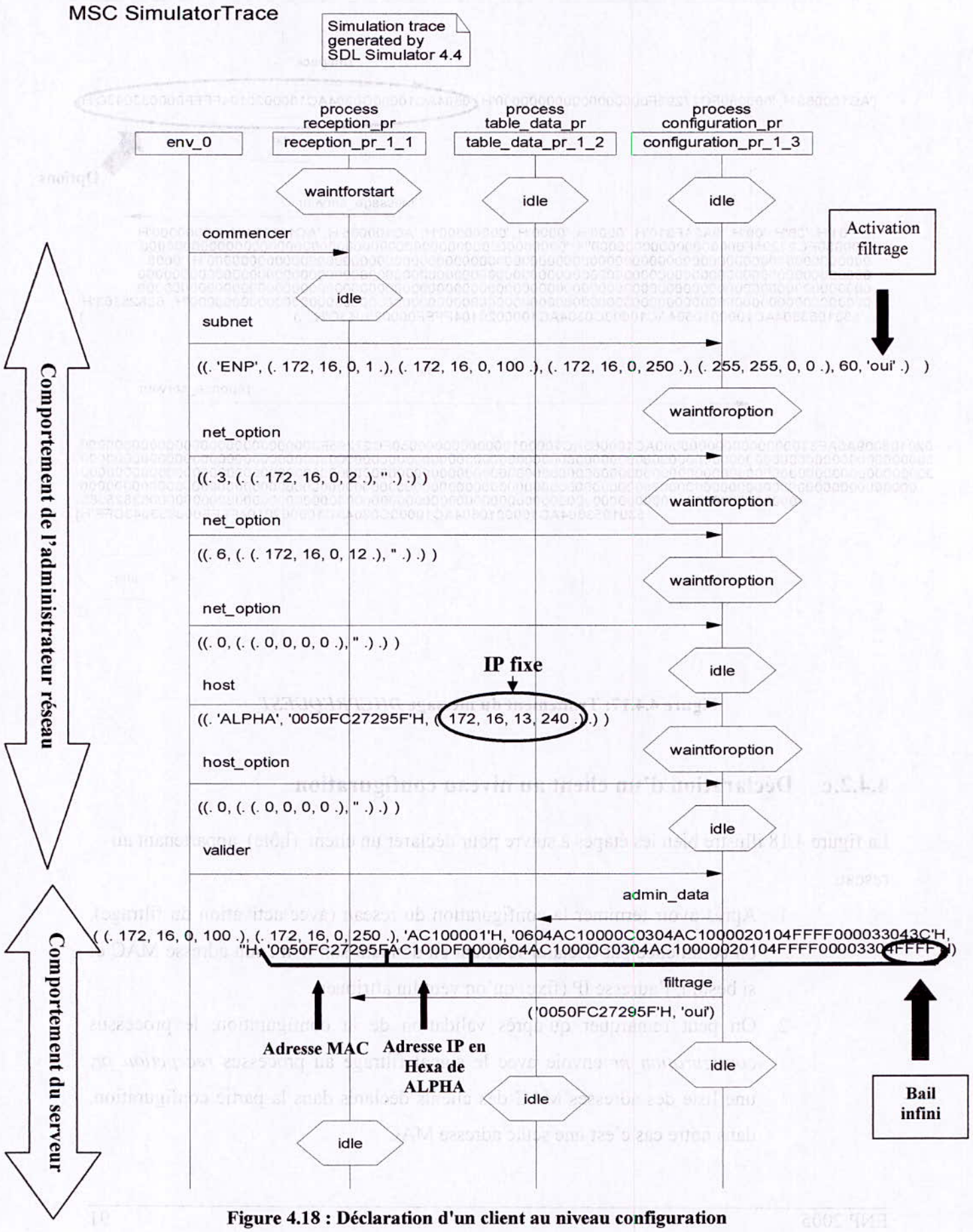


Figure 4.18 : Déclaration d'un client au niveau configuration

4.4.2.d Illustration de la procédure de filtrage

Après avoir déclaré un client (nom : ALPHA), nous allons à présent tester si notre serveur filtre bien les requêtes des clients, une fois cette procédure activée.

Comme le montre la figure 4.19, si notre serveur reçoit une requête d'un client (par exemple un message *DHCPDISCOVER*) non déclaré au niveau de la configuration, il restera muet. En effet, on voit bien que le processus *reception_pr* retourne à l'état *idle* dés qu'il reçoit le message d'un client non déclaré. Car, sont adresse MAC ne correspond pas à l'adresse MAC du client ALPHA.

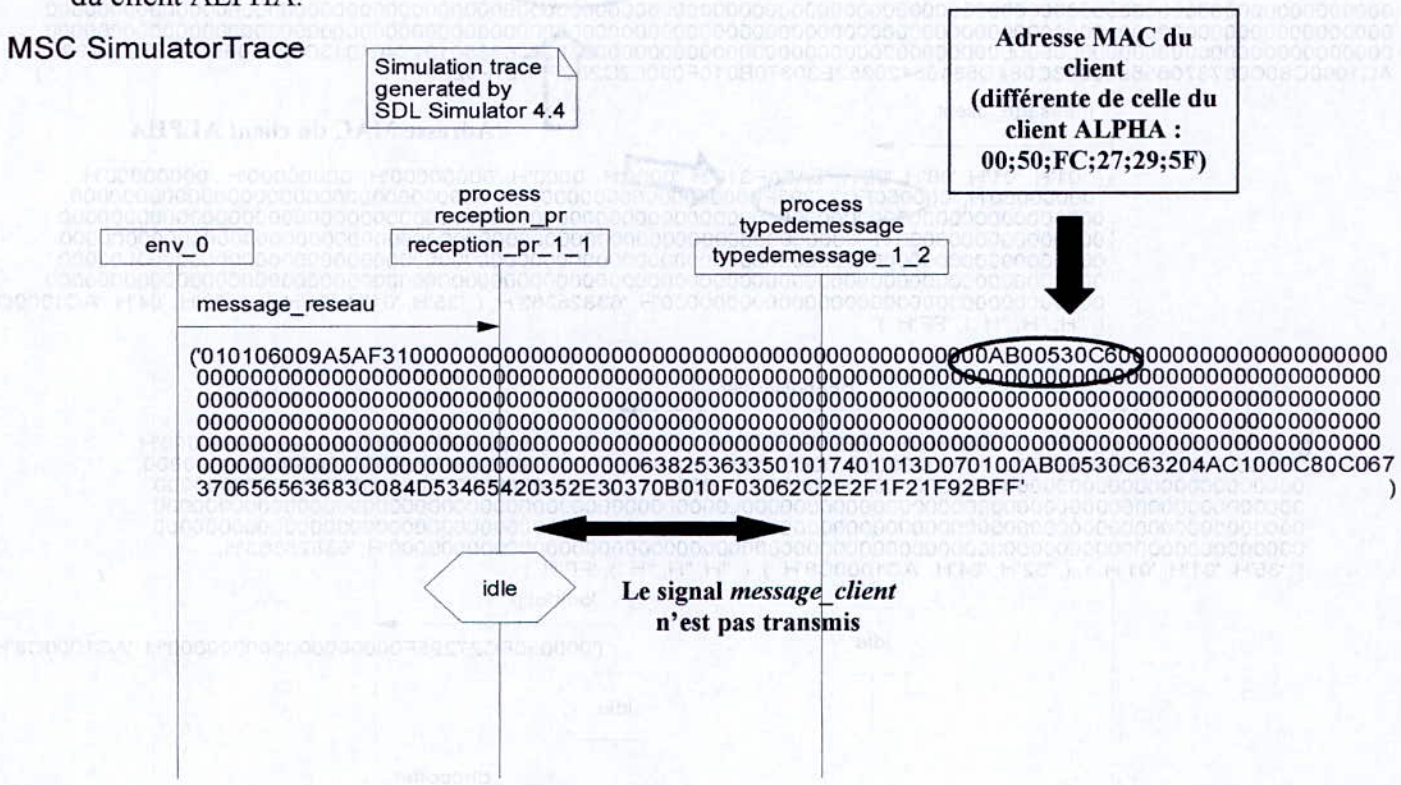
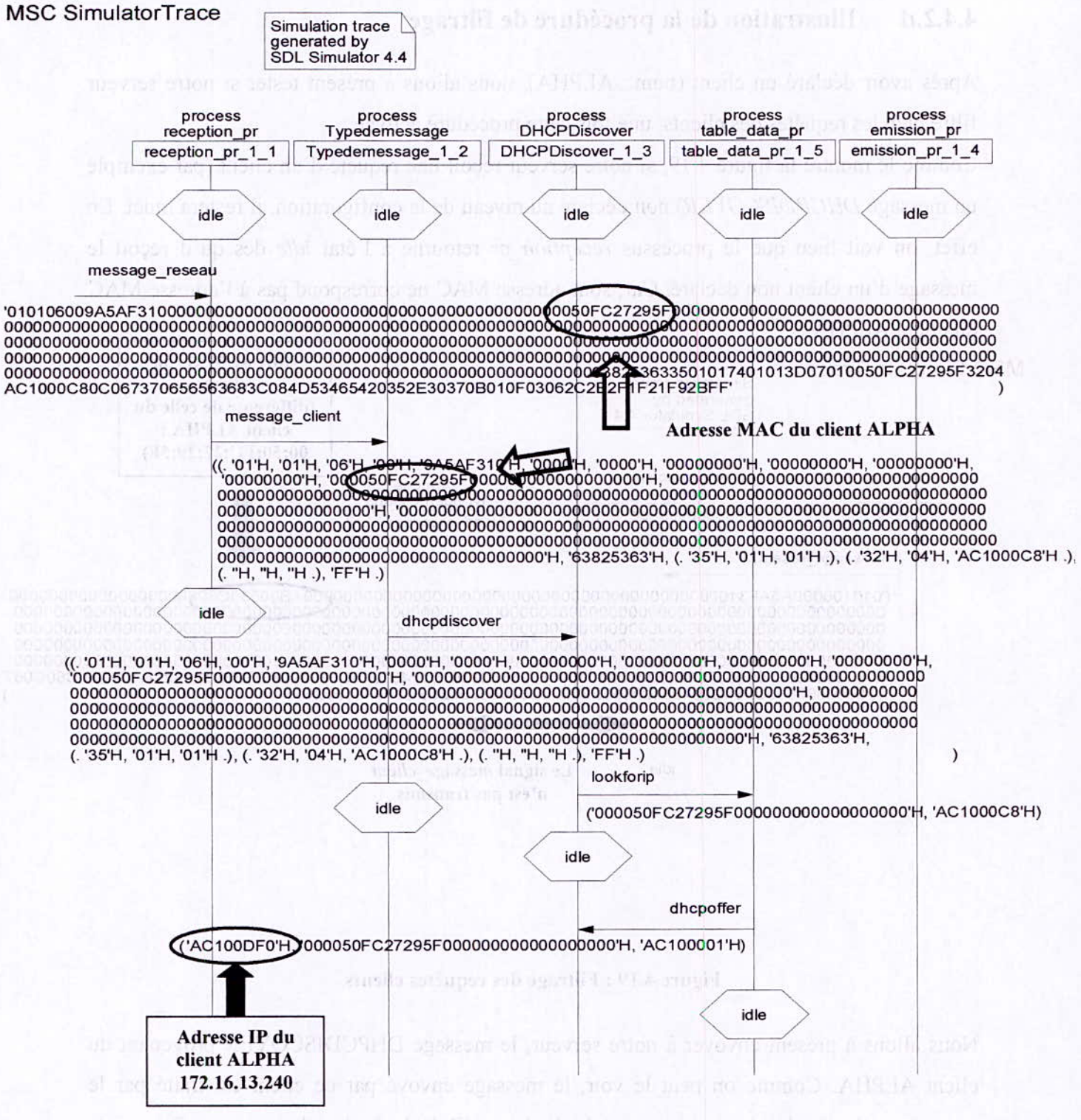


Figure 4.19 : Filtrage des requêtes clients

Nous allons à présent envoyer à notre serveur, le message DHPCDISCOVER provenant du client ALPHA. Comme on peut le voir, le message envoyé par ce client est traité par le serveur, ce dernier lui donne bien entendu, l'adresse IP déclarée dans la partie configuration, et puisque cette adresse est fixe, le bail offert au client sera infini.


MSC SimulatorTrace
Simulation trace generated by SDL Simulator 4.4
process reception_pr
reception_pr_1_1
process Typedemessage
Typedemessage_1_2
process DHCPDiscover
DHCPDiscover_1_3
process table_data_pr
table_data_pr_1_5
process emission_pr
emission_pr_1_4
idle
message_reseau
'010106009A5AF310000000000000000000000000000000000000000050FC27295F
AC1000C80C067370656563683C084D53465420352E30370B010F03062C2E2F1F21F92BFF'
message_client
Adresse MAC du client ALPHA
dhcpdiscover
lookforip
('000050FC27295F00000000000000000'H, 'AC1000C8'H)
dhcpoffer
('AC100DF0'H, '000050FC27295F00000000000000000'H, 'AC100001'H)
Adresse IP du client ALPHA 172.16.13.240

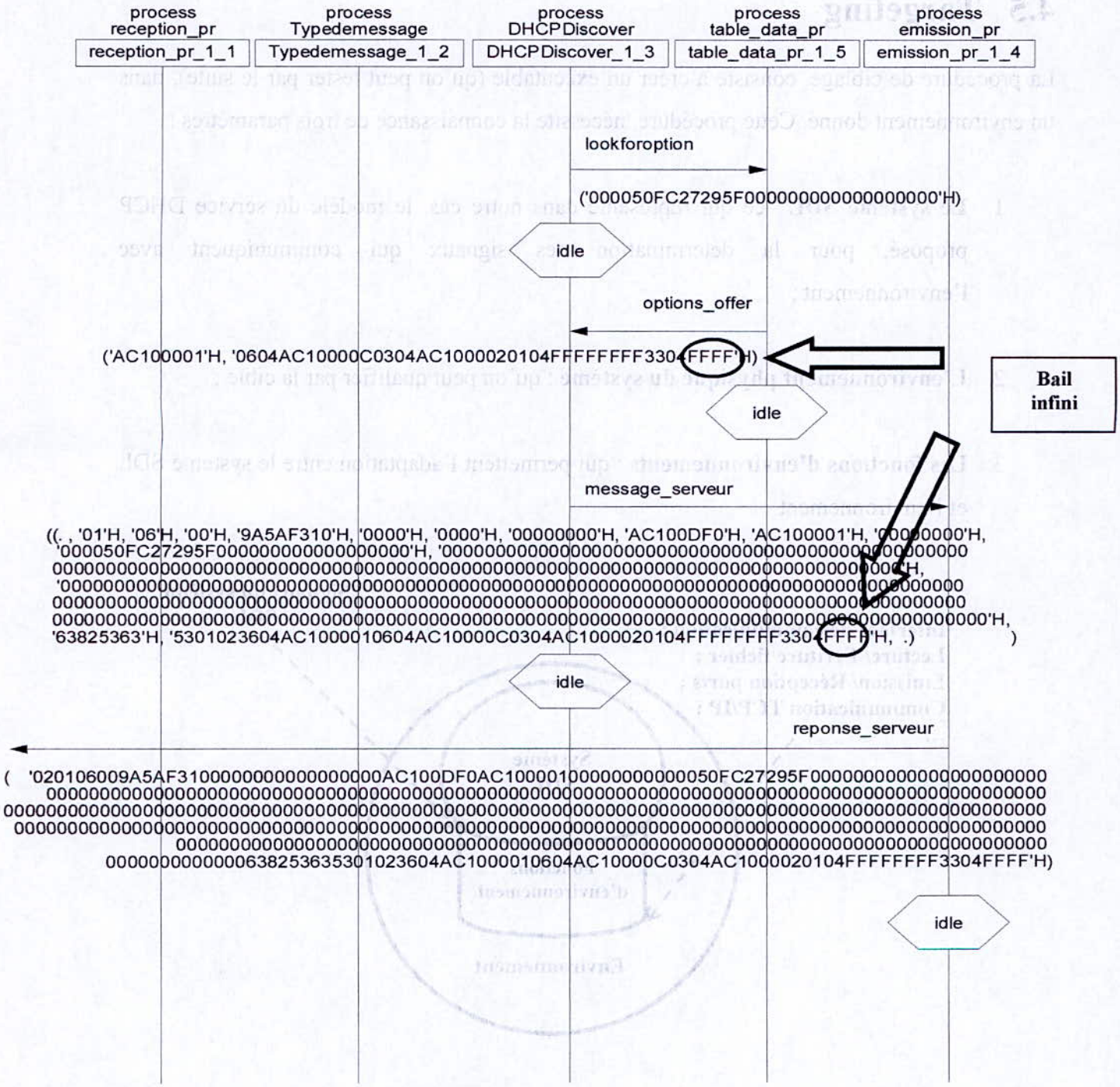


Figure 4.20 : Traitement du message du client ALPHA

4.5 Targeting

La procédure de ciblage, consiste a créer un exécutable (qu'on peut tester par le suite), dans un environnement donné. Cette procédure, nécessite la connaissance de trois paramètres :

1. **Le système SDL** : ce qui représente dans notre cas, le modèle du service DHCP proposé, pour la détermination des signaux qui communiquent avec l'environnement ;

2. **L'environnement physique du système** : qu'on peut qualifier par la cible ;

3. **Les fonctions d'environnements** : qui permettent l'adaptation entre le système SDL et l'environnement.

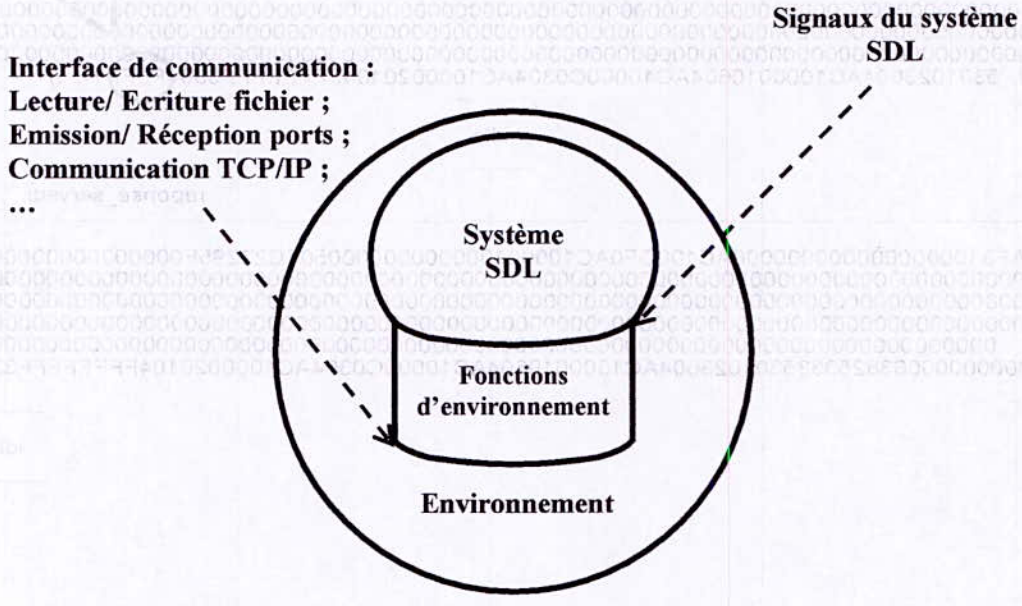


Figure 4.21: Structure d'une application

L'environnement de développement tauSDL nous permet de réaliser le ciblage à travers la l'option ***Targeting Expert.***

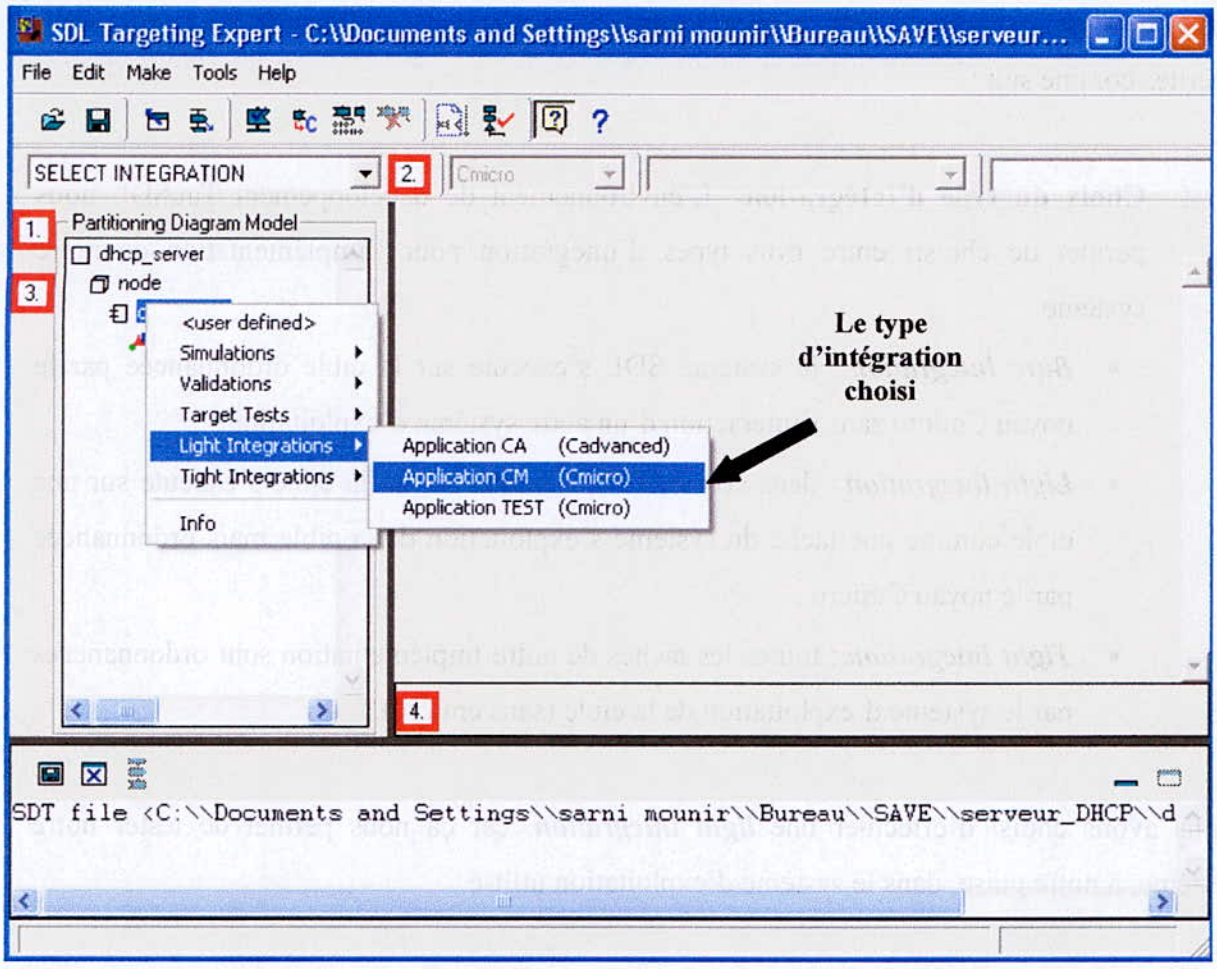


Figure 4.22: Fenêtre tauSDL du targeting

Avec une configuration par défaut, on se propose de faire la réalisation illustrée dans la figure ci-dessous, ainsi à partir d'une machine (PC1), nous allons envoyer les messages DHCP que notre serveur doit traiter, et ce, à travers une liaison série, et réciproquement le serveur DHCP devra y répondre via la même liaison.

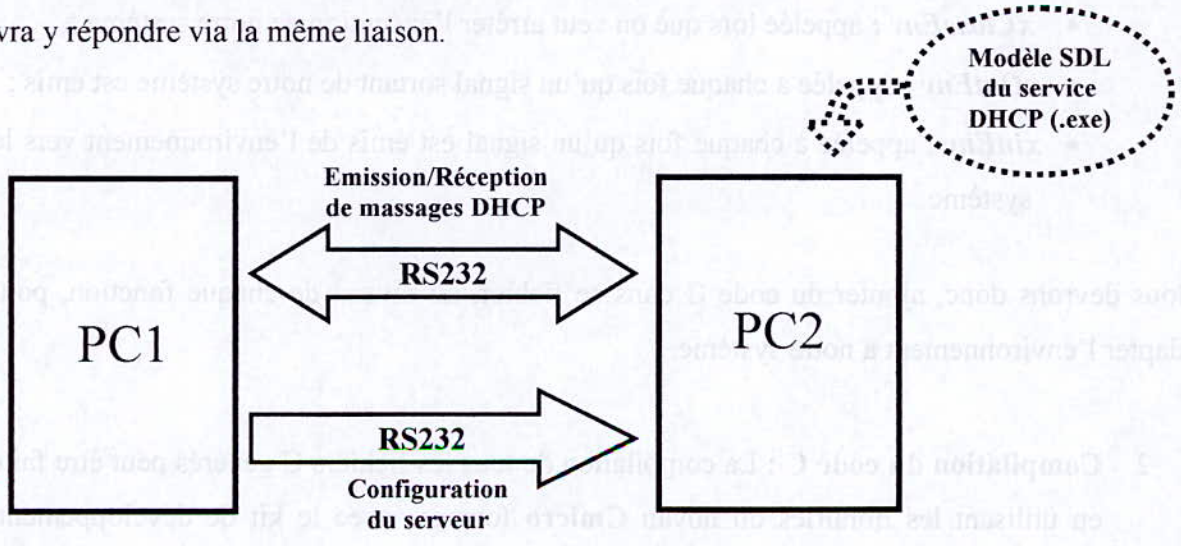


Figure 4.23 : Illustration de la communication établie en PC1 et PC2

Les étapes que nous avons suivies pour réaliser la communication ''client''-''serveur'' sont décrites comme suit :

1. **Choix du type d'intégration** : L'environnement de développement TauSDL nous permet de choisir entre trois types d'intégration pour l'implémentation de notre système :
   - ***Bare integration*** : le système SDL s'exécute sur la cible ordonnancée par le noyau Cmicro sans l'intéraction d'un autre système d'exploitation.
   - ***Ligth integration*** : dans ce cas, l'implémentation sur la cible s'exécute sur une cible comme une tâche du système s'exploitation de la cible mais ordonnancée par le noyau Cmicro .
   - ***Tight integration*** : toutes les tâches de notre implémentation sont ordonnancées par le système d'exploitation de la cible (sans cmicro).

Nous avons choisi d'effectuer une ***light integration***, car ça nous permet de tester notre système, à notre guise, dans le système d'exploitation utilisé.

2. **Edition du fichier d'environnement** : Après avoir choisi le type d'intégration, des fichiers C sont générés automatiquement, traduction de notre modélisation du service DHCP. Parmi ces fichiers, il se trouve un (le fichier ***env.c***), dans lequel les fonctions environnements sont définies, et qui sont :
   - ***xInitEnv*** : appelée lors de l'initialisation de notre système ;
   - ***xCloseEnv :*** appelée lors que on veut arrêter l'exécution de notre système ;
   - ***xOutEnv*** : appelée à chaque fois qu'un signal sortant de notre système est émis ;
   - ***xinEnv*** : appelée à chaque fois qu'un signal est émis de l'environnement vers le système.

Nous devrons donc, ajouter du code C dans ce fichier, au niveau de chaque fonction, pour adapter l'environnement à notre système.

2. **Compilation du code C :** La compilation de tous les fichiers C générés peut être faite en utilisant les librairies du noyau **Cmicro** fournies avec le kit de développement, nécessaire à celle-ci. Ce qui produira un fichier exécutable prêt à l'emploi, qu'on peut

éventuellement tester, a travers le ***Targert's Tester*** de l'environnement de développement TauSDL.

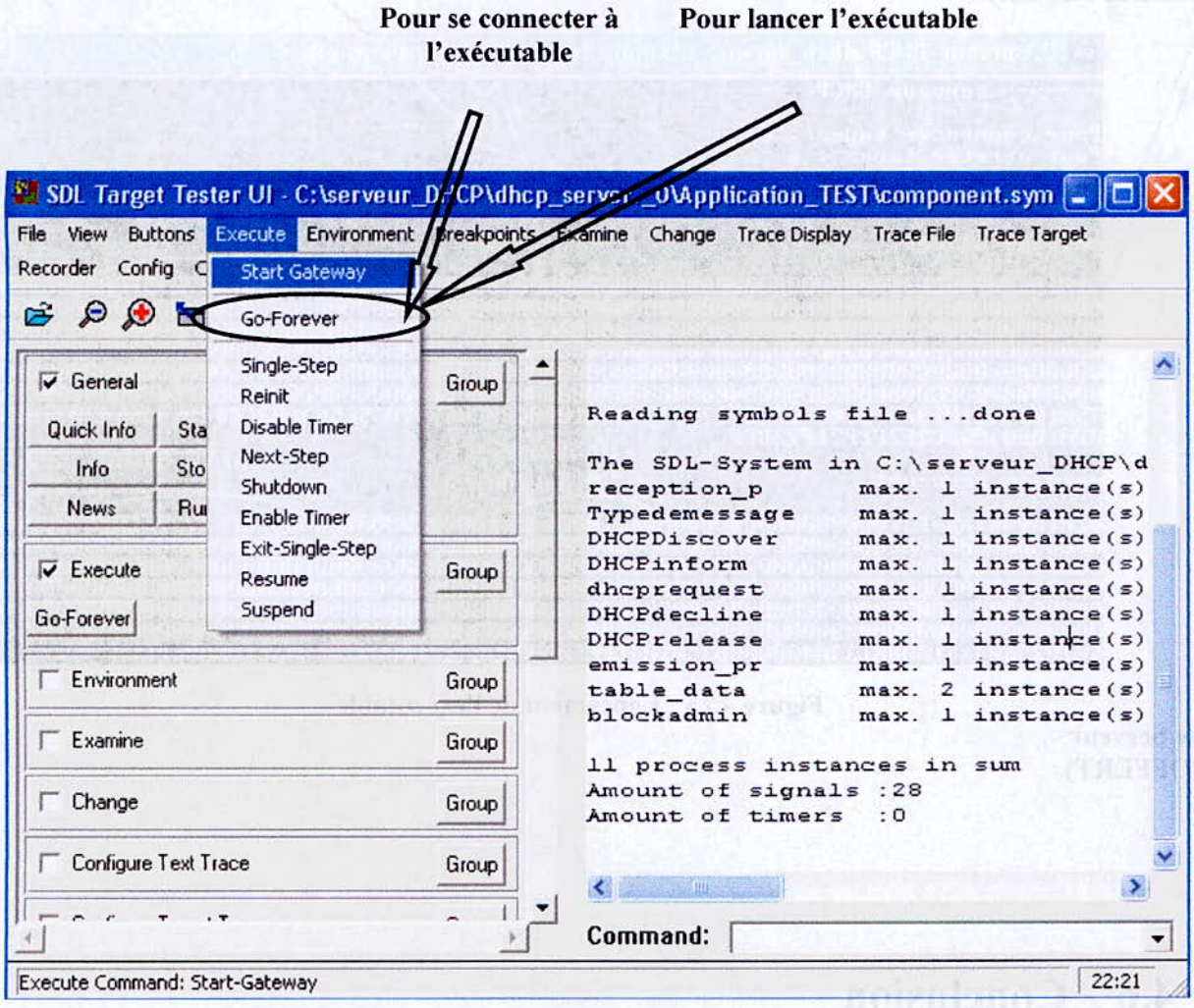


Figure 4.24 : Fenêtre *Target Tester* de TauSDL

La figure suivante représente le terminal affiché, après le lancement de notre exécutable, et après que le PC1 ait envoyé le message client :

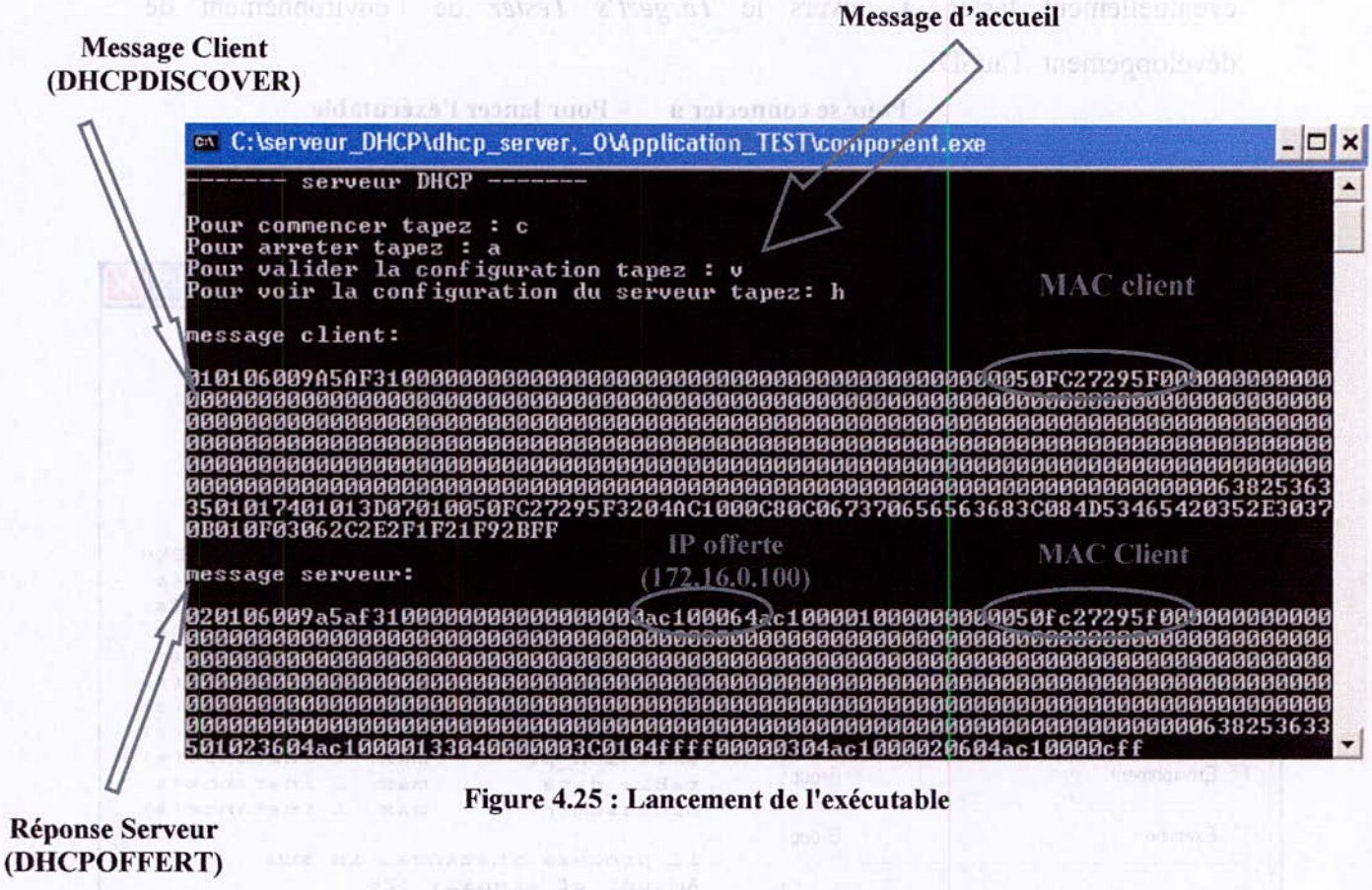


Figure 4.25 : Lancement de l'exécutable

4.6 Conclusion

Bien que le modèle du service DHCP présenté dans ce chapitre se restreint à la gestion d'un seul sous-réseau en offrant un seul bail , à travers les étapes de conception suivies, l'obtention d'un exécutable à partir de la description SDL du DHCP, a permis de valider le modèle proposé. Ainsi, dans un réseau local, où l'aspect sécurité n'est pas important, le code C généré est tout a fait exploitable.

Conclusion et Perspectives

Au cours de ce projet un modèle du serveur DHCP a été établi, à l'aide du langage SDL. L'environnement TauSDL a permis de simuler son comportement, et d'obtenir également un exécutable grâce auquel nous avons pu tester notre système et le valider.

Toutefois, le serveur DHCP proposé ne gère qu'un seul réseau et n'offre qu'un seul bail pour tous les clients. Ce modèle demeure donc perfectible.

La méthodologie de développement suivie pour arriver au système sur puce, à savoir, la modélisation du protocole avec un langage de description formelle, la validation du modèle et enfin l'utilisation du code C généré comme application logicielle s'exécutant sur un environnement matériel, est une procédure rapide et simple, qui peut être appliquée à tout système de communication.

Lors de l'implémentation, la carte choisie est une carte à base de circuit FPGA (spartan3), et la plate-forme de développement à utilisée est EDK (Embedded Developpement Kit) de Xilinx [11].

Grâce au principe du Co-Design, cet environnement nous permet d'une part de développer le hardware (configuration du cœur de processeur logiciel MicroBlaze et de ces périphériques) et d'autre part, de développer le software (les applications), puis de générer un même Bitstream qui contient à la fois le matériel et le logiciel et qui sera chargé sur le circuit FPGA. L'application logicielle nécessaire est le code C généré à partir de la modélisation en SDL. Cependant, cet environnement fait appel au compilateur GNU (Gcc) de Linux, différemment de l'environnement Tau SDL qui lui utilise celui de microsoft.
Donc un problème d'adaptation de compilateur s'est posé.

Comme perspectives, citons donc, la possibilité d'exploiter le code C généré à partir de notre modélisation sur n'importe quelle plate-forme de Co-design.

Aussi, si on revenait au modèle en couches, en plus de notre modélisation, qui se situe dans la couche application du modèle TCP/IP, comme décrit dans la chapitre1, il faut intégrer les fonctionnalités des couches inférieures, soit d'une manière logicielle, exemple la modélisation de toutes ces couches avec un langage formel tel que SDL, ou bien d'une manière matérielle jumelée au logiciel, afin que le circuit soit fonctionnel sur un réseau.

Grâce à ce projet, nous avons pu acquérir des connaissances précieuses dans une branche très moderne de la communication, soit les réseaux TCP/IP.
Nous avons pu également avoir accès à une infime partie des immenses potentialités et perspectives qu'offre la conception sur puce (SOC), dans un monde caractérisé par une course effrénée vers la miniaturisation et la vitesse.
Nous n'avons qu'un seul regret, c'est de n'avoir pu découvrir plus tôt au cours de notre cursus, le langage formel SDL compte tenu de toute sa contribution au domaine des télécommunications.

A Annexe : Symboles graphiques de SDL

La grammaire graphique est décrite dans la norme ITU-T Z.100. Nous présentons, dans les figures A.1 et A.2, les différents symboles graphiques pour permettre la compréhension de système SDL décrit graphiquement.

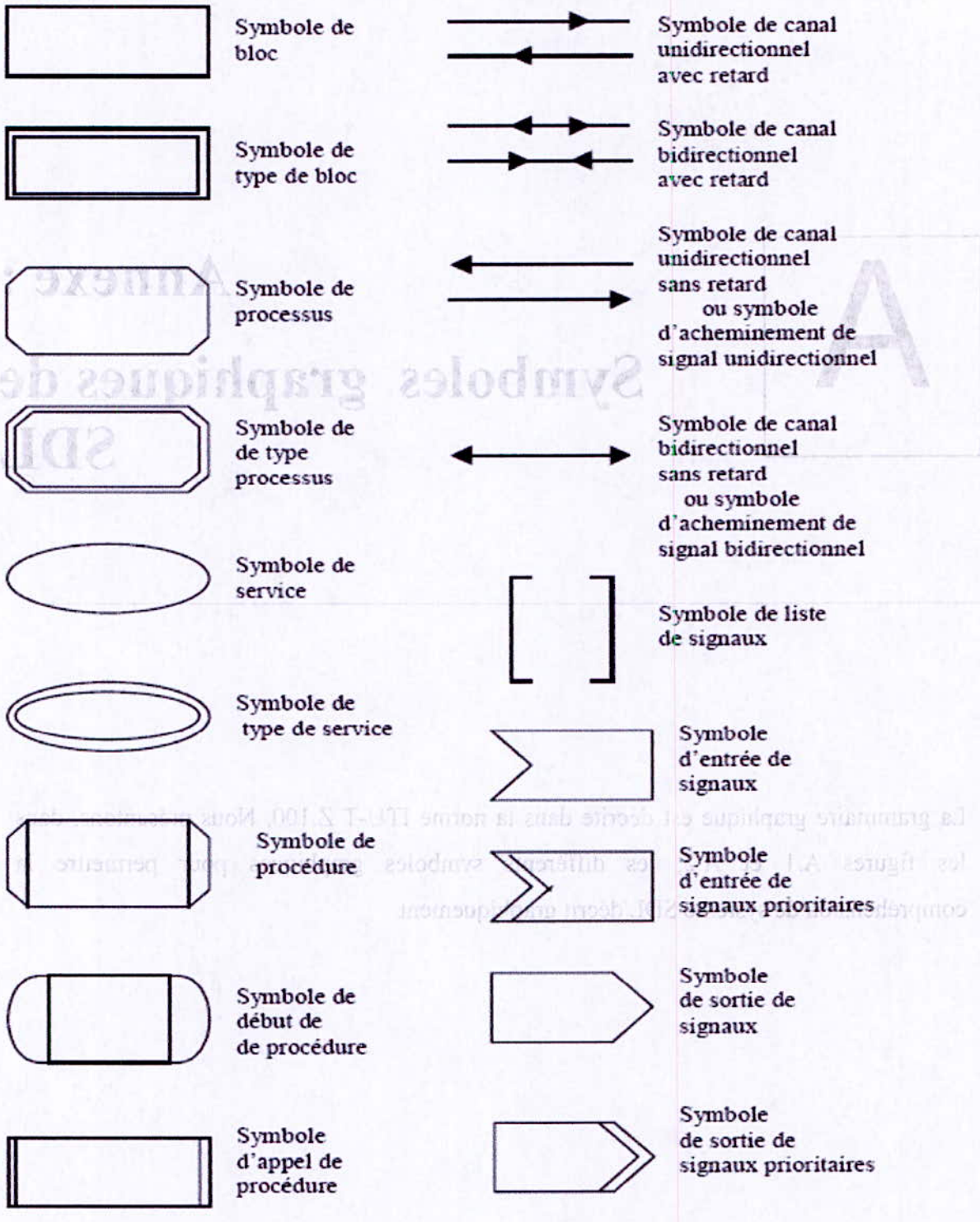


Figure A.1: Symboles graphiques de SDL

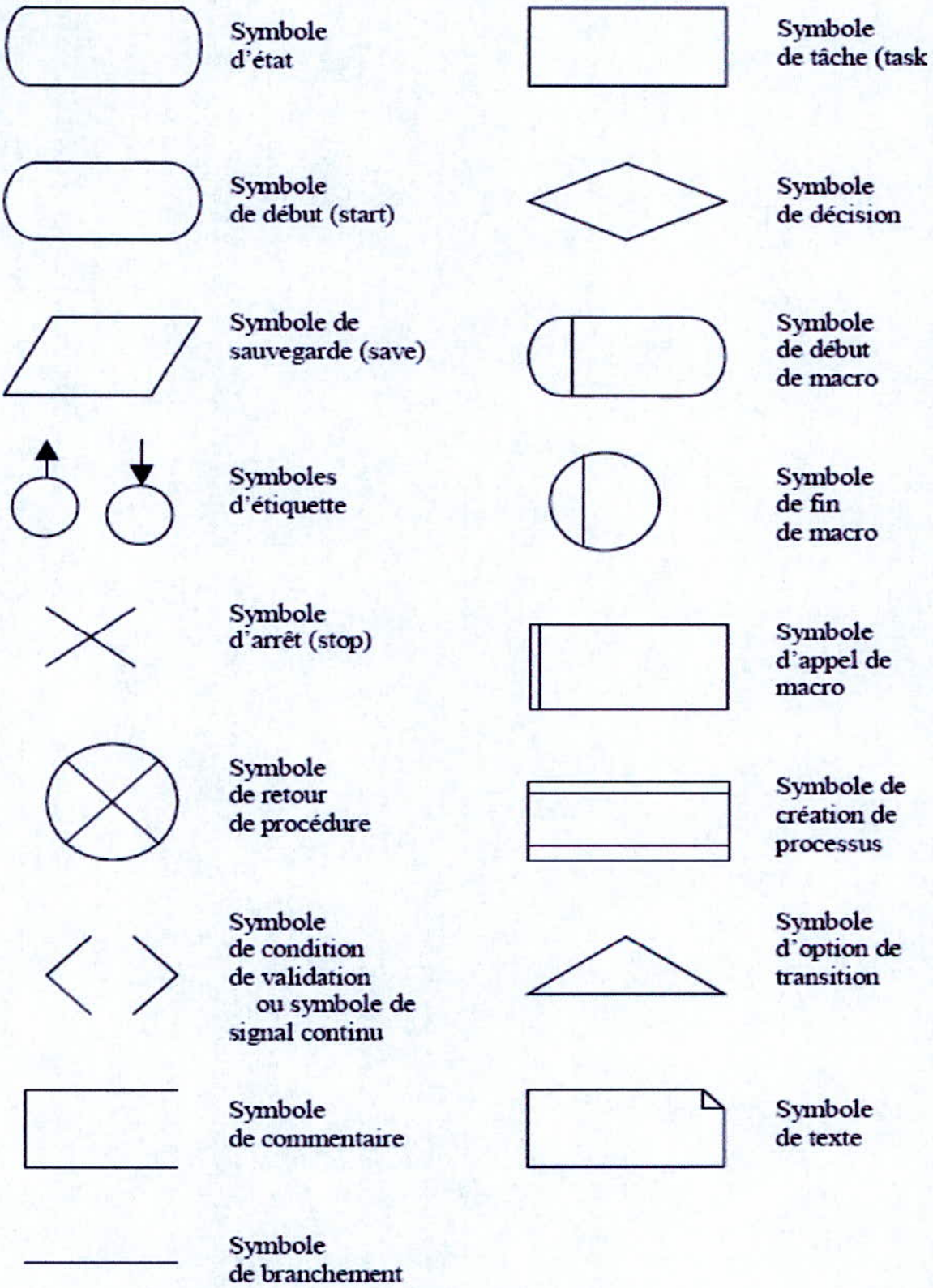
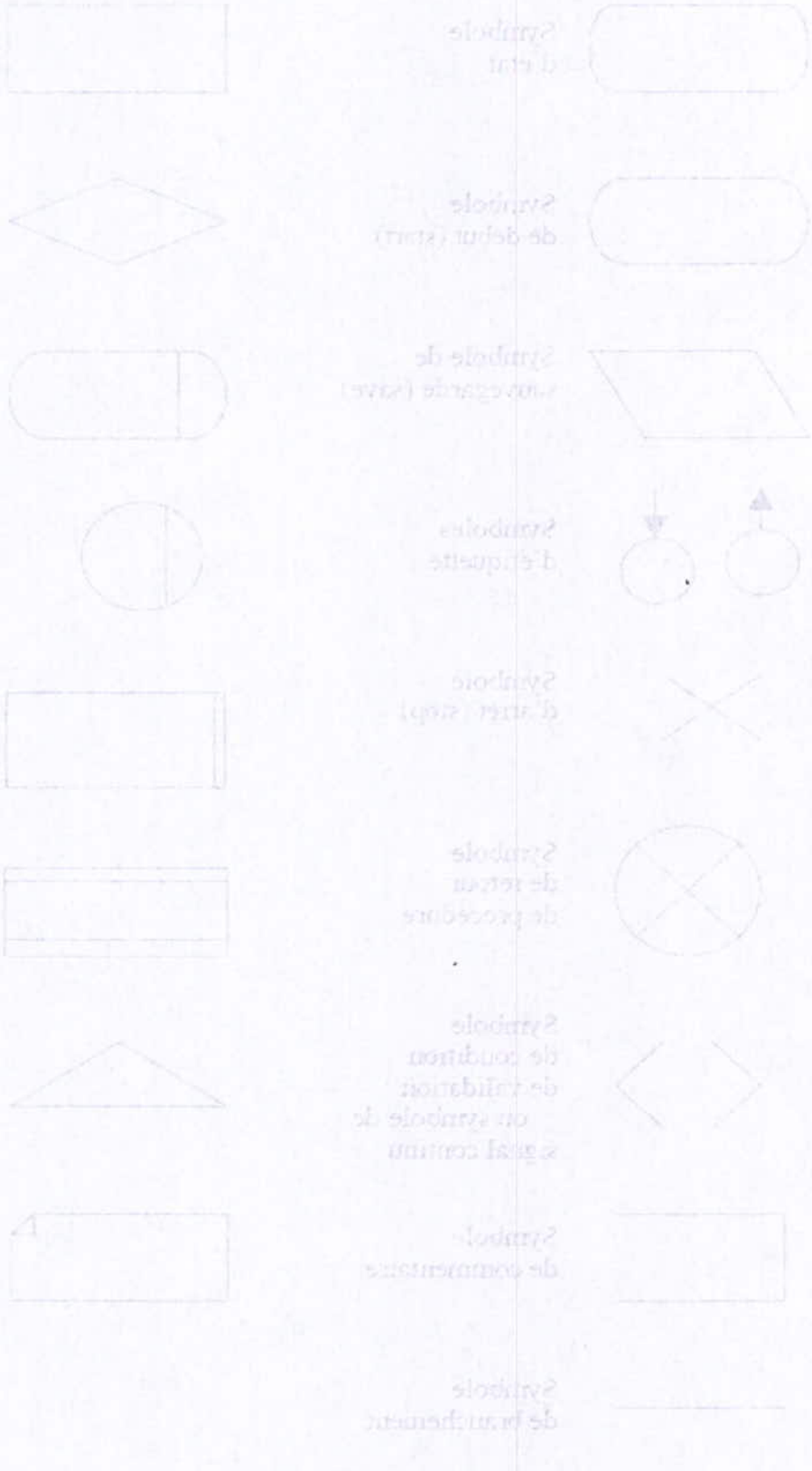


Figure A.2 : Symboles graphique de SDL



B

Annexe : Description SDL du service DHCP

Les figures suivantes représentent la description de tout les processus du modèle du service DHCP proposé dans le chapitre 4. Nous commencerons par donner les figures concernant le bloc *reception,* en suite le bloc *emission*, puis le bloc *configuration* et enfin nous terminerons pa+r la description des processus du bloc *table_data* et *Traitement.*

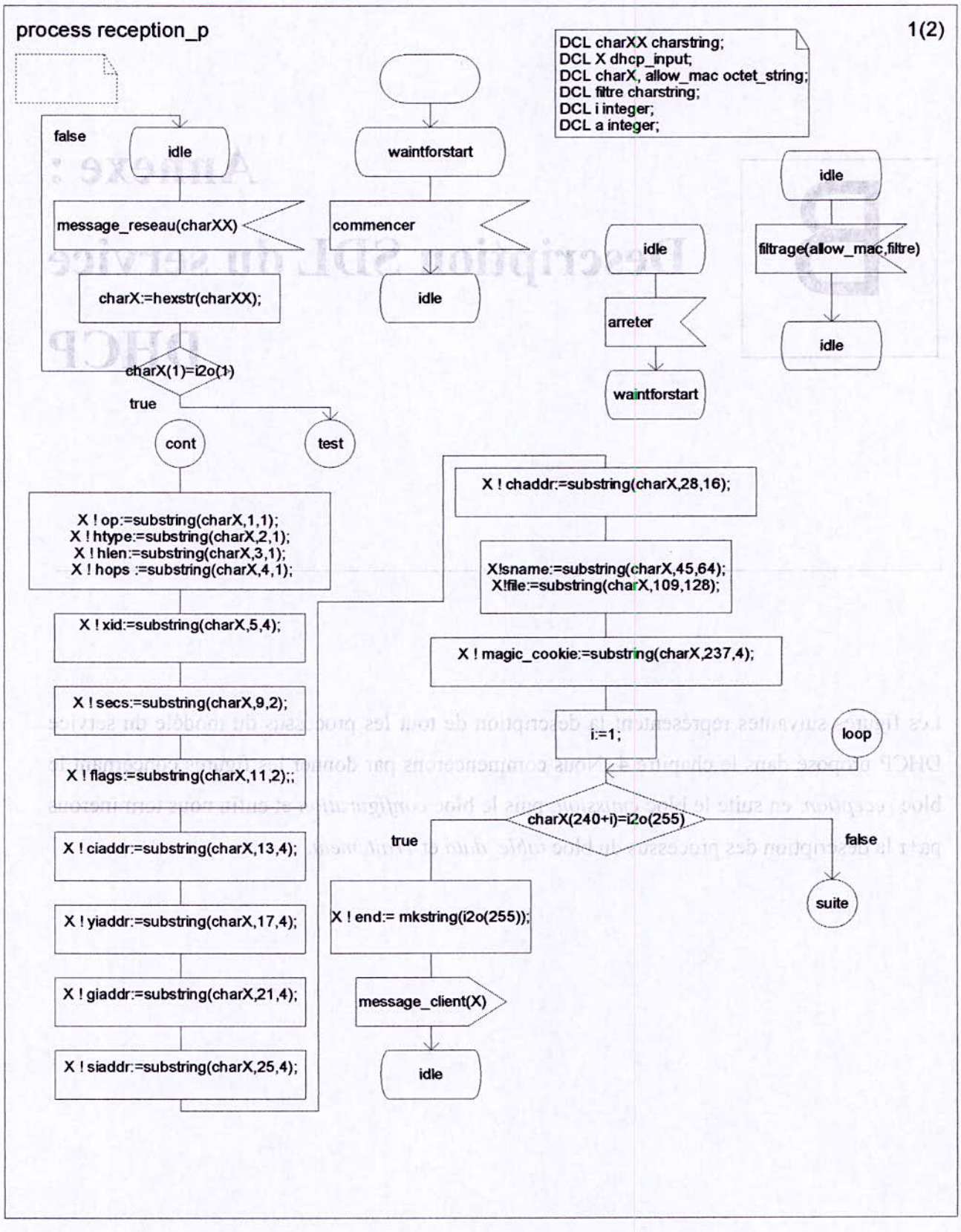


Figure B.1 : Processus *reception* (page 1/2)

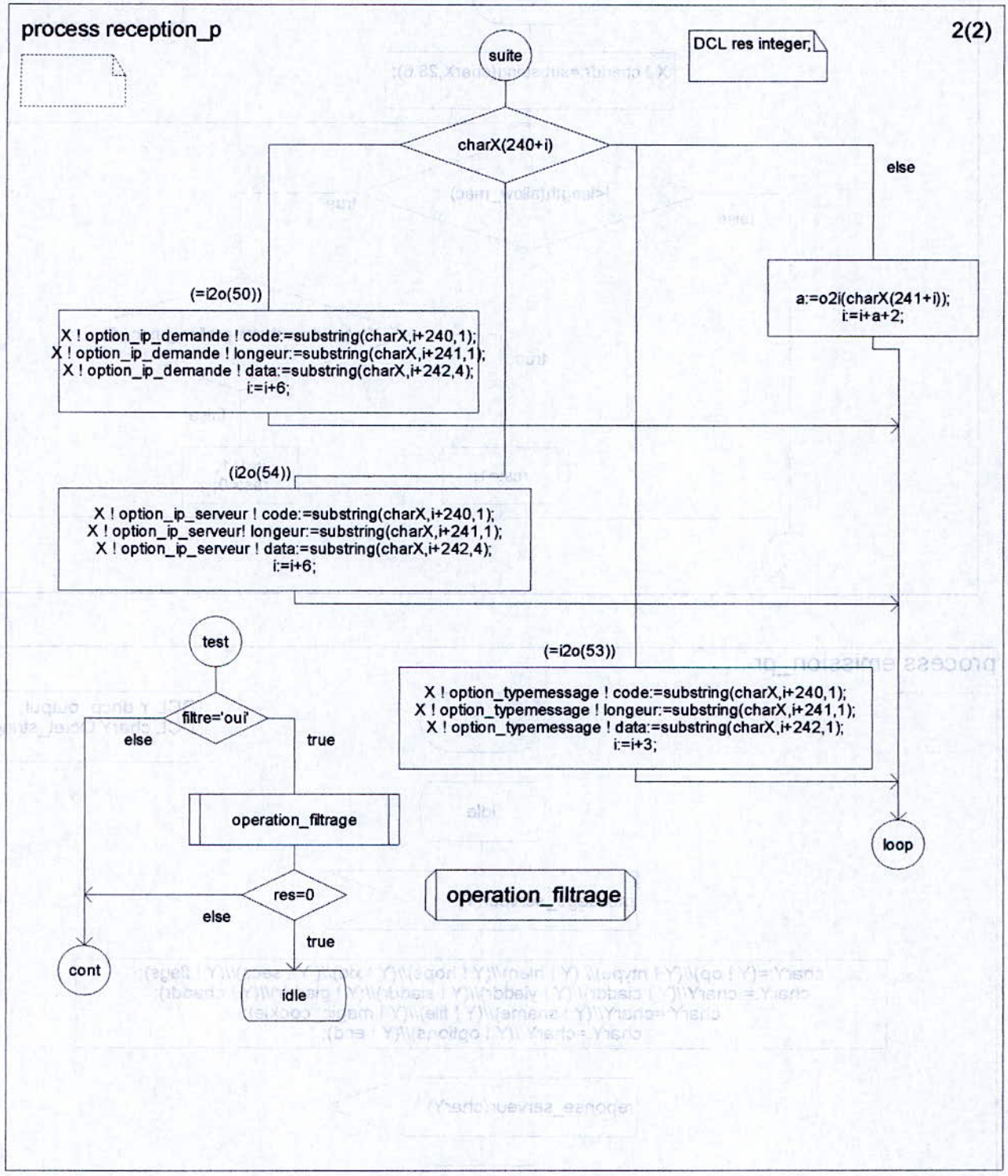
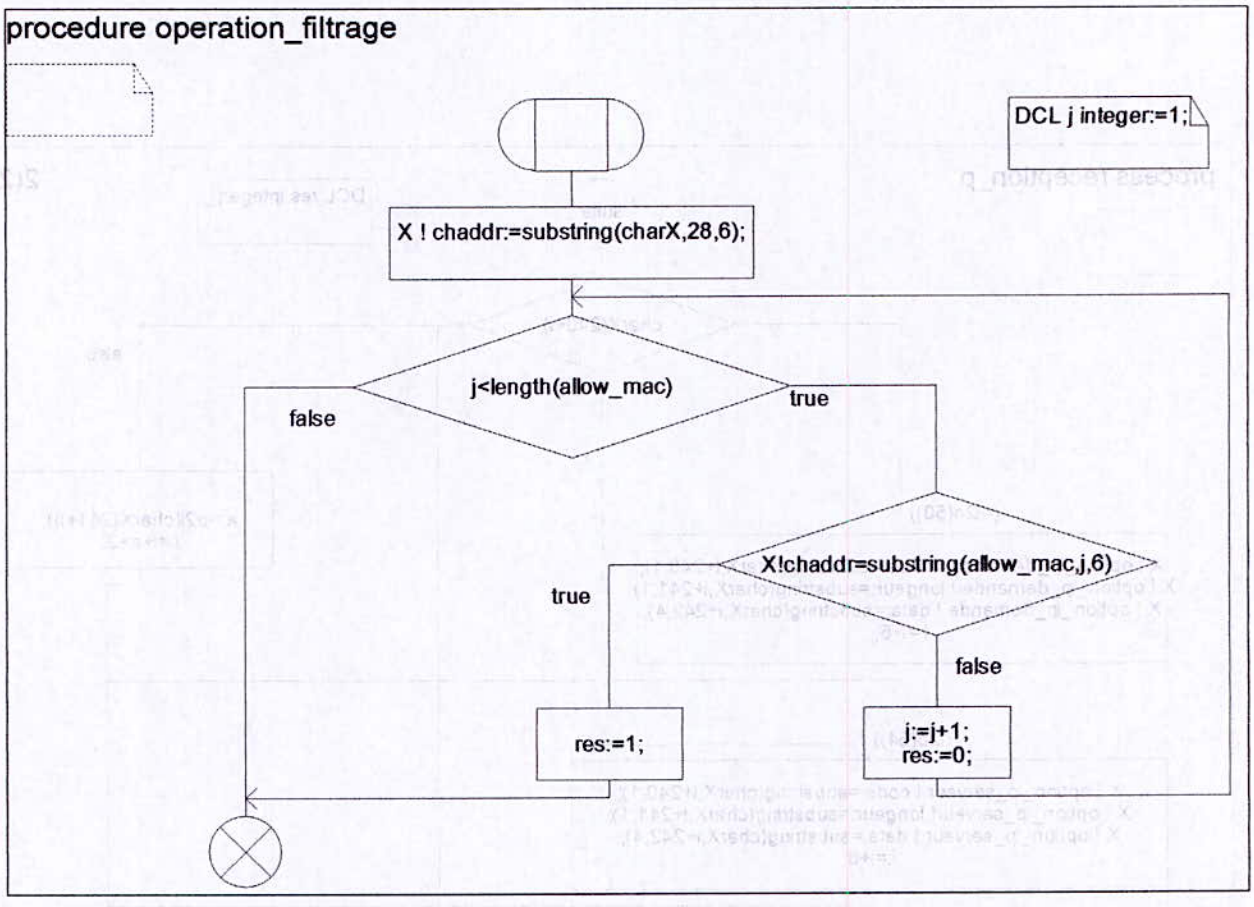


Figure B.2 : Processus *reception_pr* (page 2/2)



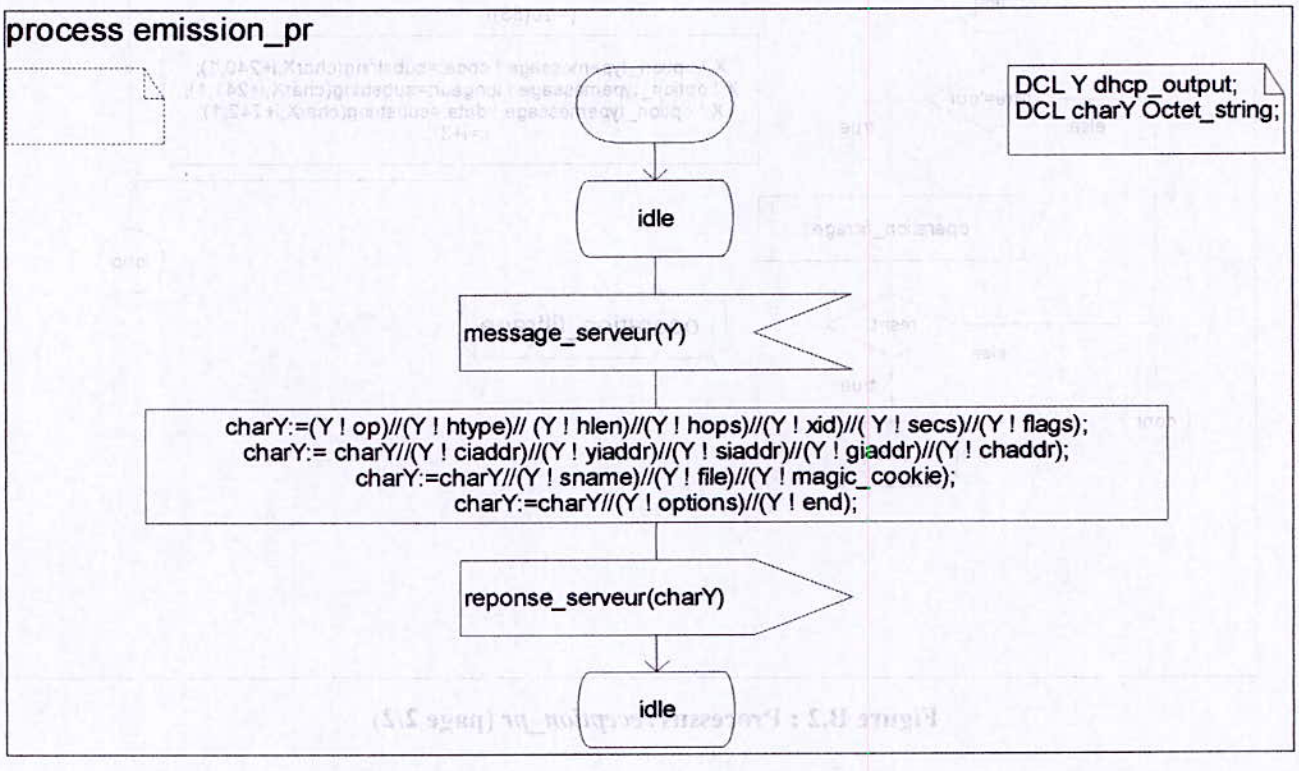


Figure B.4 : Processus *emission_pr* (page 1/1)

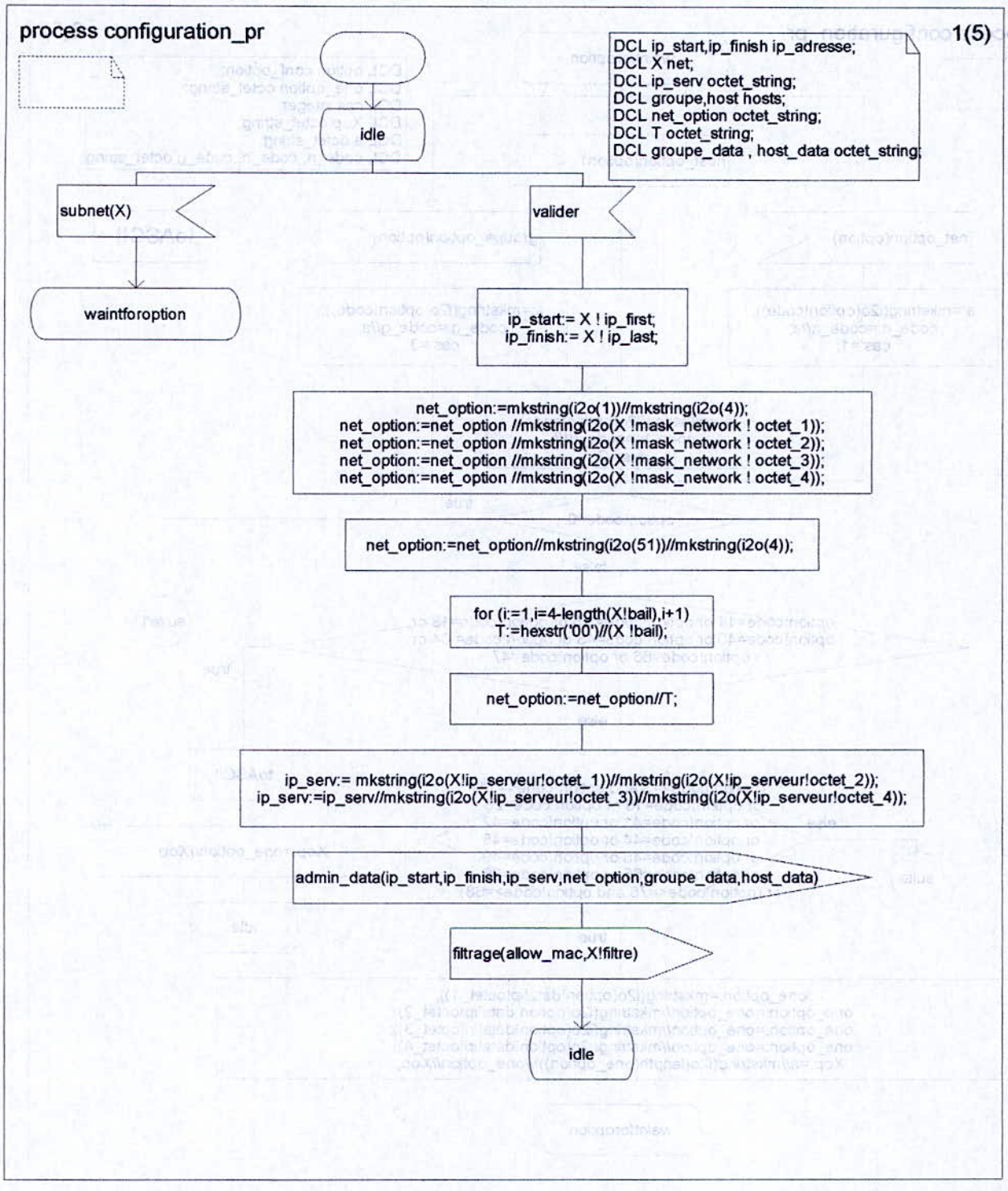


Figure B.5 : Processus *configuration_pr* (page 1/5)

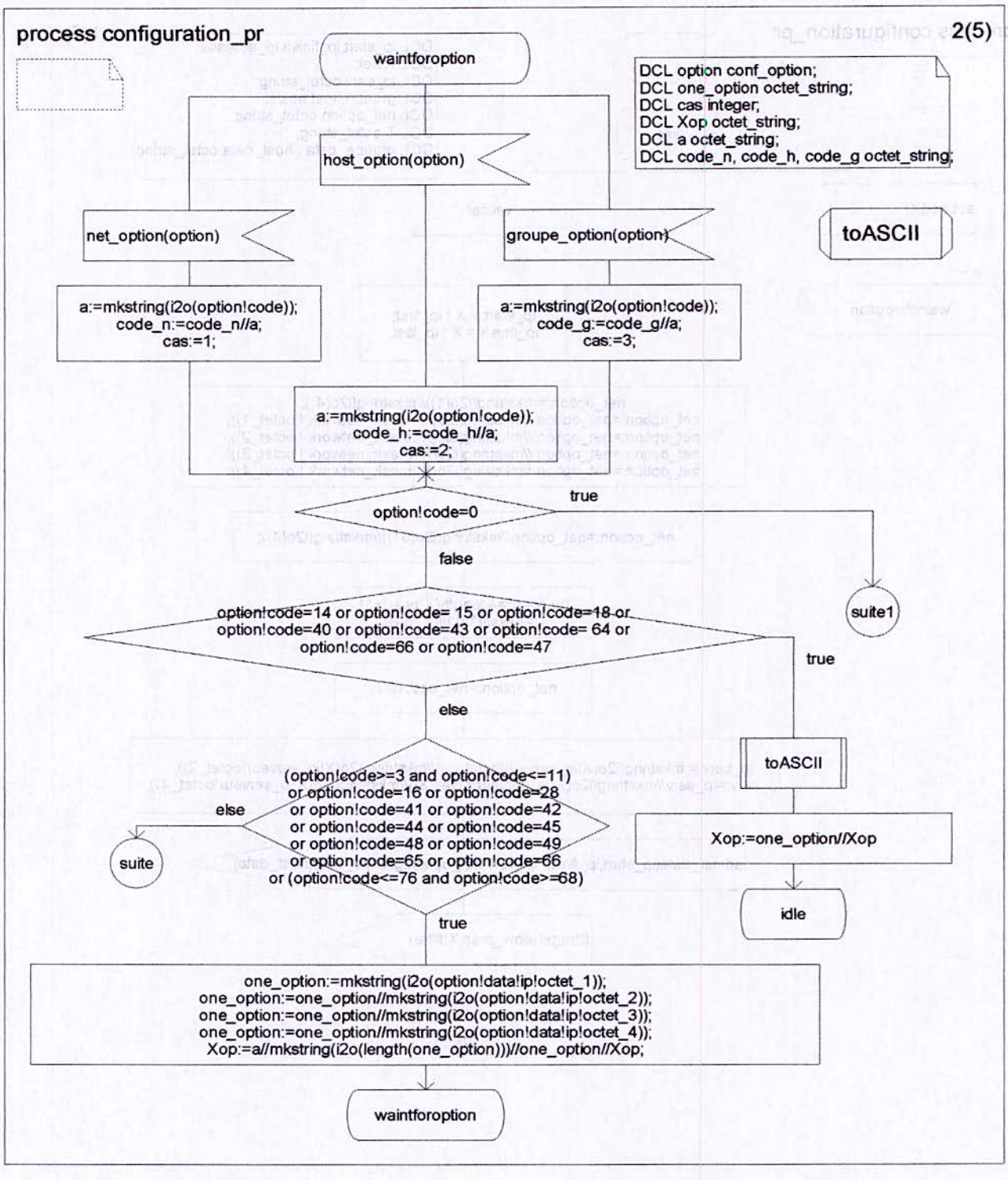


Figure B.6 : Processus *configuration_pr* (page 2/5)

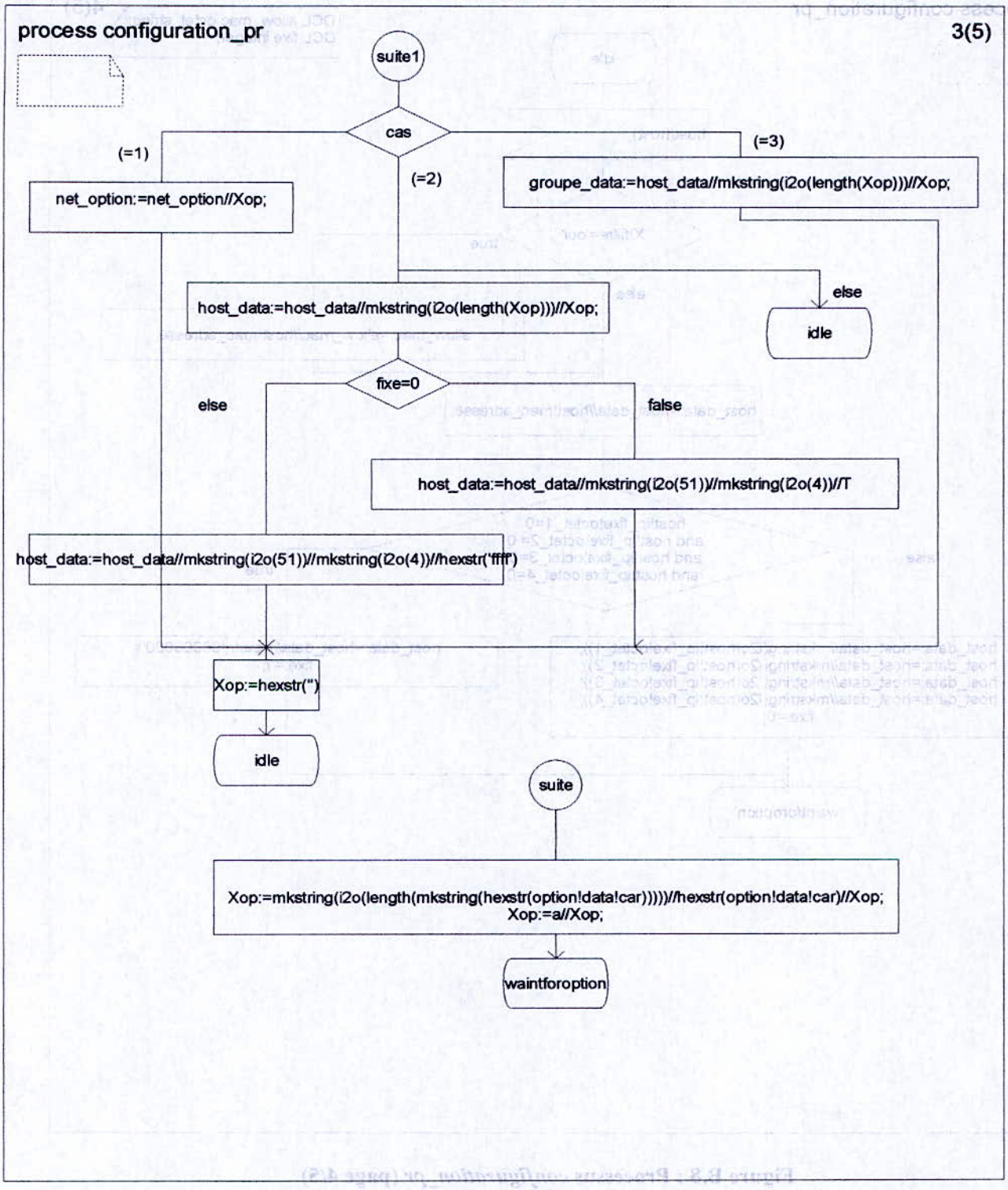


Figure B.7: Processus configuration_pr (page 3/5)

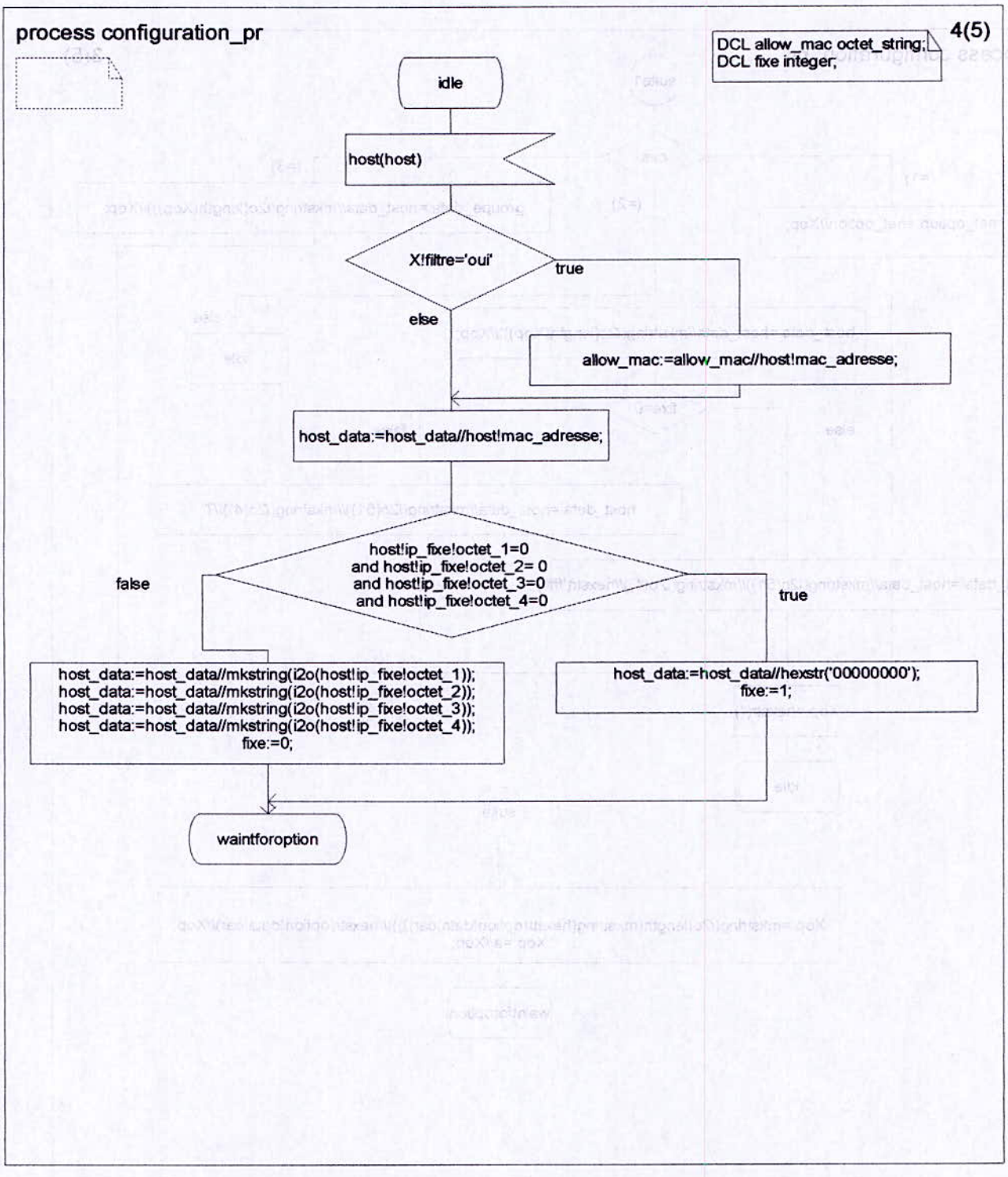


Figure B.8 : Processus *configuration_pr* (page 4/5)

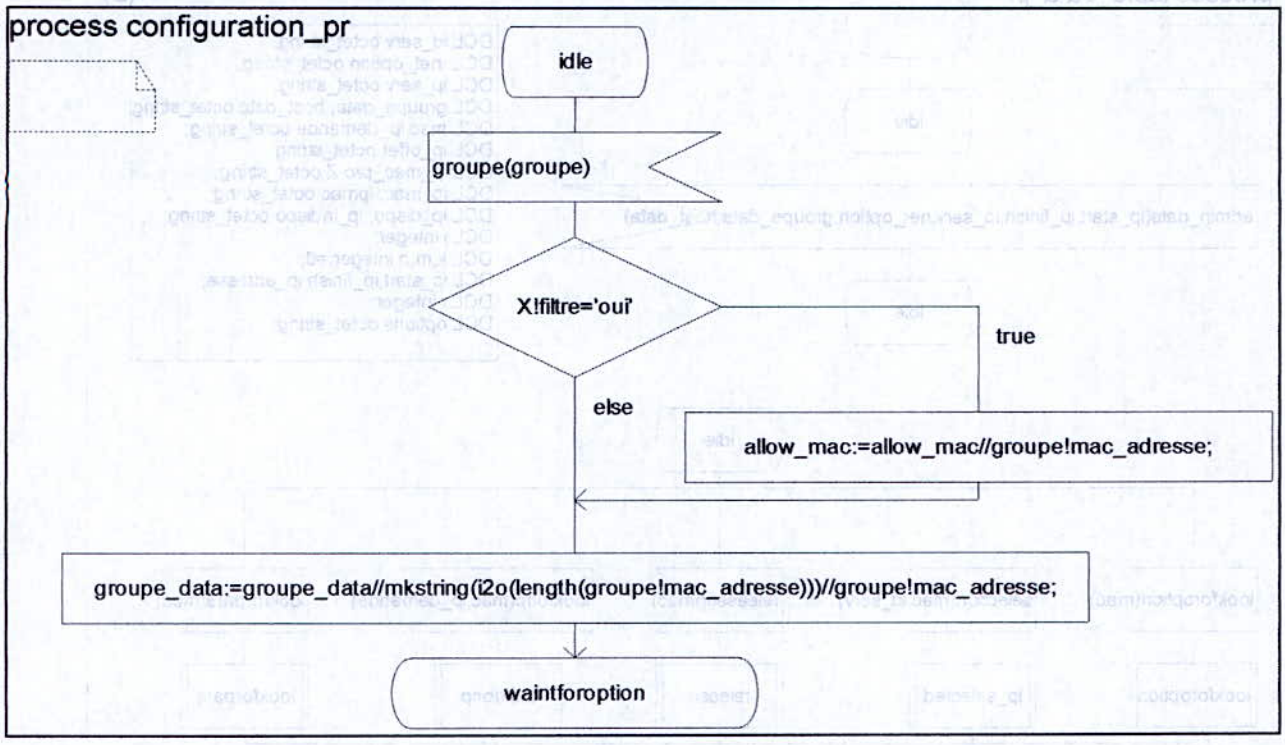


Figure B.9 : Processus *configuration_pr* (page 5/5)

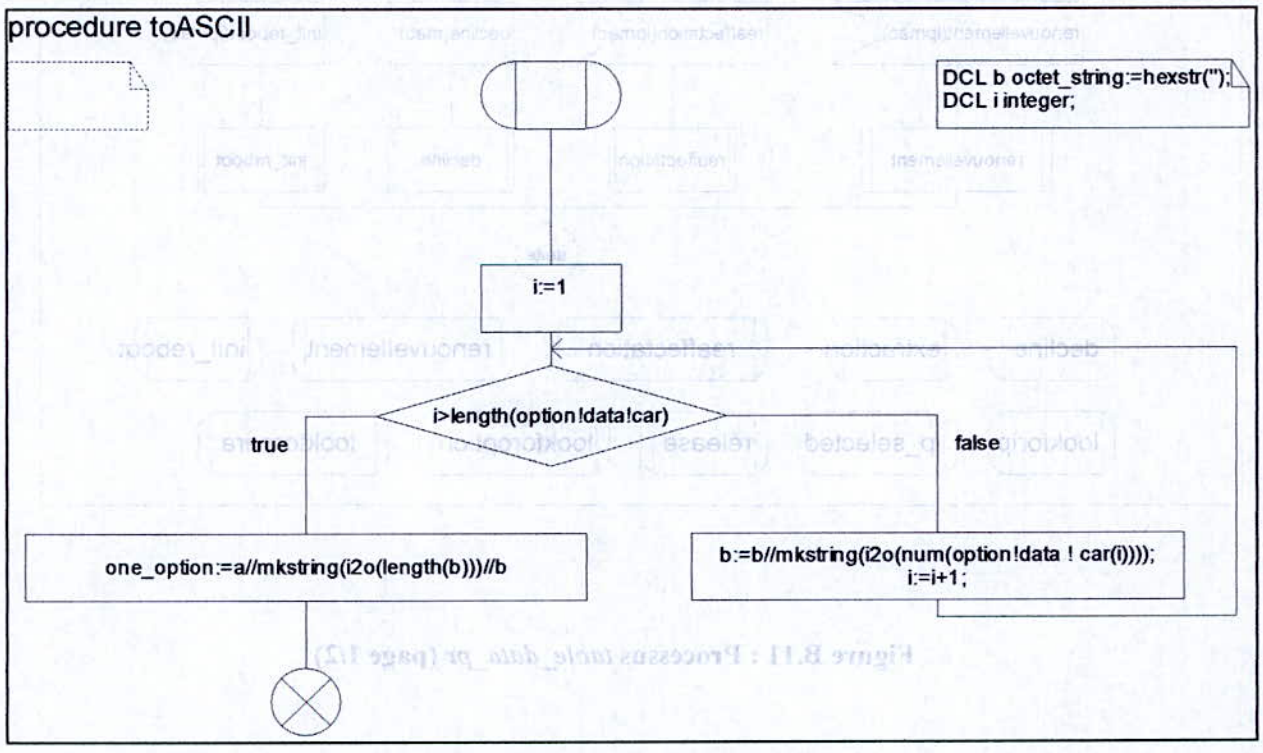


Figure B.10 : Procédure *toASCII*

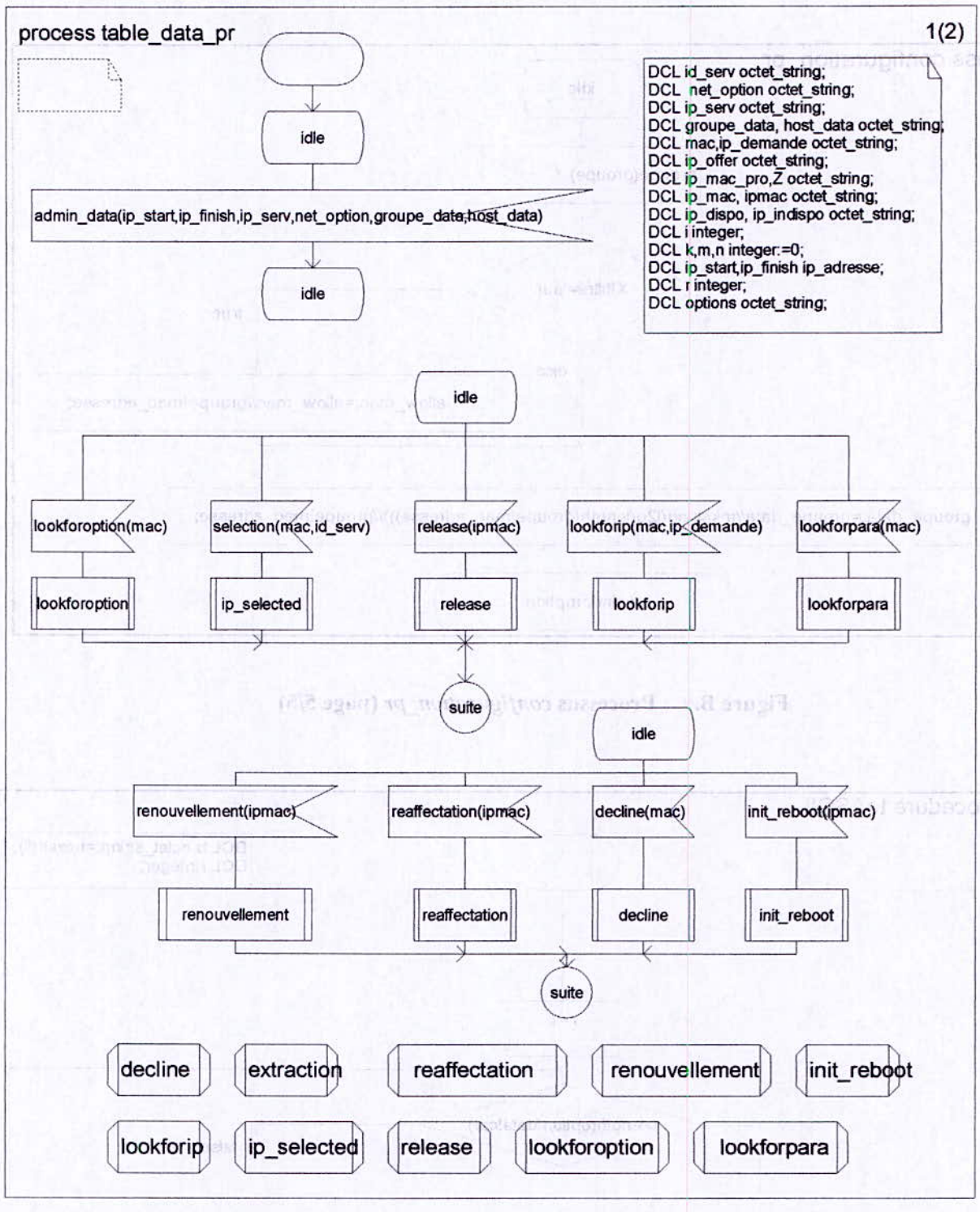


Figure B.11 : Processus *table_data_pr* (page 1/2)

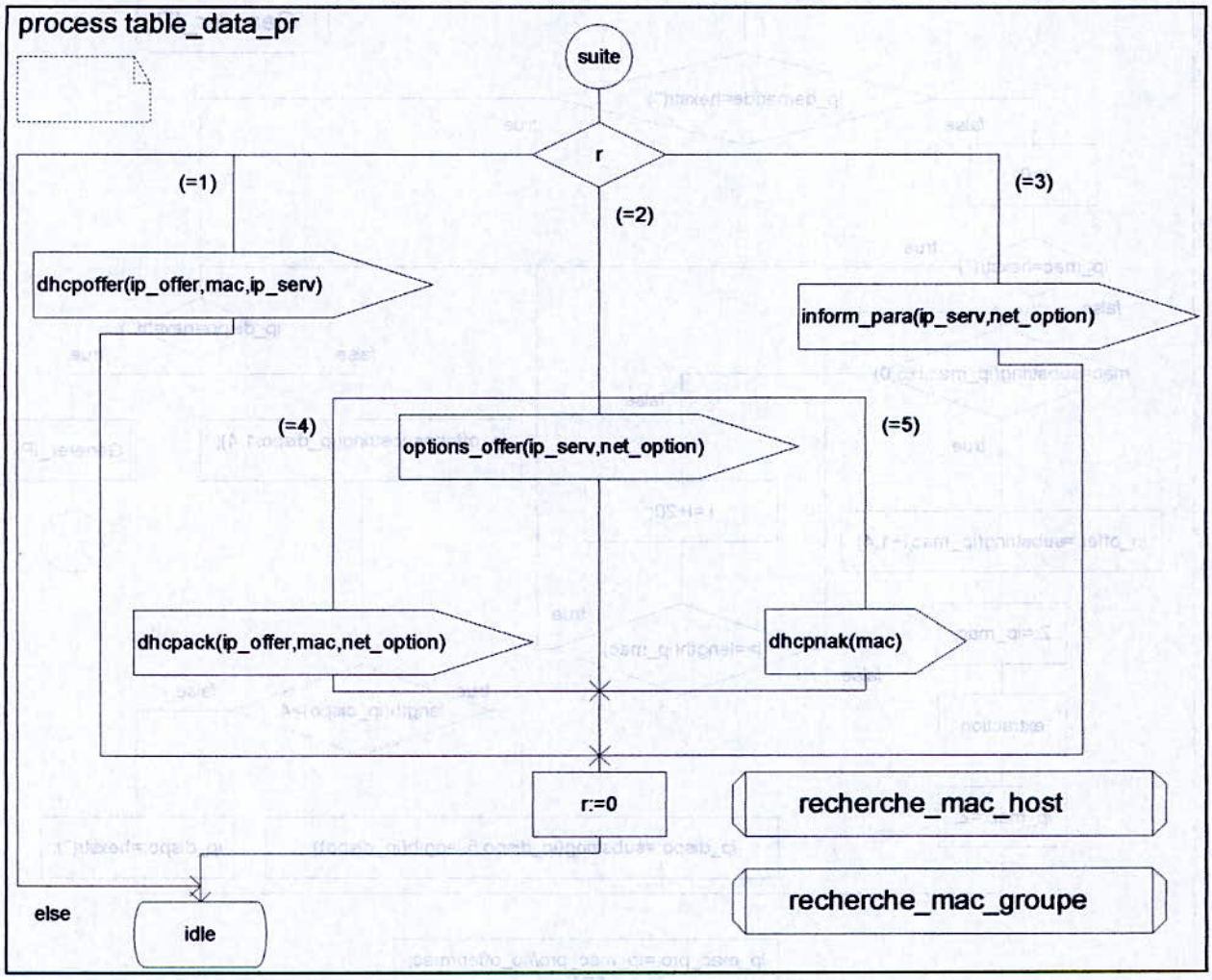


Figure B.12 : Processus *table_data_pr*(page 2/2)

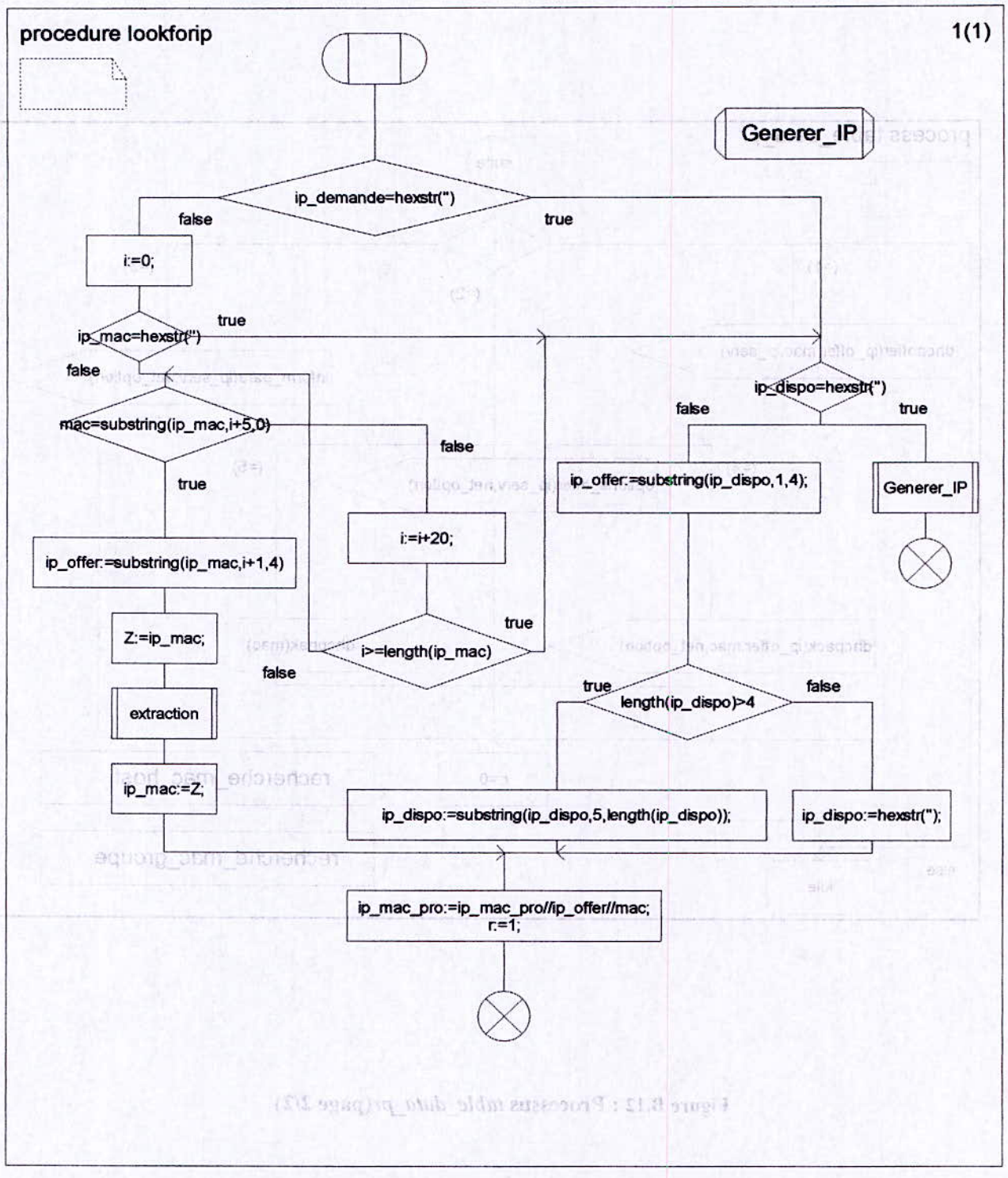


Figure B.13 : Procédure *lookforip*

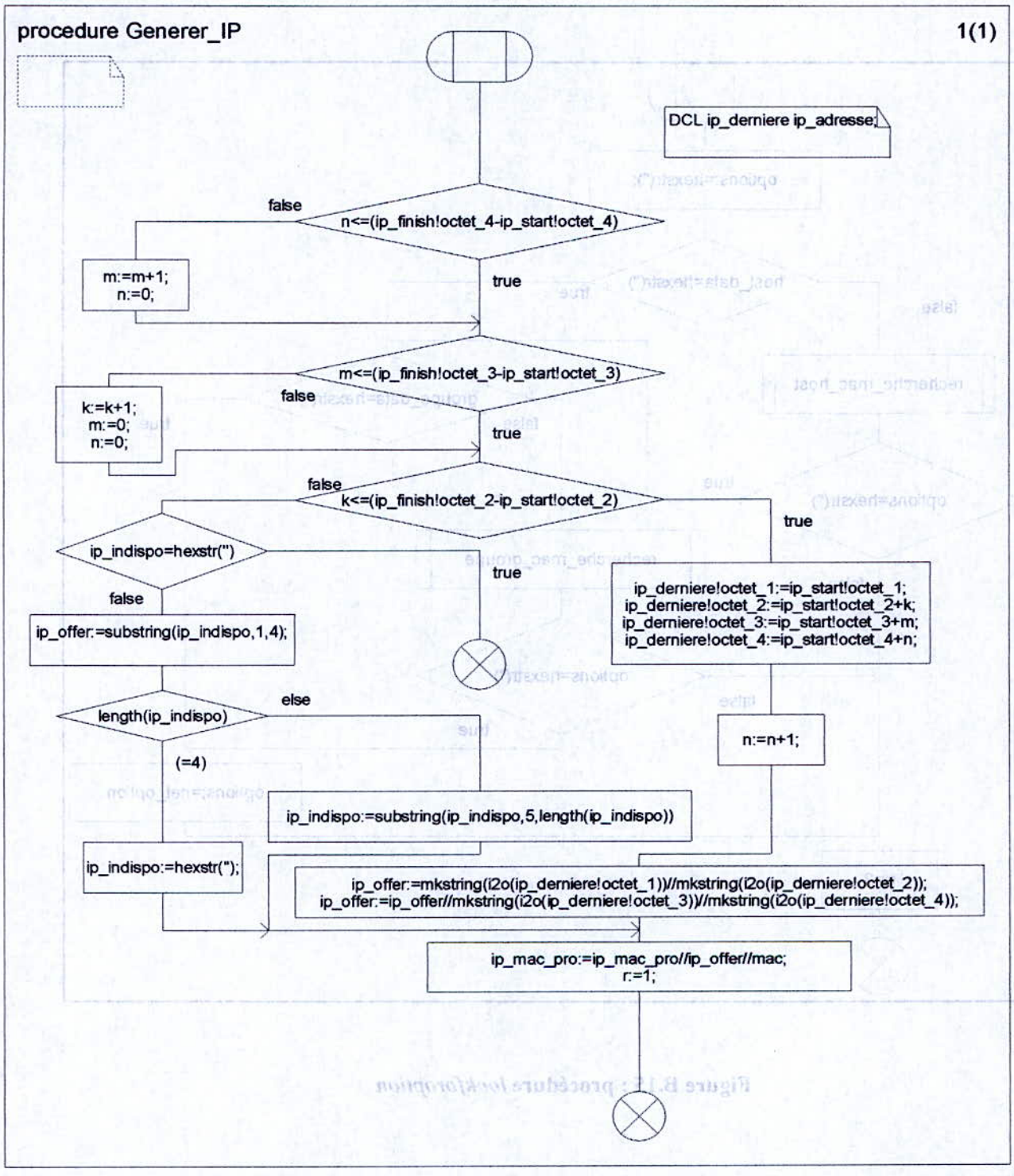


Figure B.14 : Procédure *Generer_IP*

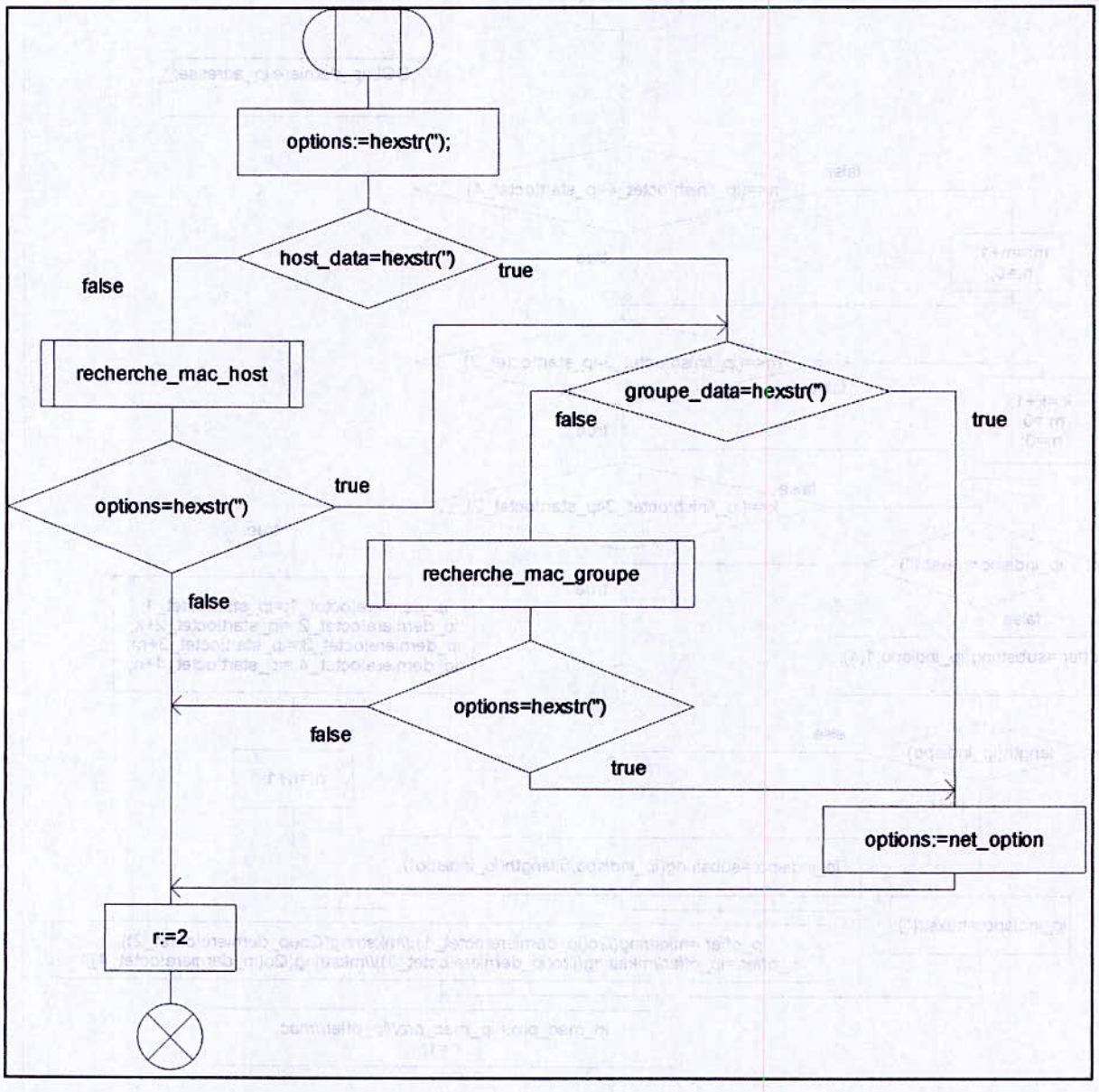


Figure B.15 : procédure *lookforoption*

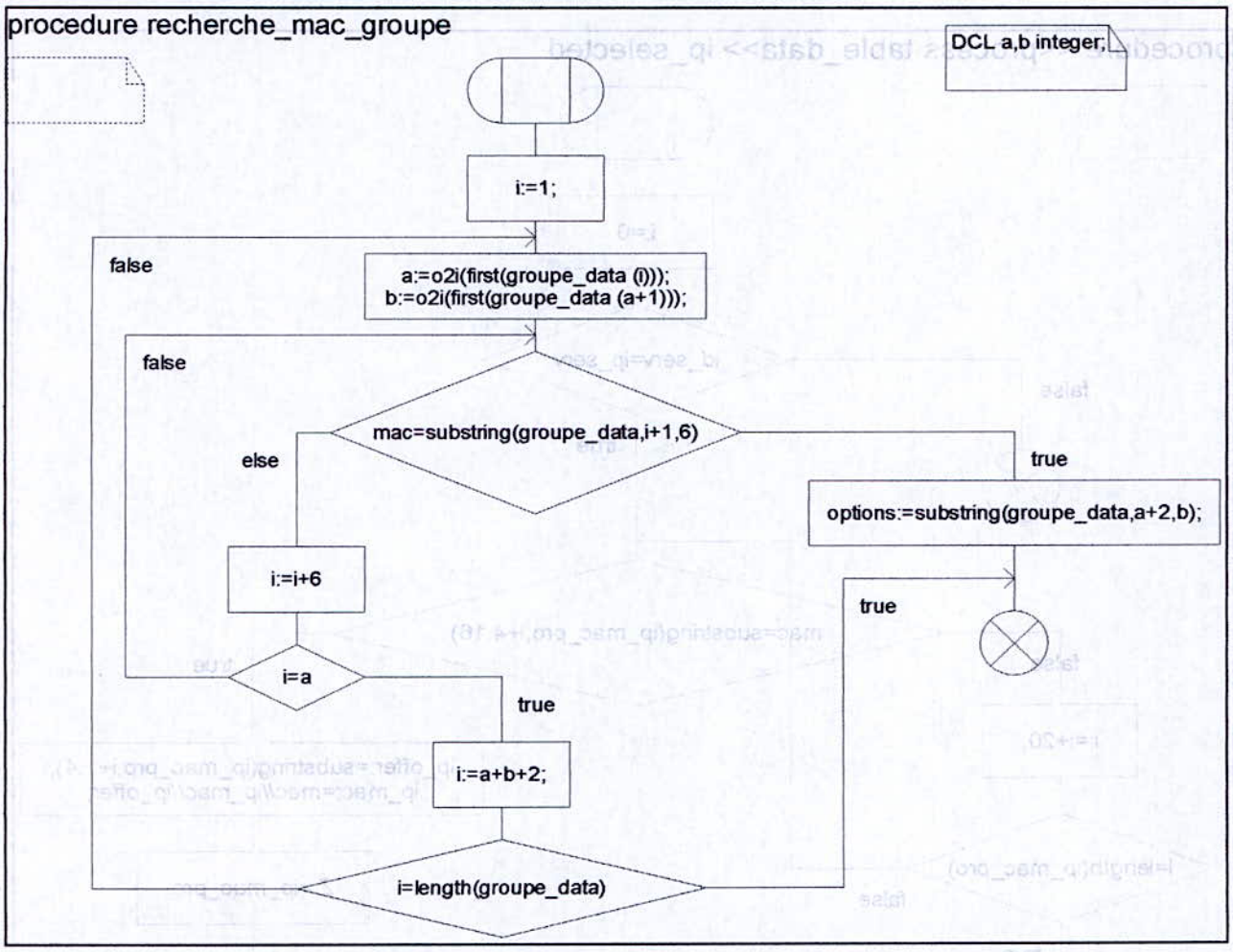


Figure B.16 : procédure *recherche_mac_groupe*

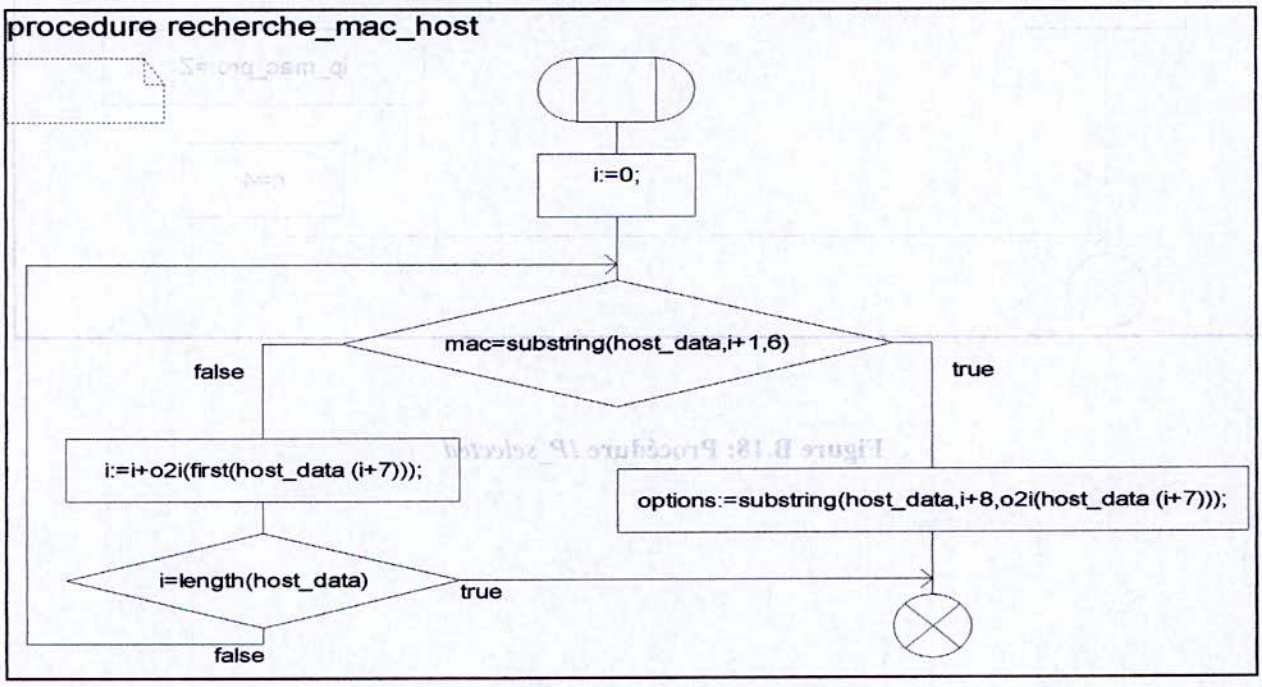


Figure B.17 : Procédure *recherche_mac_host*

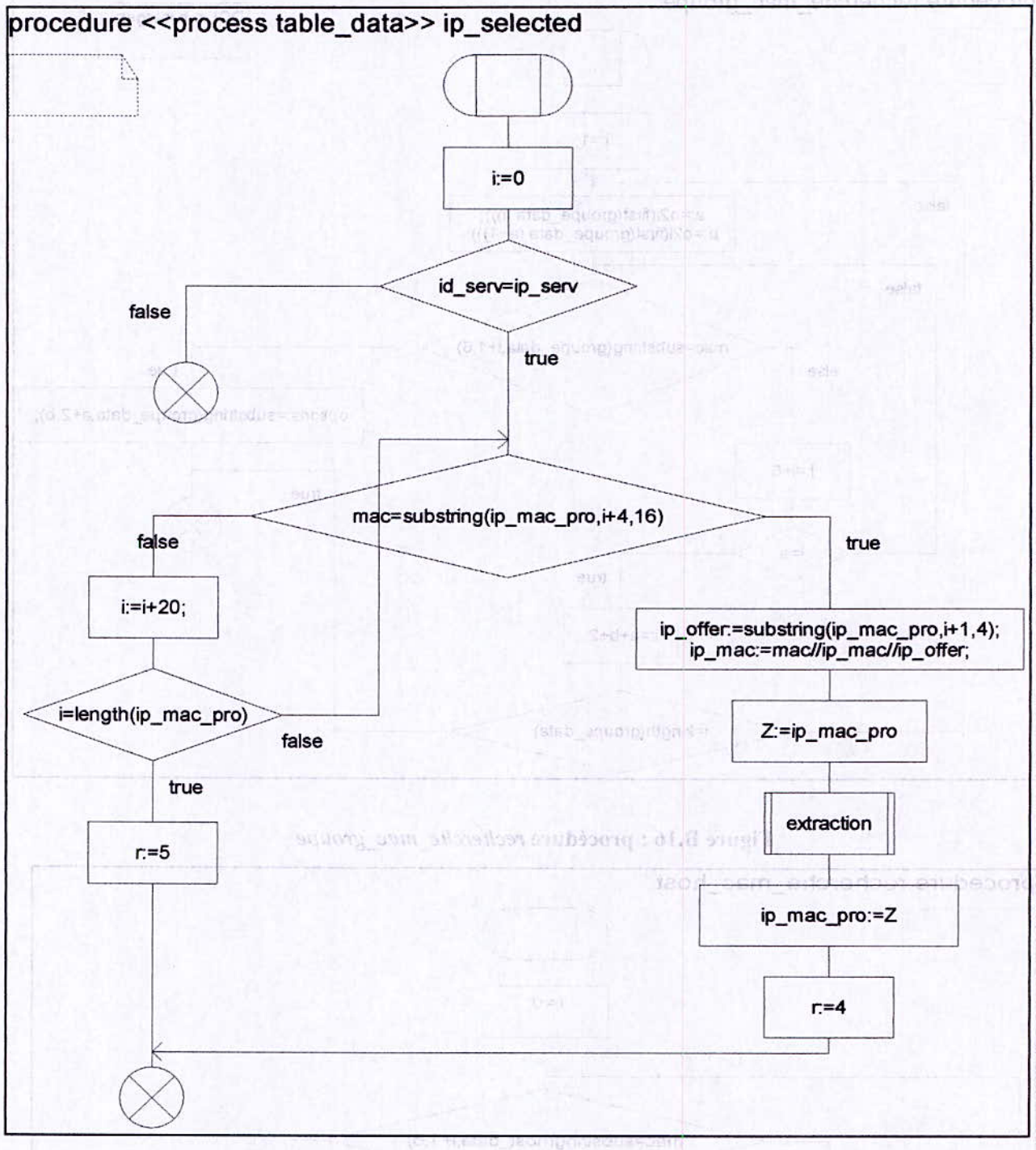


Figure B.18: Procédure *IP_selected*

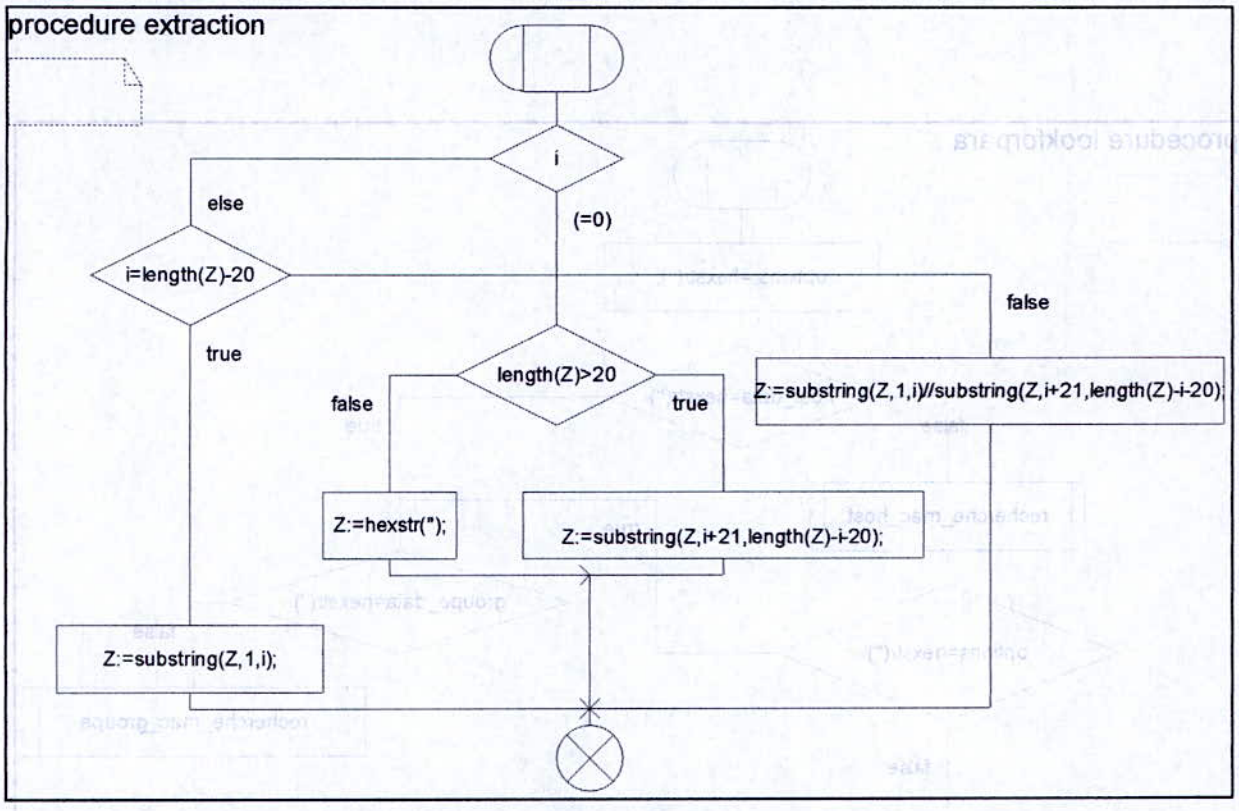


Figure B.19: Procédure *extraction*

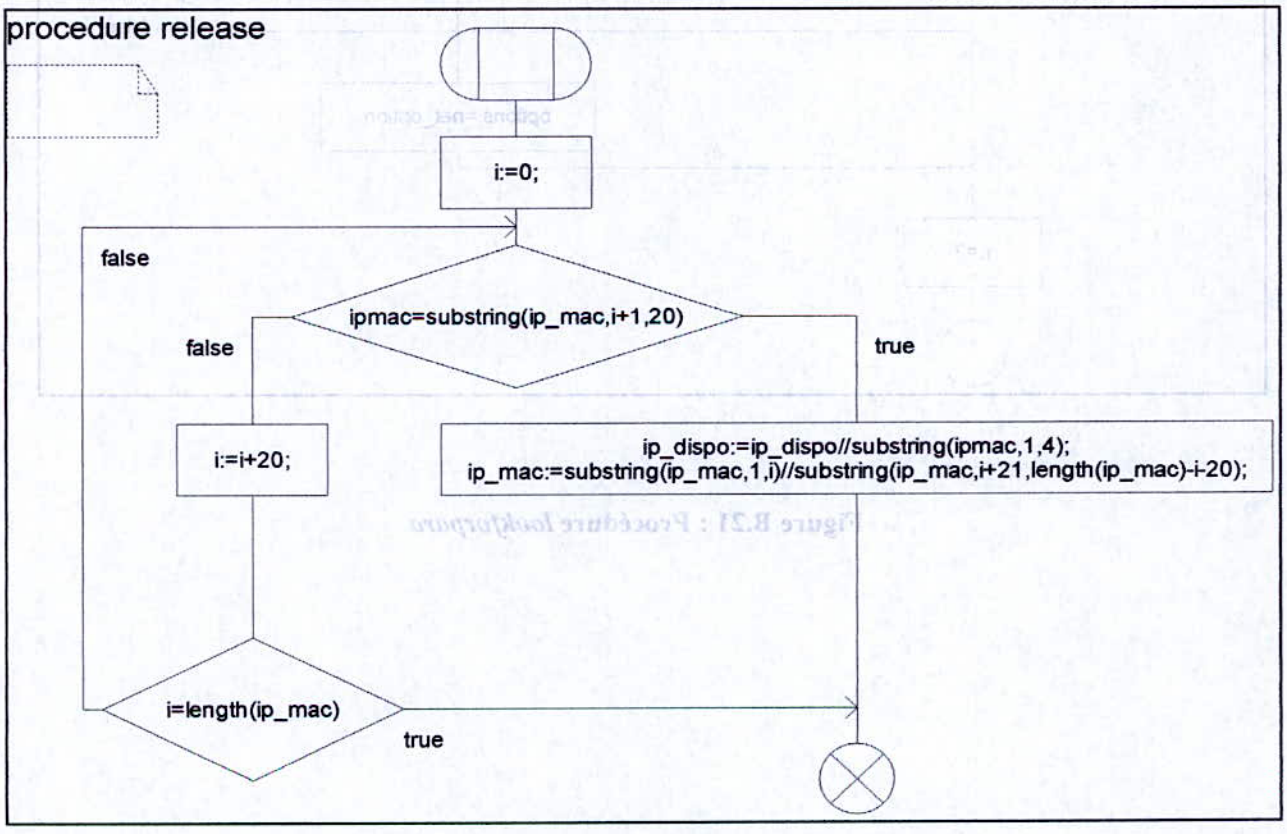


Figure B.20 : Procédure *release*

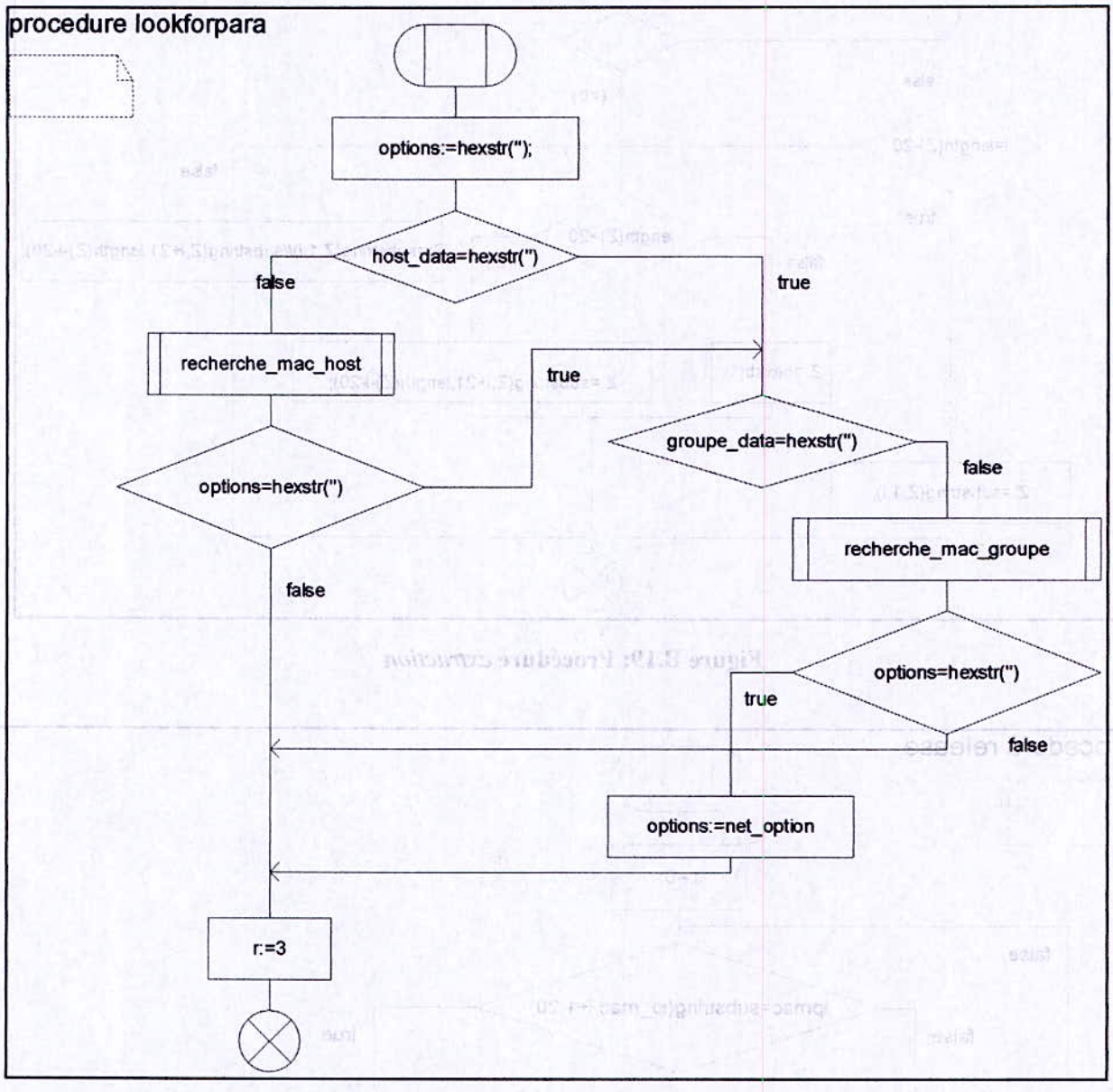
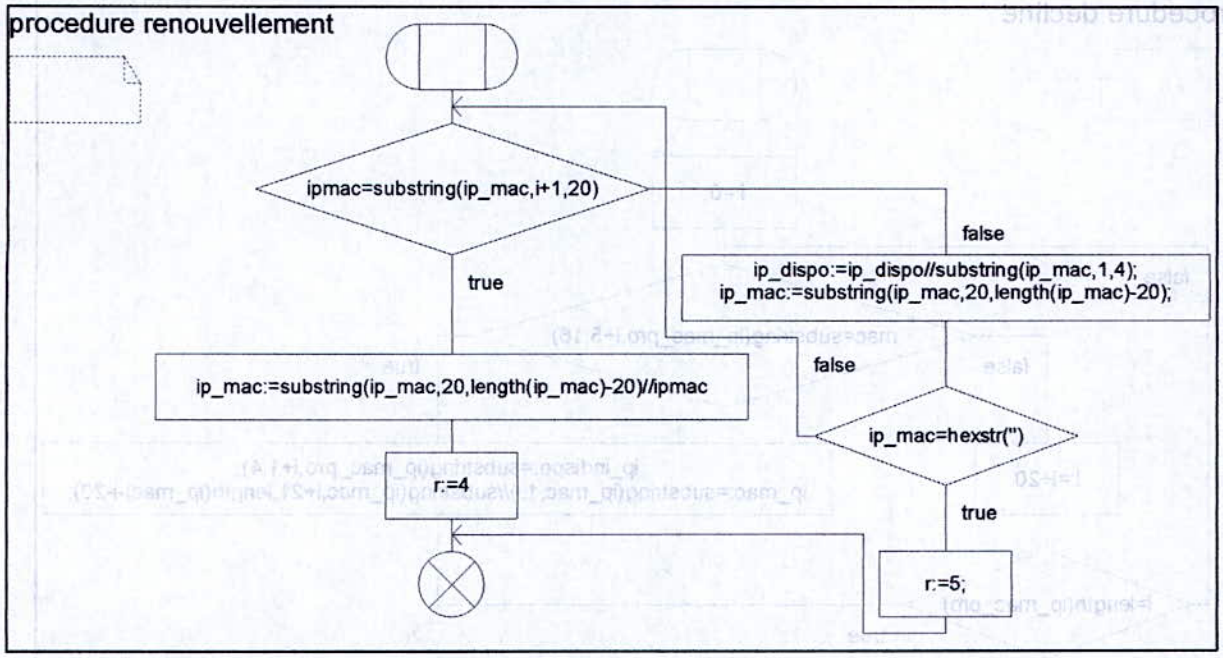
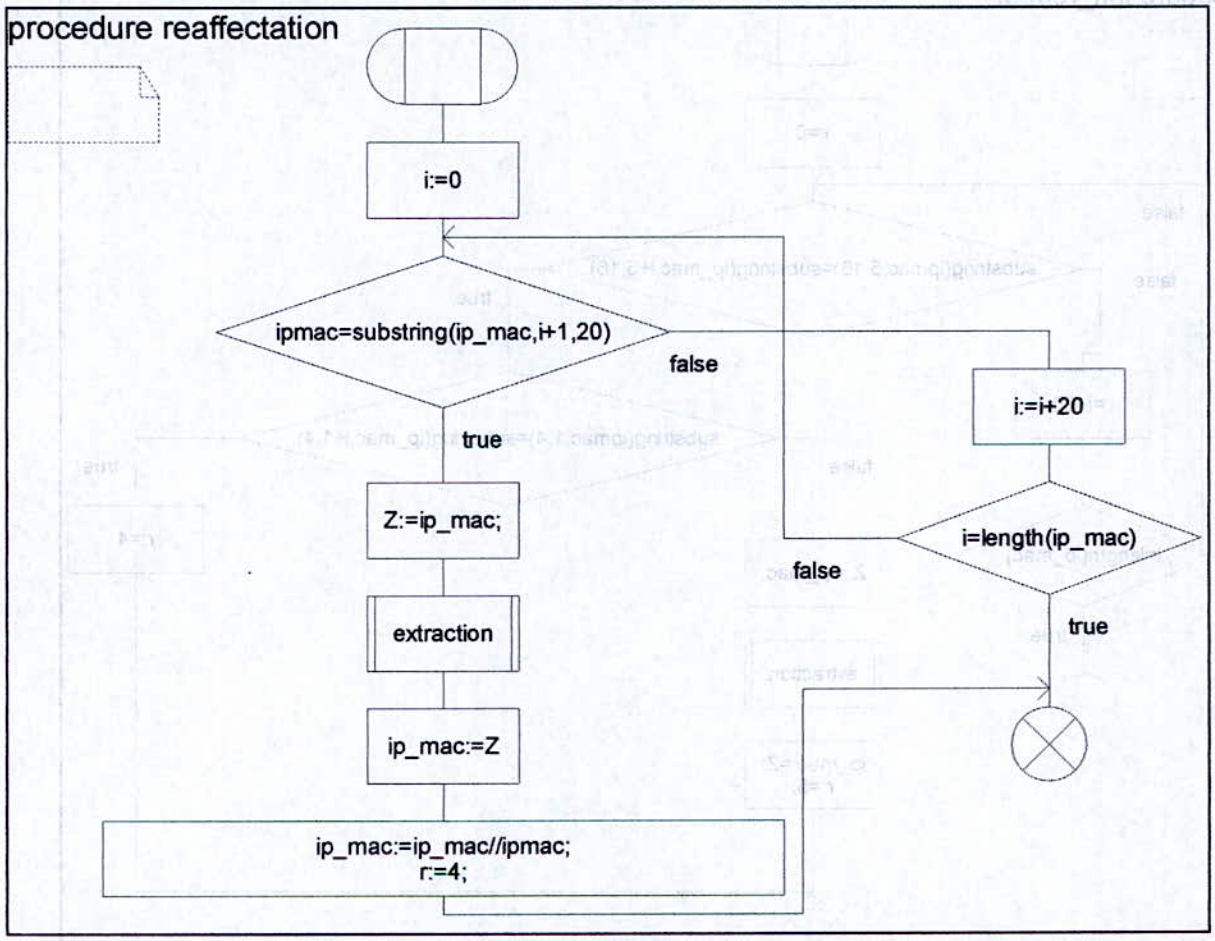


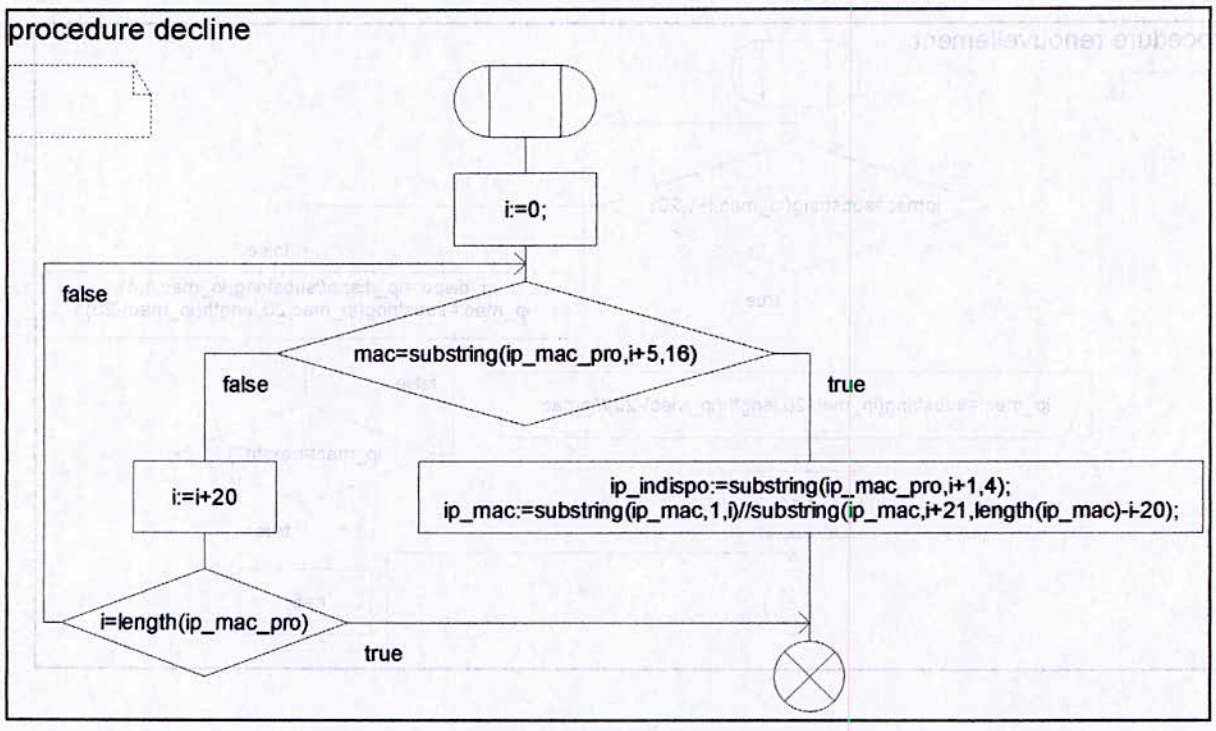
Figure B.21 : Procédure *lookforpara*



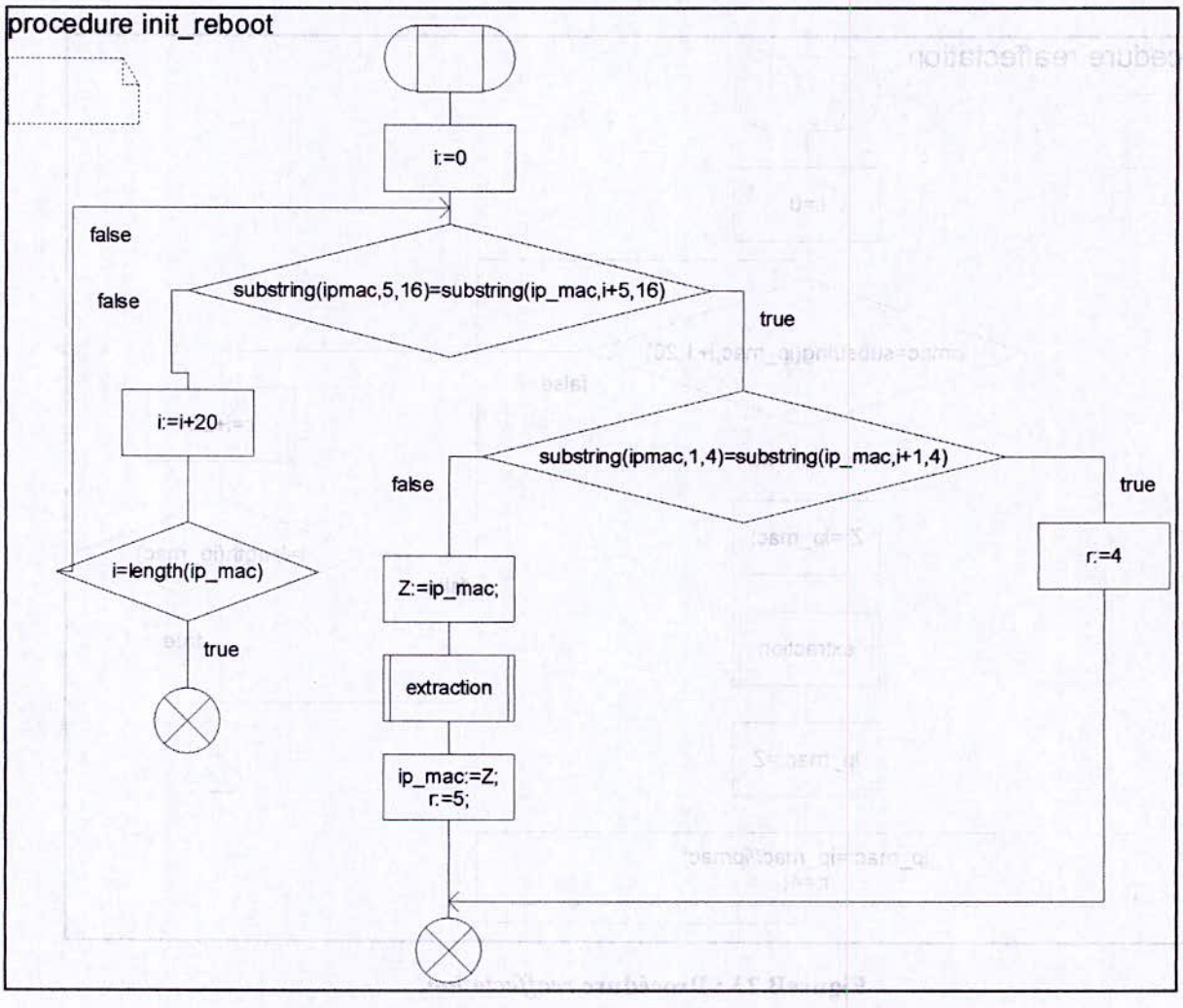
FigureB.22 : Procédure *renouvellement*



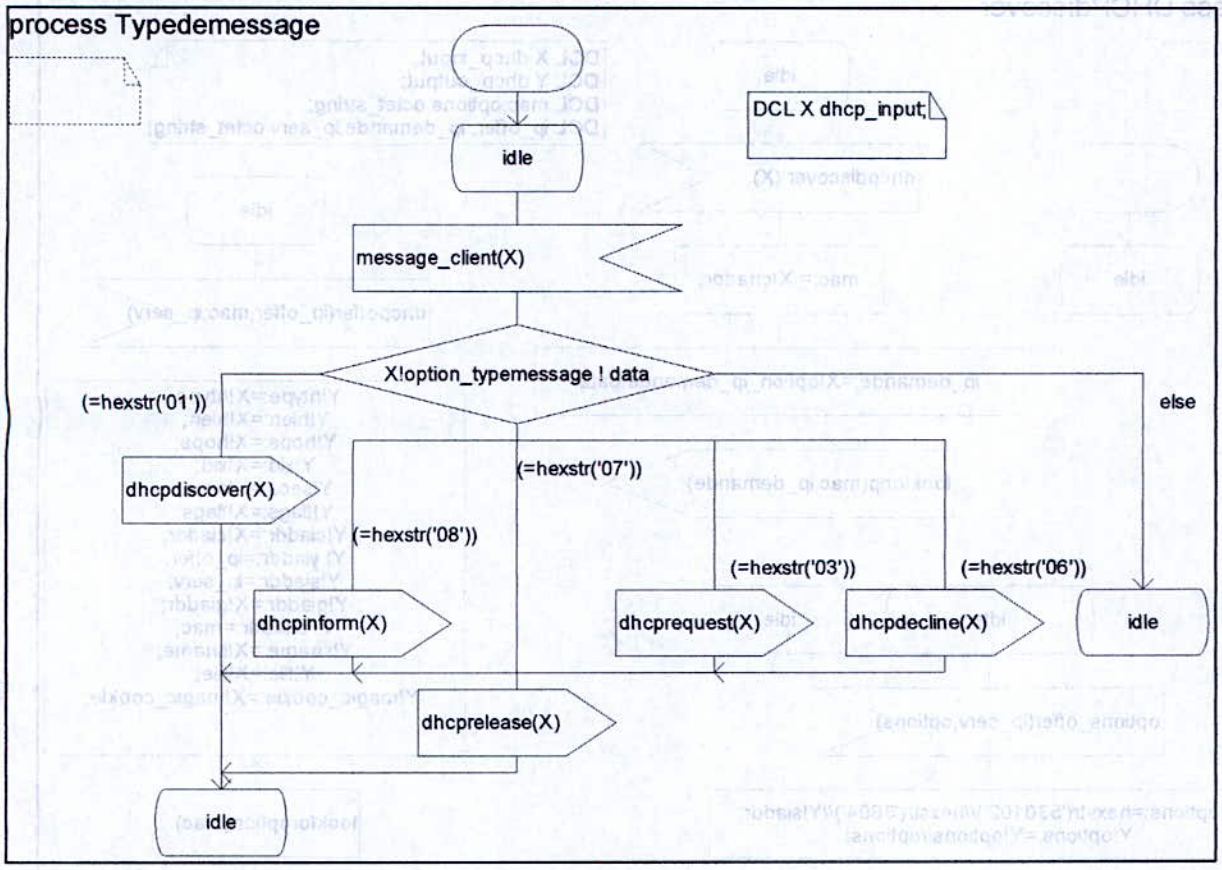
FigureB.23 : Procédure *reaffectation*



FigureB.24 : Procédure *decline*



FigureB.25 : Procédure *Init_reboot*



FigureB.26 : Processus *typedemassage* (page 1/1)

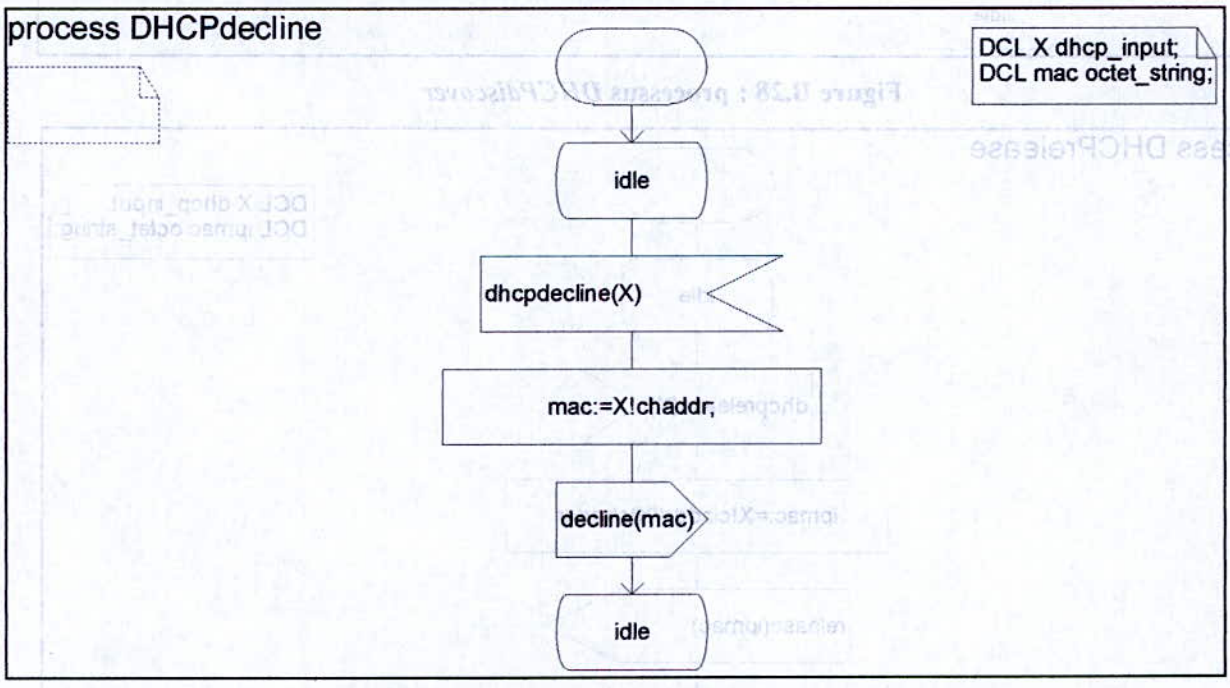


Figure B.27 : Processus *DHCPdecline*

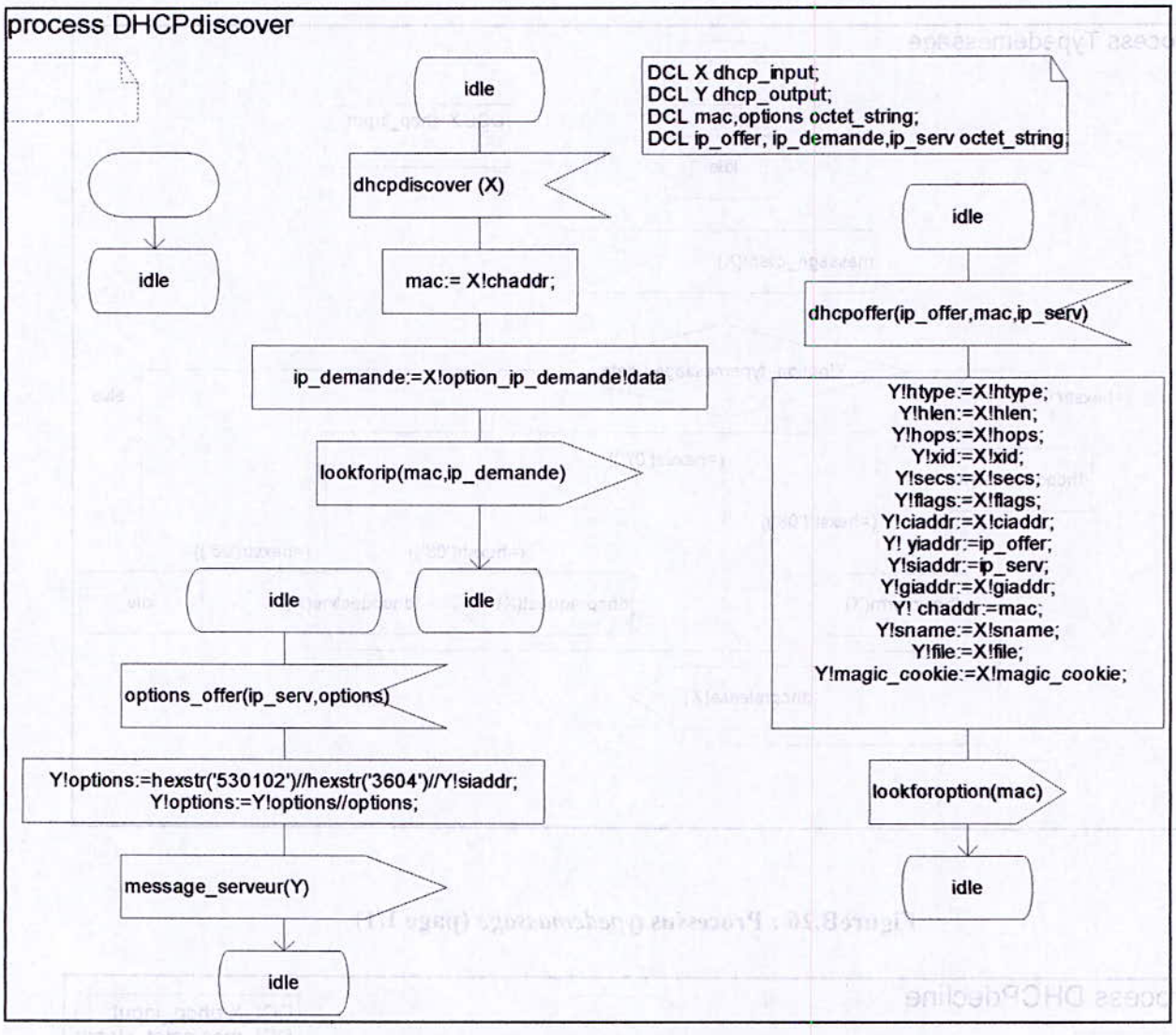


Figure B.28 : processus *DHCPdiscover*

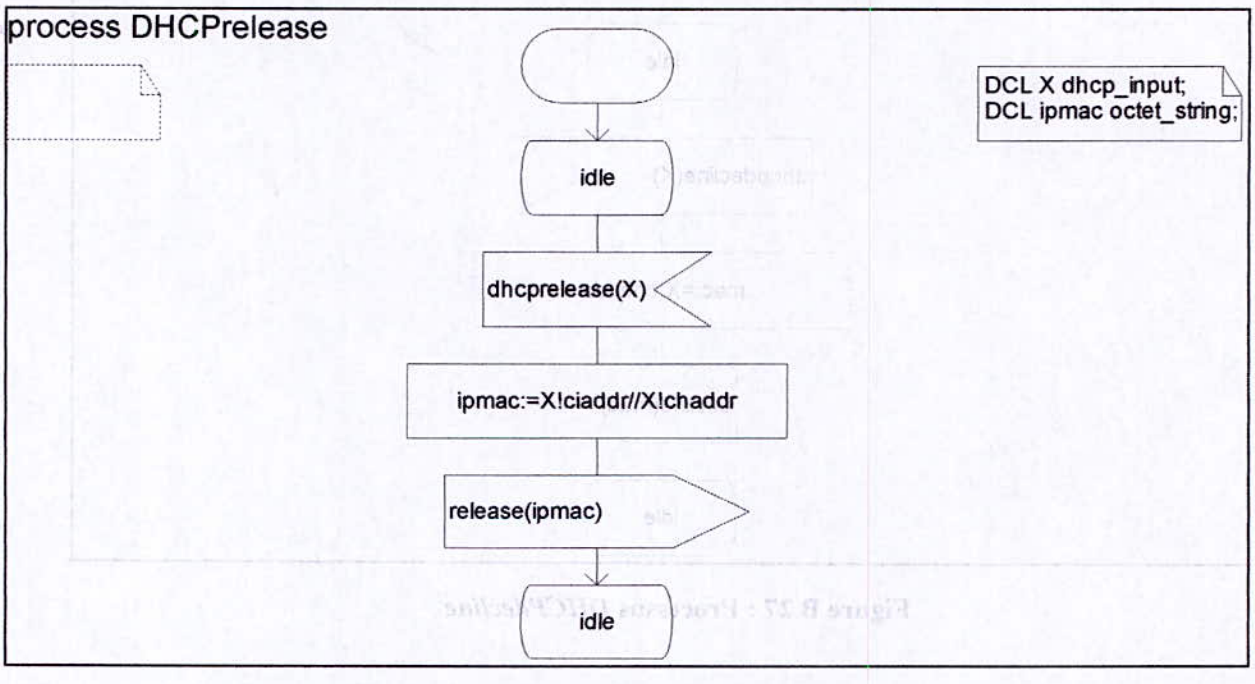
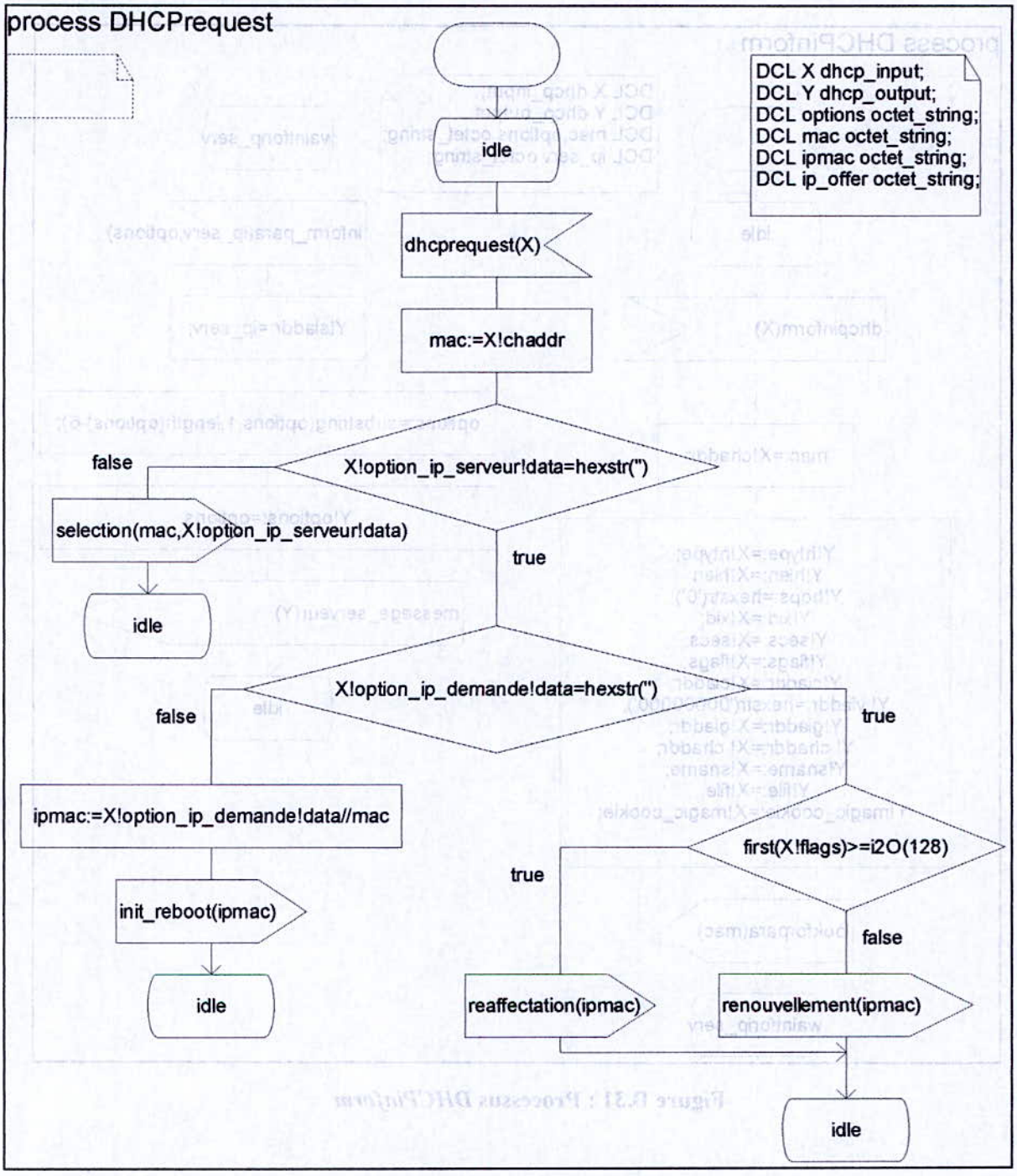


Figure B.29 : Processus *DHCPrelease*



FigureB.30: Processus *DHCPrequest*

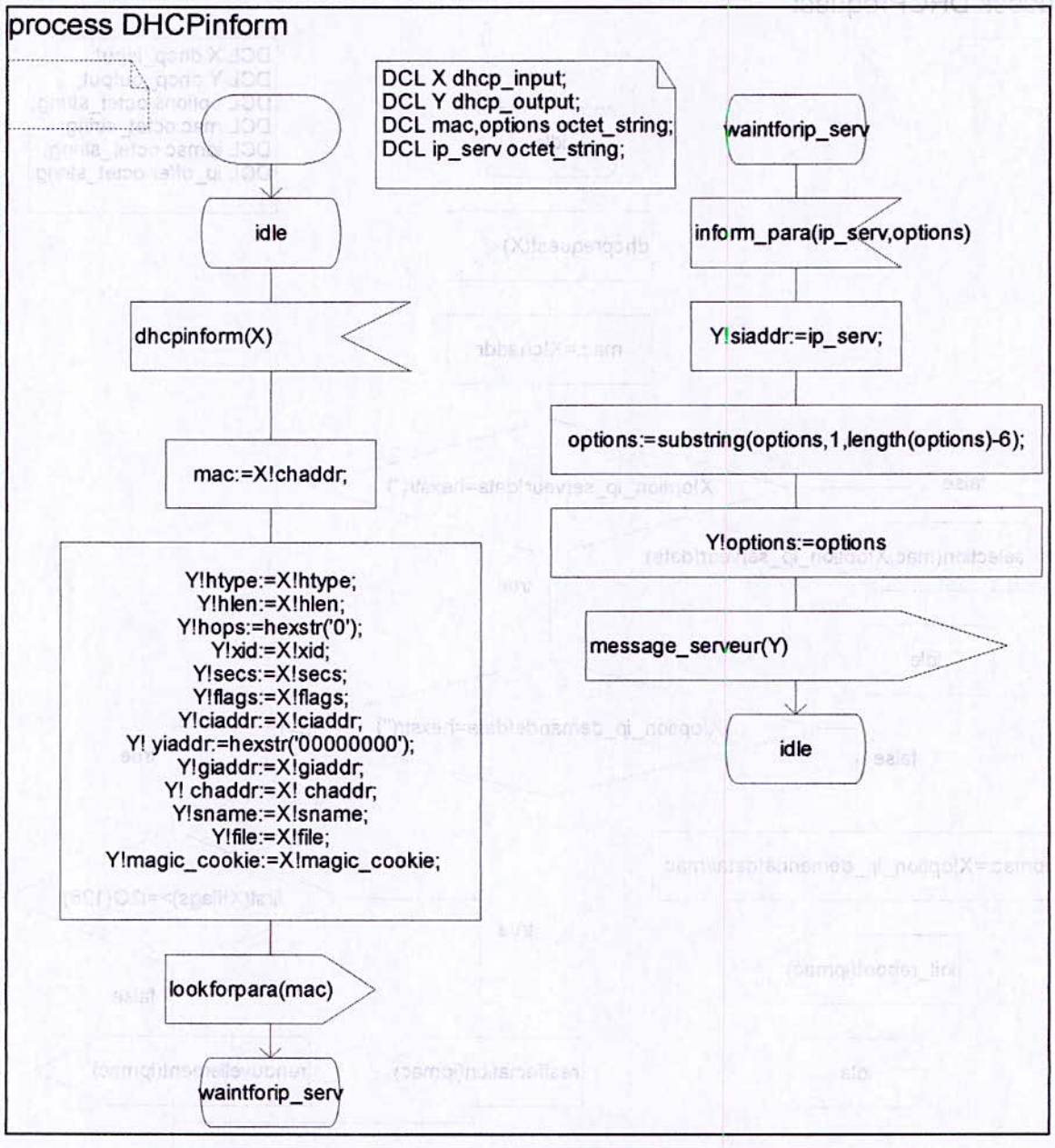


Figure B.31 : Processus *DHCPinform*

Références

[1] URL : www.commentcamarche.net

[2] URL : www.frameip.com

[3] RFC 951: *Bootstrap Protocol.* W.J. Croft, J. Gilmore. Sep-01-1985.

[4] URL : www.ethereal.com

[5] Jean-Marc DAVEAU, *Spécification système et synthèse de la communication pour le co-design Logiciel / Matériel* , Institut National Polytechnique de Grenoble, pages de 37 à 59, 1997.

[6] Emmanuel Gaudin, *Langage de développement SDL et UML: mariage de raison pour la conception des logiciels temps réel,* Article de Mars 2004 n°145 – Electronique.

[7] Laurant DOLDI , *Validation of Telecom Systems with SDL,* John Wiley & Sons Ltd, England, pp.9-24, 2003

[8] Z.MAMMERIE, *Introduction au langage de description et de spécification,* Université Paule Sabatier, Toulouse, 2001

[9] ITU-T Z.100 *Functionnal Specifcation and Description Language*, Recommandation Z.100 - Z.104, March 1993.

[10] URL : www.telelogic.com

[11] URL : www.Xilinx.com

ملخص

موضوع مشروع نهاية دراستنا هوالمساهمة في تنفيذ الموزع DHCP فوق شريحة. إعتمادا علىRFC المحددين للبروتكول DHCP، انجز وصفا جازما لهذه الخدمة، التي صممت بفضل اللغة الناطقةSDL، تحت مجال التطويرTauSDL.

تشكيل برنامج C انطلاقا من التمثيل، مكن من خلق منفذ يسمح من تثبيت النموذج المنشأ.

الكلمات المفتاحية: شبكة, PTC/PI, موزع, زبون,DHC , بروتوكول , RFC , لغة ناطقة, SDL , برنامج C.

Résumé

Ce projet de fin d'étude a pour thème la contribution à l'implémentation du serveur DHCP (*Dynamic Host Configuration Protocol)* sur puce. En se basant sur les RFC (*Request For Comments)* définissant le protocole DHCP, une description formelle de ce service a été faite grâce au langage SDL (*Specification and Description Language*), sous l'environnement de développement TauSDL. La génération d'un code C à partir de la modélisation a permis la création d'un exécutable validant le modèle établi.

Mots clés : Réseau, TCP/IP, Serveur, Client, DHCP, Protocole, RFC, Formel, SDL, Code C.

Abstract

Our graduation project deals with implementing the server DHCP (Dynamic Host Protocol) on chip. Based on the RFC (*Request For Comments)* defining the *DHCP* protocol, a formal description of this service has been done thanks to SDL (*Specification and Description Language*), under the developpement environnement TauSDL. The code C generated from the modelisation allowed to create a feasible which permitted us to validate our model.

Key words: Network, TCP/IP, server, customer(client), DHCP, Protocole, RFC, Formal, SDL, C code.