

L/003-14/04

Université Abou Bekr Belkaid



جامعة أبي بكر بلقايد

تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid- Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études
Pour l'obtention du diplôme
Licence en Informatique
Option : Informatique

Université Abou Bakr Belkaid
Faculté des Sciences
Chef de Département
d'Informatique

Thème

Filtrage des paquets TCP/IP

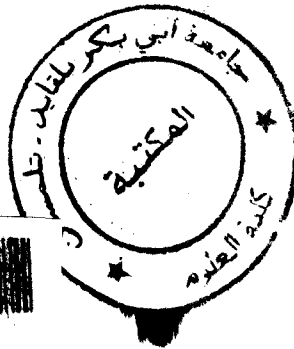
Réalisé par :

- Safi Mohamed
- Sahraoui Djamilia

Présenté le 1 juillet 2010 devant le jury composé de MM.

- Mana Mohammed (Encadreur)
- Benammar Abdelkrim (Examineur)
- Lahsaini Mohamed (Examineur)
- Merzoug (Examineur)

Année universitaire : 2009-2010

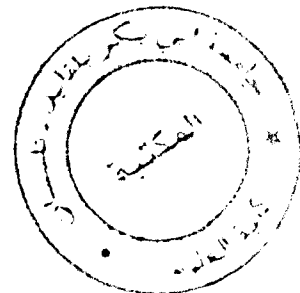


SOMMAIRE

INTRODUCTION GENERALE	4
CHAPITRE 1 : réseaux informatique	
Introduction.....	6
Définition.....	6
L'intérêt des réseaux informatiques.....	6
Les différents types de réseaux.....	7
Nature des informations échangées dans un réseau.....	7
Supports de transmission.....	7
La paire torsadée.....	7
Câble Coaxial.....	8
Câble à fibre optique.....	8
Ondes (transmission sans fils).....	9
Transmission infrarouge.....	9
Transmission radio à bande étroite.....	9
Mode de Transmission.....	9
Transmission asynchrone.....	9
Transmission synchrone.....	10
Multiplexage.....	10
Multiplexage fréquentiel (ou spatial).....	10
Multiplexage temporel.....	10
Technologies réseau.....	11
Ethernet.....	11
Token Ring.....	12
Réseau ATM (Asynchronous Transfer Mode).....	12
Réseau FDDI (Fiber Distributed Data Interface).....	12
Relais de trame.....	13
Extension réseau.....	13
Répéteurs et concentrateurs.....	13
Ponts.....	13
Commutateurs.....	14
Routeurs.....	15
Passerelles.....	15
Topologies des réseaux.....	16
Topologie en bus.....	16
Topologie en étoile.....	17
Topologie en anneau.....	17
Topologie maillée.....	18
Topologie hybride.....	18
Modèle OSI.....	18
Présentation.....	18
Fonctions des couches du model OSI.....	18

Conclusion.....	19
CHAPITRE 2 : le protocole TCP/IP	
Introduction.....	20
Pile protocolaire TCP/IP.....	20
Définition.....	20
Modèle d'architecture	21
Couche physique.....	21
Couche de liaison de données.....	22
Couche transport.....	23
Couche d'application.....	25
système d'adressage.....	25
Le principe de base d'un réseau IP.....	25
L'ADRESSE IP.....	26
LES DIFFERENTES CLASSES D'ADRESSES INTERNET.....	26
Conclusion.....	27
CHAPITRE 3 : les sockets	
Introduction.....	28
Position des sockets dans le modèle OSI.....	28
Déroulement d'une communication avec les sockets.....	29
Les fonctions des sockets	30
La fonction socket().....	30
La fonction bind().....	31
La fonction listen().....	31
La fonction accept().....	32
La fonction connect().....	32
La fonction recv().....	32
La fonction send().....	33
Les fonctions close() et shutdown().....	33
Architecture Client/serveur en java avec les sockets.....	34
Introduction.....	34
Le client.....	34
Les Sockets.....	34
D'Autres usages.....	35
Le serveur.....	36
Introduction.....	36
Le Socket serveur.....	36
Fonctions sur les sockets.....	37
Conclusion.....	37
CHAPITRE 4 : développement de l'application	
Introduction.....	38
Application de JPCAP.....	38
La sécurité	38

Scanneur de port.....	39
LES ANALYSEURS.....	39
LES SYSTEMES DE DETECTION D'INTRUSION.....	39
Moniteur de comportement.....	40
Contrôleur d'intégrité.....	40
Les algorithmes de l'application	40
Obtenir la liste des interfaces du réseau.....	40
Ouvrir l'interface réseau	41
Capturer les paquets d'interface réseau	41
Filtrer les paquets capturés.....	41
Sauvegarder les paquets capturés dans un fichier.....	42
Lire Les paquets sauvegardés à partir d'un fichier.....	43
Transmettre les paquets vers une autre interface réseau	43
Présentation de la simulation.....	44
Langage « NetBeans ».....	44
Définition du « NetBeans ».....	44
Caractéristique de « NetBeans ».....	44
Principes de développement en « NetBeans ».....	44
Le concept de classe.....	44
Les Objets.....	45
Les méthodes.....	45
La partie programmation.....	45
Obtenir la liste des interfaces réseaux.....	45
Ouvrir l'interface réseau.....	47
Capturer les parquets d'interface réseau.....	47
Filtre les parquets capturés.....	48
Sauvegarde les parquets dans un fichier.....	48
Lire les parquets sauvegardés dans fichier.....	49
Transmettre des parquets via d'interface réseau.....	50
Conclusion.....	54
Conclusion générale.....	55
Références Bibliographiques.....	56
Liste de figures.....	57
Liste des tableaux.....	58



Introduction générale

Le développement croissant des systèmes informatiques et l'avènement de l'internet ont bouleversé notre vie quotidienne. Tout le monde maintenant utilise l'outil informatique et communique via internet. Cette technologie est devenue de plus en plus indispensable et nécessaire à la réussite des personnes et entreprises car elle a participé grandement à rendre la vie plus aisée. Grâce à internet, le monde est devenu un petit village; on peut atteindre n'importe quel point de ce monde à n'importe quel moment.

Les avantages de cette technologie sont nombreux et indénombrables. D'un autre côté, ses inconvénients ou ses risques sont inévitables. Pas mal de personnes ou d'entreprises qui ont subi des dégâts sérieux à cause des intrusions et des virus.

Pour minimiser tout risque, la mise en place d'un système de sécurité est nécessaire et important. Plusieurs systèmes de sécurité ont été développés tels que les par feux, les antivirus, la cryptographie,...etc.

Dans notre projet, nous visons à développer une application permettant de surveiller et identifier tout le trafic qui circule sur le réseau informatique. Pour cela, nous avons adopté le plan de travail suivant:

En premier chapitre, nous avons présenté d'une manière générale les réseaux informatiques; le deuxième chapitre a été consacré au protocole TCP/IP. Nous avons abordé également les sockets en troisième chapitre et le dernier chapitre a été destiné à la conception et l'implémentation de l'application.

Chapitre 1

Introduction aux Réseaux Informatiques

1. Introduction

La communication entre les hommes comme entre les machines est de nos jours bien souvent le point clé pour la réussite d'une entreprise. Pour faciliter ces communications, des réseaux ont été mis en place. Le sens du mot réseau varie grandement selon les contextes, et il peut désigner l'ensemble des appareils physique et des câbles, les PC et les serveurs ainsi que leur système d'exploitation de réseau (comme Netware, Windows, Unix, Linux, Mac ...) et tout ce qui est nécessaire pour relier tout cela. Le terme peut également désigner une plage d'adresse IP, comme le réseau 10.0.0.0 ou 192.168.0.0. Dans d'autres contextes encore, il peut s'appliquer à un segment de liaison de données, comme un unique anneau FFDI ou Ethernet. Enfin, il peut aussi désigner les câbles, les commutateurs, les routeurs, les concentrateurs et tous les équipements différents appartenant à une société donnée ou à une Zone géographique particulière. En combinant plusieurs de ces réseaux, on obtient un interréseau ; Internet est l'exemple le plus connu de ces réseaux de réseaux, mais il en existe bien d'autres, de par la grande diversité des types d'informations (voix, vidéo, données informatiques...).

2. Définition

Un réseau informatique est un groupe d'ordinateurs reliés les uns aux autres qui permettent aux utilisateurs d'échanger des informations et de partager du matériel tel qu'une imprimante.

3. L'intérêt des réseaux informatiques

- Echange et partage de données informatiques
- Partage d'une connexion Internet
- Messagerie électronique
- Transfert de fichiers
- Lancement de procédures distantes (client/ serveur)
- Accès à des bases de données centralisées ou réparties
- Partage de logiciels
- Partage de périphériques : accès à des imprimantes, des traceurs, ...
- Archivage : utilisation d'espace disque pour l'archivage ou la sauvegarde
- Etc.

4. Les différents types de réseaux

On peut distinguer les réseaux par les critères suivants :

- les distances qu'ils couvrent
- Par leur débit
- Temps de réponse
- Par d'autres caractéristiques techniques.

On peut distinguer cinq grandes classes de réseaux en prenant comme critère la distance :

Les bus, les structures d'interconnexions, les réseaux locaux (LAN), les réseaux métropolitaines (MAN) et les réseaux longues distances (WAN).

5. Nature des informations échangées dans un réseau

L'information qui transite sur les réseaux consiste en messages de types divers :

SONS : parole, musique

IMAGES : fixes, animées (vidéo), noir et blanc, nuances de gris, couleurs

TEXTES : avec styles et formats

DONNEES INFORMATIQUES : informations codées en binaire

6. Supports de transmission

Les supports de transmission peuvent être matériels (fils, câbles, ...) ou immatériels (ondes).

a. La paire torsadée

Une ligne de transmission est constituée de 2 fils au minimum ce qu'on appelle une "paire". Les paires métalliques sont généralement constituées de cuivre, Les deux fils de la paire sont torsadés l'un sur l'autre afin de présenter une meilleure immunité aux perturbations électromagnétiques extérieures.

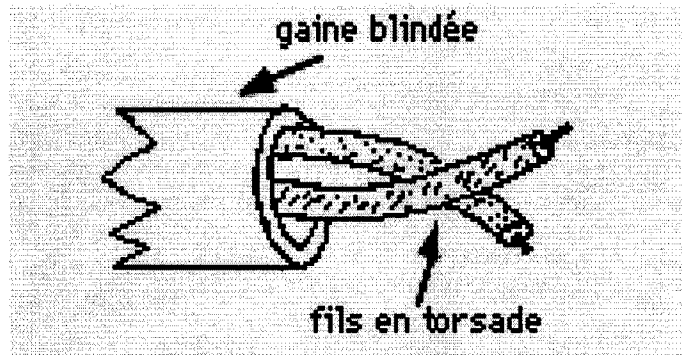


Figure 1.1 paire torsadée

b. Câble Coaxial

Ce support est constitué de 2 conducteurs à symétrie cylindrique de même axe, l'un central de rayon R_1 , l'autre périphérique de rayon R_2 , séparés par un isolant.

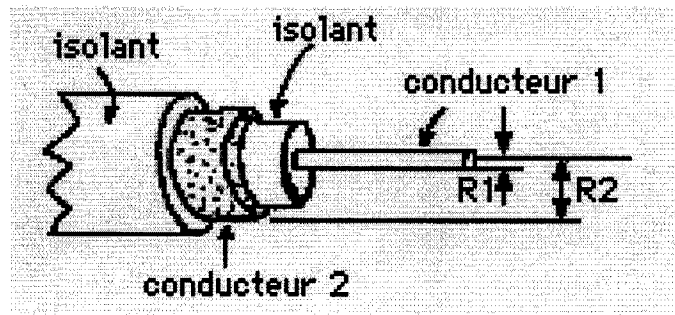


Figure 1.2 câble coaxial

c. Câble à fibre optique

Le câble à fibre optique est utilisé pour transporter des signaux de données numériques, sous forme d'impulsions lumineuses. Il est bien adapté à une transmission de données rapide et fiable, car le signal est transmis très rapidement et est très peu sensible aux interférences.

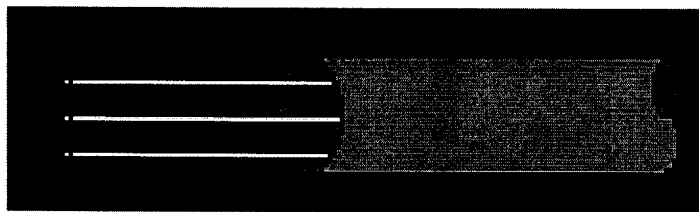


Figure 1.3 câble à fibre optique

d. Ondes (transmission sans fils)

Un réseau sans fil standard fonctionne pratiquement comme un réseau câblé : une carte réseau sans fil dotée d'un émetteur- récepteur (périphérique transmettant et recevant des signaux analogiques et numérique) est installée dans chaque ordinateur. L'utilisateur communique avec le réseau comme s'il s'agissait d'un ordinateur câblé.

Il existe deux techniques courantes de transmission sans fil pour un réseau local : la transmission infrarouge et la transmission radio à bande étroite.

e. Transmission infrarouge

Cette technique fait appel à un faisceau de lumière infrarouge pour transporter les données entre les périphériques. Il ne doit y avoir aucun obstacle entre l'émetteur et le récepteur. En effet, tout objet qui bloquerait le signal infrarouge empêcherait la communication de s'établir. Ces systèmes doivent générer des signaux forts, car les signaux de transmission faibles sont sensibles aux interférences des sources lumineuses, telles que les fenêtres.

f. Transmission radio à bande étroite

L'émetteur et le récepteur doivent être réglés sur une certaine fréquence. La transmission radio à bande étroite ne nécessite pas de visibilité entre l'émetteur et le récepteur, puisqu'elle utilise des ondes radio. Toutefois, cette technique est sujette aux interférences provenant des objets métalliques. La transmission radio à bande étroite est un service nécessitant un abonnement. L'utilisateur paie un droit d'utilisation.

7. Mode de Transmission

a. Transmission asynchrone

Les caractères sont codés sur un nombre fini de positions binaires, et sont encadrés par des bits délimiteurs (bit START et bit STOP).

Le mode asynchrone oblige à envoyer les informations caractère par caractère.

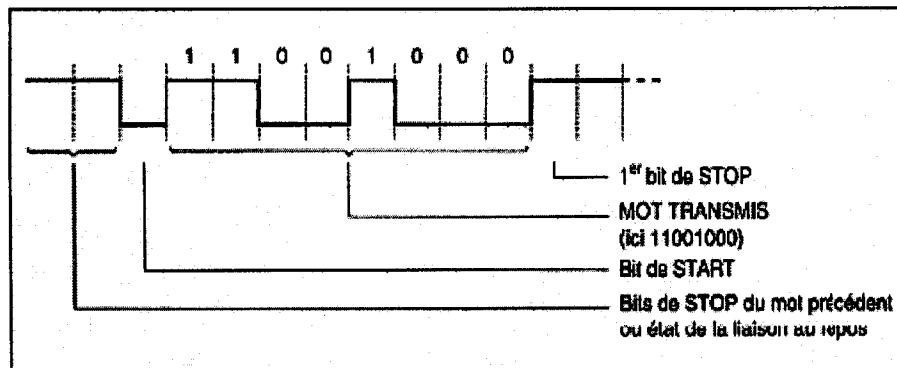


Figure 1.4 transmission asynchrone

b. Transmission synchrone

Dans la transmission synchrone, le récepteur reçoit, avec les données, l'horloge de synchronisation qui donne le rythme des bits (base de temps).

8. Multiplexage

Le multiplexage consiste à faire passer plusieurs messages sur un même tronçon de réseau.

On distingue deux types de multiplexage :

a. Multiplexage fréquentiel (ou spatial)

La bande passante du canal est divisée en sous-bandes (canaux) chaque message correspond à une sous-bande de fréquence ; un multiplexeur mélange les différents messages ; un démultiplexeur, à l'arrivée, sépare, grâce à un filtrage en fréquence, les messages.

b. Multiplexage temporel

Ce type de multiplexage est bien adapté aux réseaux à commutation de paquets. Le multiplexeur n'est autre qu'un mélangeur de paquets, le démultiplexeur est un trieur de paquets.

9. Technologies réseau

Différentes technologies réseau sont utilisées pour permettre aux ordinateurs de communiquer sur des réseaux locaux et étendus. Chaque technologie réseau utilise sa propre méthode d'accès. Une méthode d'accès est la manière de placer et de retirer des informations du réseau. On parle aussi de média réseau. Les principales technologies réseau sont les suivantes :

- Ethernet
- Token ring
- Réseau ATM (Asynchronous transfer mode)
- Réseau FDDI (Fiber Distributed Data Interface)
- Relais de trame

a. Ethernet

Ethernet est une technologie de réseau local très répandue. Elle fait appel au protocole CSMA/CD (Carrier sense Multiple Access With Collision Detection : détection de signal et de collision) entre les clients, et peut être utilisée avec différents types de câbles. Un réseau Ethernet est connecté au moyen d'une topologie en bus, dans laquelle le câble est terminé aux deux extrémités.

La méthode d'accès au réseau utilisée avec Ethernet est appelée CSMA/CD. CSMA/CD est un ensemble de règles qui déterminent la façon dont les périphériques du réseau répondent lorsque deux de ces périphériques tentent de transmettre simultanément des données sur le réseau. Un ordinateur ne transmet des données que lorsqu'il détecte que le câble est libre. Une fois que l'ordinateur a transmis des données sur le câble, aucun autre ordinateur ne peut transmettre des données tant que les données d'origine n'ont pas atteint leur destination. Libérant ainsi le câble. Lorsqu'il détecte une collision, un périphérique attend pendant un délai aléatoire, puis tente de retransmettre le message. S'il détecte de nouveau une collision, il attendra deux fois plus longtemps avant de retransmettre le message. Les vitesses de transfert sur un réseau Ethernet sont 10Mbits/s, 100Mbits/s ou 1Gbits/s.

b. Token Ring

Les réseaux token ring sont implémentés dans une topologie en anneau. Toute fois, la topologie physique est en étoile et c'est dans le concentrateur que se font les liaisons d'un ordinateur à l'autre. La méthode d'accès pour cette technologie est le passage de jeton. Un jeton est une séquence spéciale de bits qui transitent sur l'anneau. Un ordinateur ne peut pas transmettre des données tant qu'il n'est pas en possession du jeton. Lorsque le premier ordinateur de l'anneau se trouve en ligne, le réseau génère un jeton. Ce jeton transite sur l'anneau jusqu'à ce que l'un de ces ordinateurs prenne le contrôle de jeton. La vitesse de transfert d'un réseau Token Ring est comprise entre 4 et 16Mb/s.

c. Réseau ATM (Asynchronous Transfer Mode)

ATM est un réseau à commutation de paquets de taille fixe sur des réseaux locaux ou étendus, alors que les autres technologies utilisent des paquets de longueurs variable. Un réseau ATM utilise la méthode d'accès point à point. Cette technologie consiste à transférer des paquets de longueur fixe d'un ordinateur à un autre au moyen d'un commutateur ATM. Il en résulte une technologie qui transmet rapidement de petits paquets de données. La vitesse de transfert d'une ligne ATM varie de 155Mbits/s à 622Mbits/s.

d. Réseau FDDI (Fiber Distributed Data Interface)

Les réseaux FDDI ressemblent aux réseaux Token Ring à jeton. Ils sont constitués de deux anneaux appelés anneau principal et anneau secondaire. En cas de problème avec l'anneau principal, par exemple une défaillance de l'anneau ou une rupture de câble, l'anneau se reconfigure en transférant les données sur l'anneau secondaire, qui continue à transmettre. La méthode d'accès utilisée dans un réseau FDDI est le passage de jeton, toutefois, cette méthode est plus efficace que le Token Ring traditionnel car plusieurs trames peuvent circuler sur l'anneau simultanément. La vitesse de transfert d'un réseau FDDI est comprise entre 155Mbits/s et 622Mbits/s.

e. Relais de trame

Un réseau à relais de trame est constitué de plusieurs entités de routage reliées les unes aux autres par des liaisons. Une trame peut être divisée et prendre différents chemins. Elle sera réassemblée à l'arrivée. La méthode d'accès à un réseau à relais de trame est le point à point. La vitesse dépend des supports matériels

10. Extension réseau

Pour répondre à l'accroissement des besoins de gestion de réseau d'une entreprise, vous devez augmenter la taille de son réseau ou améliorer les performances de ce dernier. Il ne suffit pas d'ajouter de nouveaux ordinateurs et d'avantage de câble pour étendre un réseau. Chaque topologie ou architecture réseau possède ses propres limites. Il est toutefois possible d'installer des composants pour augmenter la taille du réseau au sein de l'environnement existant. Les composants suivants permettent d'étendre un réseau :

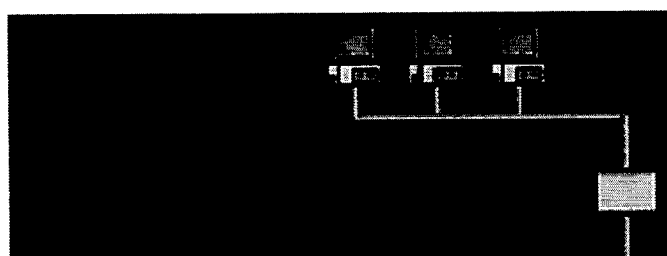
- Répéteurs et concentrateurs (hubs)
- Ponts
- Commutateurs
- Routeurs
- Passerelles

a. Répéteurs et concentrateurs

Les répéteurs et les concentrateurs (hubs) permettent d'étendre un réseau en lui ajoutant plusieurs segments de câble. Il s'agit de périphériques courants, peu coûteux et faciles à installer.

Répéteurs

L'installation d'un répéteur entre des segments de câble permet aux signaux de transiter sur de plus longues distances. Les répéteurs ne peuvent s'employer que pour communiquer entre réseaux similaires (par exemple deux réseaux Ethernet), cela est dû au fait que le répéteur ne convertit pas la transmission, mais ne fait que répéter la transmission électronique afin qu'elle puisse atteindre le réseau receveur. Le répéteur ne travaille qu'au niveau de la couche physique.



c. Commutateurs

Les commutateurs sont similaires aux ponts, mais ils offrent une connexion plus directe entre les ordinateurs sources et des destinations. Lorsqu'un commutateur reçoit un paquet de données, il crée une connexion interne séparée, ou segment entre deux de ses ports, et ne transmet le paquet de données qu'au port approprié de l'ordinateur de destination, en fonction des informations qui figure dans l'en-tête de chaque paquet. Cette connexion est ainsi isolée des autres ports et permet aux ordinateurs sources et de destination d'accéder à la totalité de la bande passante d'un réseau.

d. Routeurs

Un routeur est un périphérique qui joue le rôle de pont ou de commutateur, mais qui propose un plus grand nombre de fonctionnalités. Lorsqu'il transmet des données entre différents segments du réseau, le routeur examine l'en-tête de chaque paquet pour déterminer le meilleur itinéraire par lequel acheminer le paquet. Le routeur connaît l'itinéraire de tous les segments du réseau, grâce aux informations stockées dans sa table de routage. Le routeur utilisant les trois couches inférieures du modèle OSI

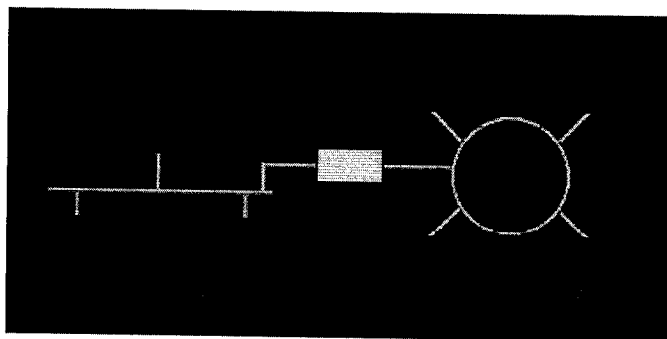


Figure 1.7 extension réseau en utilisant un routeur

e. Passerelles

Les passerelles permettent à des architectures réseau différentes de communiquer entre elles. Une passerelle joue le rôle d'un « interprète ». Par exemple, si deux groupes de personnes peuvent se parler mais n'utilisent pas la même langue, elles ont besoin d'un interprète pour communiquer. De même, deux réseaux peuvent être physiquement connectés, mais ils peuvent avoir besoin d'une passerelle pour traduire les données qu'ils s'échangent. La passerelle possède une pile complète des sept couches du modèle OSI.

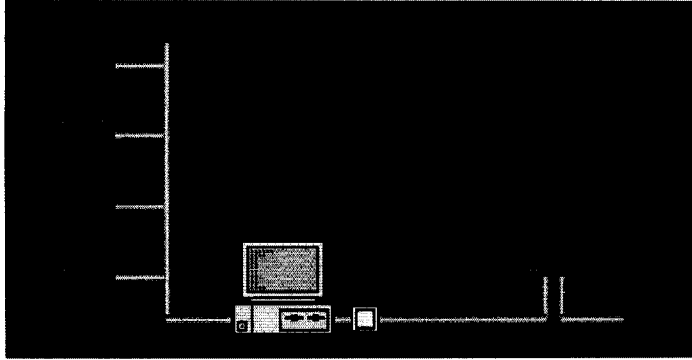


Figure 1.8 extension réseau en utilisant un routeur

11. Topologies des réseaux

La topologie d'un réseau décrit la disposition des ordinateurs, des câbles et des autres composants d'un réseau. Il s'agit d'une représentation graphique du réseau physique. Le type de topologie utilisée affecte le type et les capacités du matériel du réseau, sa gestion et ses possibilités d'extension. Il existe cinq principaux types de topologies :

a. Topologie en bus

Dans la topologie en bus, tous les ordinateurs sont reliés au même câble. Chaque extrémité est reliée à une terminaison. En cas de rupture du câble en un point, toutes les communications sont interrompues. A chaque extrémité du câble il est nécessaire d'avoir un bouchon terminateur. Plus le nombre d'ordinateurs sur le segment est élevé, plus l'efficacité du réseau diminue.

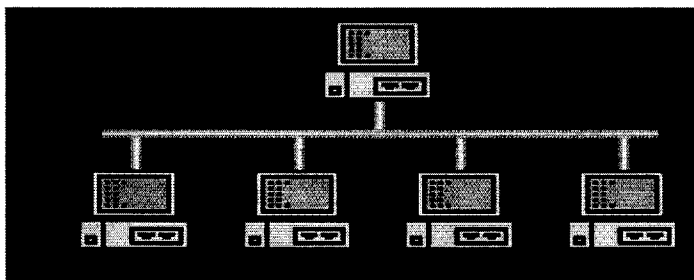


Figure 1.9 topologie bus

b. Topologie en étoile

Dans une topologie en étoile, tous les ordinateurs sont reliés à l'aide d'un câble à un concentrateur. Si l'un des câbles se rompt seul l'ordinateur relié à ce câble en est affecté, toutefois, si le concentrateur tombe en panne, l'ensemble des ordinateurs ne peut plus communiquer.

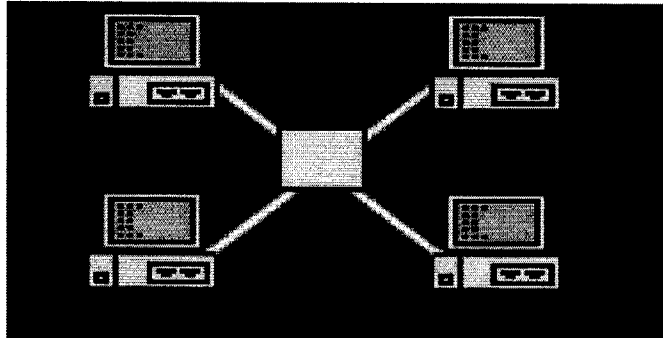


Figure 1.10 topologie étoile

c. Topologie en anneau

Dans une topologie en anneau, les ordinateurs sont reliés à un seul câble en anneau. Les signaux transitent dans une seule direction. Chaque ordinateur joue le rôle de répéteur, régénérant le signal, ce qui en préserve la puissance. Dans cette topologie, les ordinateurs « parlent » à tour de rôle. Un jeton circulant sur le réseau donne le droit d'émettre des données. Lorsqu'un ordinateur reçoit le jeton et qu'il souhaite « parler », il stock le jeton, puis envoie ses données, attend de recevoir la confirmation de réception envoyée par l'ordinateur destinataire, puis enfin, passe le jeton. Cette topologie est la plus efficace dans des réseaux où le trafic est élevé.

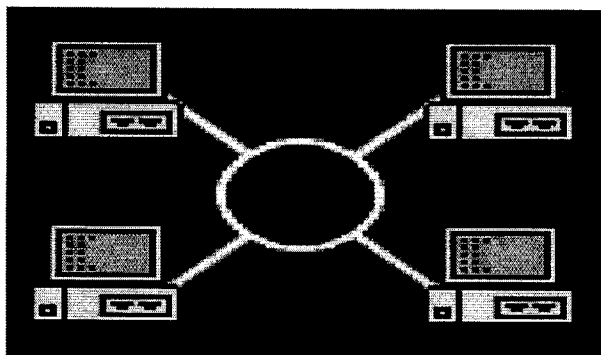


Figure 1.11 topologie anneau

d. Topologie maillée

Dans une topologie maillée, chaque ordinateur est connecté à chacun des autres ordinateurs par un câble séparé. Son principal avantage, est sa capacité de tolérance de panne. En effet, lorsqu'un câble se rompt, il existe de nombreux autres itinéraires routés. Cette topologie est toutefois très coûteuse.

e. Topologie hybride

Dans une topologie hybride, plusieurs topologies sont combinées. La topologie étoile/ bus et étoile/ anneau sont les plus répandues.

12. Modèle OSI

a. Présentation

Les systèmes de communication en réseau sont souvent décrits grâce au modèle de référence Open Systems Interconnection (OSI). Ce modèle a été développé par l'ISO (International Organization for Standardization). Le modèle OSI est constitué de 7 couches remplissant chacune une fonctionnalité particulière. L'illustration ci-après présente la structure de ce modèle :

Couche application
Couche présentation
Couche session
Couche transport
Couche réseau
Couche liaison de données
Couche physique

Tab 1.1 modèle OSI

b. Fonctions des couches du model OSI

Couche application : ensemble de services proposées aux utilisateurs : transfert de fichier, messagerie, appel de procédure distante,....

Couche présentation : mode de présentation des données échangées afin d'assures la compatibilité entre système non nécessairement identiques.

Couche session : gestion des dialogues et des transactions.

Couche transport : assemblage des paquets en message ou réciproquement désassemblage des messages en paquets et contrôle de l'acquittement des messages.

Couche réseau : routage de l'acheminement de paquets de données traversant les nœuds de réseau avec contrôle du flux pour éviter les engorgements.

Couche liaison : acheminement sans erreur de blocs d'information (trames) entre deux nœuds consécutifs du réseau.

Couche physique : transport physique de l'information considérée comme un train de bits.

13. Conclusion

Dans ce chapitre, nous avons abordé les notions de base des réseaux informatiques. Le chapitre suivant sera consacré à l'étude de la pile protocolaire tcp/ip.

Chapitre 2

Le protocole

TCP/IP

1. Introduction

Lorsque les systèmes multi-utilisateurs ont connu une stabilité satisfaisante, les recherches se sont orientées vers un protocole de communication de ces systèmes, entraînant diverses expérimentations dans le monde. C'est l'armée américaine qui accélèrera ce processus, avec le financement de programmes de recherches universitaires pour le développement de voies de télécommunications redondantes (ARPANet). En cas de conflit militaire, la bonne circulation de l'information et sa confidentialité sont majeures, et les liaisons de communication étaient les premières cibles militaires. Le standard des implémentations TCP/IP est issu des centres de Recherches en Systèmes Informatiques de l'Université de Berkeley (Californie, USA). Sa diffusion a été assurée par le système 4.X BSD. Aujourd'hui, les implémentations TCP/IP (*Transmission Control Protocol/Internet Protocol*) sont caractérisées par des versions majeures Net/X1 du *BSD Networking Software*.

2. Pile protocolaire TCP/IP

a. Définition

Le sigle TCP/IP signifie « Transmission Control Protocol/Internet Protocol ». Il provient des noms des deux protocoles majeurs de la suite de protocoles, c'est-à-dire les 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.

b. Modèle d'architecture

TCP/IP ne correspond pas exactement au modèle OSI. TCP/IP combine plusieurs couches OSI en une couche unique et n'utilise pas certaines couches. Le tableau suivant indique les couches de l'implémentation Solaris de TCP/IP. Les couches sont répertoriées de la couche la plus haute (application) à la couche la plus base (réseau physique).

Tableau 1-2 Pile de protocoles TCP/IP

OSI No. de couche	Réf. de	Couche équivalente	OSI	Couche TCP/IP	Exemples de protocoles TCP/IP	
5, 6, 7		Application, session, présentation		Application	NFS, NIS, DNS, LDAP, telnet, ftp, rlogin, rsh, rcp, RIP, RDISC, SNMP, etc.	
4		Transport		Transport	TCP, UDP, SCTP	
3		Réseau		Internet	IPv4, IPv6, ARP, ICMP	
2		Liaison données	de	Liaison données	de	PPP, IEEE 802.2
1		Physique		Réseau physique	Ethernet (IEEE 802.3), Token Ring, RS-232, FDDI, etc.	

Tab 2.1 protocole TCP/IP

Le tableau répertorie les couches de protocoles TCP/IP et les couches équivalentes dans le modèle OSI. Il indique également des exemples de protocoles disponibles à chaque niveau de la pile de protocoles TCP/IP. Chaque système impliqué dans une transaction de communication exécute une implémentation unique de la pile de protocoles.

o Couche physique

La couche physique spécifie les caractéristiques du matériel à utiliser pour le réseau. Par exemple, une couche réseau physique spécifie les caractéristiques physiques du média de communications. La couche physique de TCP/IP décrit les standards matériels

tels que IEEE 802.3, la spécification du média réseau Ethernet et RS-232, spécification dédiée aux connecteurs à broche standard.

- **Couche de liaison de données**

La couche de liaison de données identifie le type de protocole réseau du paquet, dans cette instance TCP/IP. En outre, cette couche de liaison de données assure le contrôle des erreurs et l'"encadrement". Par exemple, les encadrements Ethernet IEEE 802.2 et PPP (Point-to-Point Protocol, protocole point à point) constituent des protocoles de couche de liaison de données.

- **Couche Internet**

La couche Internet, également appelée couche réseau ou couche IP, accepte et distribue les paquets pour le réseau. Cette couche inclut le puissant protocole Internet (IP, Internet Protocol), le protocole ARP (Address Resolution Protocol, protocole de résolution d'adresse) et le protocole ICMP (Internet Control Message Protocol, protocole de message de contrôle Internet).

Protocole IP

Le protocole IP et les protocoles de routage associés sont sûrement les protocoles les plus importants de la suite TCP/IP. IP prend en charge les opérations suivantes :

- **Adressage IP** : les conventions d'adressage IP appartiennent au protocole IP. L'adressage IPv4 est décrit à la section Conception d'un schéma d'adressage IPv4 et l'adressage IPv6 est décrit à la section Présentation de l'adressage IPv6.
- **Communications d'hôte à hôte** : IP détermine le chemin qu'un paquet doit utiliser en fonction de l'adresse IP du système récepteur.
- **Formatage de paquet** : IP rassemble les paquets en unités appelées datagrammes. Les datagrammes sont décrits en détail à la section Couche Internet : préparation des paquets pour la distribution.

- **Fragmentation** : si un paquet est trop volumineux pour être transmis via le média réseau, le protocole IP sur le système émetteur scinde le paquet en fragments plus petits. Ensuite, le protocole IP du système récepteur réunit les fragments pour reconstituer le paquet d'origine.

Le SE Solaris prend en charge les formats d'adressage IPv4 et IPv6 décrits dans ce manuel. Pour éviter toute confusion lors de l'adressage du protocole IP, l'une des conventions ci-dessous est appliquée :

- Lorsque le terme IP est employé dans une description, la description s'applique à IPv4 et à IPv6.
- Lorsque le terme IPv4 est employé dans une description, la description s'applique exclusivement à IPv4.
- Lorsque le terme IPv6 est employé dans une description, la description s'applique exclusivement à IPv6.

Protocole ARP

Conceptuellement, le protocole ARP (Address Resolution Protocol, protocole de résolution d'adresse) existe entre la couche de liaison de données et la couche Internet. ARP permet à IP de diriger les datagrammes vers le système récepteur adéquat en mappant les adresses Ethernet (48 bits) vers des adresses IP connues (32 bits).

Protocole ICMP

Le protocole ICMP (Internet Control Message Protocol, protocole de message de contrôle Internet) détecte et signale les conditions d'erreur réseau. ICMP génère des rapports sur :

- **Paquets abandonnés** : paquets arrivant trop rapidement pour être traités ;
- **un échec de connectivité** : le système de destination est inaccessible ;
- **la redirection** : redirection d'un système émetteur vers un autre routeur.

○ **Couche transport**

La couche transport TCP/IP assure l'arrivée des paquets dans l'ordre et sans erreur, en échangeant les accusés de réception de données et en retransmettant les paquets

perdus. Cette communication est dite de type de bout en bout. Les protocoles de la couche transport à ce niveau sont TCP (Transmission Control Protocol, protocole de contrôle de la transmission), UDP (User Datagram Protocol, protocole de datagramme utilisateur) et SCTP (Stream Control Transmission Protocol, protocole de transmission de contrôle de flux). TCP et SCTP assurent des services de bout en bout fiables. UDP assure des services de datagramme peu fiables.

Protocole TCP :

TCP permet aux applications de communiquer les unes avec les autres comme si elles étaient physiquement connectées. TCP semble transmettre les données caractère par caractère, non sous forme de paquets individuels. Cette transmission s'effectue comme suit :

- point de départ, qui initialise la connexion ;
- transmission dans l'ordre des octets ;
- point d'arrivée, qui interrompt la connexion.

TCP joint un en-tête aux données transmises. Cet en-tête contient de nombreux paramètres qui facilitent la connexion des processus du système émetteur aux processus homologues du système récepteur.

TCP confirme l'arrivée du paquet à destination en établissant une connexion de bout en bout entre les hôtes émetteur et récepteur. TCP est donc considéré comme un protocole "fiable et orienté connexion".

Protocole SCTP :

SCTP est un protocole de couche transport fiable et orienté connexion. Il fournit aux applications les mêmes services que TCP. De plus, SCTP peut prendre en charge les connexions entre les systèmes possédant plusieurs adresses, ou multi réseau. La connexion SCTP entre les systèmes émetteur et récepteur est appelée association. Dans l'association, les données sont classées en blocs. Comme SCTP prend en charge les systèmes multi réseau, certaines applications, notamment des applications employées dans le secteur des télécommunications, doivent s'exécuter sur SCTP, non TCP.

Protocole UDP :

UDP assure la distribution de datagramme. UDP ne vérifie pas les connexions entre les hôtes émetteur et récepteur. Comme UDP élimine les processus d'établissement et de vérification des connexions, les applications qui envoient des petites quantités de données utilisent UDP.

o **Couche d'application**

La couche d'application définit les services Internet standard et les applications réseau à la disposition des utilisateurs. Ces services fonctionnent conjointement avec la couche transport pour assurer l'envoi et la réception de données. Il existe de nombreux protocoles de couche d'application. Des exemples de protocoles de couche d'application sont répertoriés ci-dessous :

- services TCP/IP standard, tels que les commandes ftp, tftp et Telnet
- commandes UNIX "r", telles que rlogin et rsh
- services de noms, tels que NIS et le DNS (Domain Name System, système de noms de domaine)
- services d'annuaire (LDAP, Lightweight Directory Access Protocol, protocole d'accès annuaire léger)
- services de fichier, tels que le service NFS
- SNMP (Simple Network Management Protocol, protocole de gestion de réseau simple), pour la gestion de réseau
- protocoles de routage RDISC (Router Discovery Server protocol, protocole serveur de détecteur de routeur) et RIP (Routing Information Protocol, protocole d'informations de routage).

3. système d'adressage

a. Le principe de base d'un réseau IP

IP est un réseau de transport de paquets en mode non fiable et non connecté. C'est à dire que le paquet peut être perdu dans le réseau, arriver dans le désordre voire en double. La fiabilité n'est assurée que par les couches de transport qui sont dans les ordinateurs d'extrémité. Les éléments intermédiaires du réseau sont des routeurs IP qui

vont servir d'aiguillage. Un routeur peut être arrêté sans que les liaisons passant par ce routeur en soit perturbées. Le réseau se reconfigure et les paquets seront acheminés par d'autres chemins. Rien ne garantit non plus que les paquets vont prendre le même chemin. On pourrait comparer cela au réseau postal. Deux enveloppes ne passeront pas forcément par le même centre de tri, et n'arriveront pas forcément en même temps. On appelle datagramme le paquet élémentaire. Celui-ci comme une enveloppe de courrier comprend une adresse de destination et une adresse de départ. Derrière les routeurs, on trouve des réseaux locaux, des liaisons spécialisées.

b. L'ADRESSE IP

L'adresse IP est constituée de 32 bits, soit 4 octets notés de façon décimale de 0 à 255, par ex 193.50.125.2. Une adresse est affectée non pas à une machine mais à une interface d'une machine. Celle-ci peut donc avoir plusieurs adresses. L'adresse se décompose en 2 parties, une partie réseau et une partie machine. Cet adressage n'est pas hiérarchisé dans le sens que 193.50.126.0 pourrait être un réseau japonais, alors que 193.50.125.0 serait un réseau français. C'est la très grosse faiblesse de cet adressage. Le successeur (IP V6) prévoit des hiérarchies d'adresses à la manière du téléphone. Chaque machine a une ou plusieurs adresses. Elle a obligatoirement une adresse IP par carte réseau. Elle peut aussi avoir plusieurs adresses sur une seule carte (IP ALIASING). C'est le cas des machines hébergeant plusieurs sites WEB.

c. LES DIFFERENTES CLASSES D'ADRESSES INTERNET

Pour des raisons administratives et de routage, on regroupe ces adresses sous forme de classes. On pourra ensuite utiliser ces adresses à sa guise pour gérer son réseau. Ces adresses sont demandées auprès du NIC17 (Network Information Center). Le NIC France (l'INRIA18) délègue la fourniture des adresses aux grands fournisseurs d'accès au réseau. Dans le cas de nos universités, toute nouvelle adresse doit être demandée à RENATER, organisme qui s'occupe du réseau de la recherche. RENATER a plusieurs réseaux de classe B et des blocs d'adresses de classe C, qu'il va morceler en sous réseaux pour ses utilisateurs. RENATER sera donc responsable du routage de

l'ensemble des classes B et classes C qui lui ont été attribués. Le détail de ce qui sera fait dans la classe B sera invisible de l'extérieur. En principe l'adressage comprend donc $256^3 \cdot 4$ adresses c.-à-d. : 4294967296 adresses (4 Milliards). En fait, on va voir qu'il y a beaucoup de pertes et que cet adressage est au bord de la saturation.

Les adresses sont regroupées en différentes classes pour des raisons d'administration et de routage. La partie machine est réservée à l'usage du gestionnaire du réseau qui peut redécouper cette partie, c'est-à-dire « subnetter ».

- Le réseau de classe A. Il peut contenir beaucoup de machines car l'adresse est sur 7 bits L'adresse du réseau est donc sur un octet dont la valeur la plus grande est un zéro par conséquent le premier chiffre sera inférieur à 128. Le classe A démarre à 0 jusqu'à 127

0	Réseau	Machine	Machine	Machine
---	--------	---------	---------	---------

- Le Classe B: adresse sur 14 bits: commence à 128.

10	Réseau	Réseau	Machine	
----	--------	--------	---------	--

- Le Classe C, le plus utilisé en ce moment, dû à la disparition des classes B devenues indisponibles par suite de manque d'adresses. Démarre donc à l'adresse 192.

110	Réseau	Réseau	Réseau	Machine
-----	--------	--------	--------	---------

- Le Classe D est utilisé pour des groupes de multicast Commence à 224.

1110	Réseau	Réseau	Réseau	Machine
------	--------	--------	--------	---------

- Le Classe E réservé pour usage futur, commence à 240.

1111	Réseau	Réseau	Réseau	Machine
------	--------	--------	--------	---------

4. Conclusion

Le protocole TCP / IP est en fait composé de plusieurs protocoles différents qui interagissent pour fournir la fonctionnalité que nous attendons de nos réseaux. Il est devenu le standard le plus adopté par le monde entier.

Dans le chapitre suivant, nous entament la programmation réseau avec les sockets.

1. Introduction

La notion de sockets a été introduite dans les distributions de Berkeley. Il s'agit d'un modèle permettant la communication inter processus (*IPC - Inter Process Communication*) afin de permettre à divers processus de communiquer aussi bien sur une même machine qu'à travers un réseau TCP/IP.

La communication par socket est souvent comparée aux communications humaines. On distingue ainsi deux modes de communication :

- Le mode connecté (comparable à une communication téléphonique), utilisant le protocole TCP. Dans ce mode de communication, une connexion durable est établie entre les deux processus, de telle façon que l'adresse de destination n'est pas nécessaire à chaque envoi de données.
- Le mode non connecté (analogue à une communication par courrier), utilisant le protocole UDP. Ce mode nécessite l'adresse de destination à chaque envoi, et aucun accusé de réception n'est donné.

2. Position des sockets dans le modèle OSI

Les sockets se situent juste au-dessus de la couche transport du modèle TCP/IP (protocoles UDP ou TCP), elle-même utilisant les services de la couche réseau (protocole IP / ARP).

Modèle des sockets	Modèle TCP/IP
Application utilisant les sockets	Application Présentation Session
UDP/TCP	Transport
IP/ARP	Réseau
Ethernet, X25, ...	Liaison Physique

Tab 3.1 les sockets dans l'OSI

3. Déroulement d'une communication avec les sockets

Comme dans le cas de l'ouverture d'un fichier, la communication par socket utilise un descripteur pour désigner la connexion sur laquelle on envoie ou reçoit les données. Ainsi la première opération à effectuer consiste à appeler une fonction créant un socket et retournant un descripteur (un entier) identifiant de manière unique la connexion. Ainsi ce descripteur est passé en paramètres des fonctions permettant d'envoyer ou recevoir des informations à travers le socket.

L'ouverture d'un socket se fait en deux étapes :

- La création d'un socket et de son descripteur par la fonction *socket()*
- La fonction *bind()* permet de spécifier le type de communication associé au socket (protocole TCP ou UDP)

Un serveur doit être à l'écoute de messages éventuels. Toutefois, l'écoute se fait différemment selon que le socket est en mode connecté (TCP) ou non (UDP).

- **En mode connecté**, le message est reçu d'un seul bloc. Ainsi en mode connecté, la fonction *listen()* permet de placer le socket en mode passif (à l'écoute des messages). En cas de message entrant, la connexion peut être acceptée grâce à la fonction *accept()*. Lorsque la connexion a été acceptée, le serveur reçoit les données grâce à la fonction *recv()*.
- **En mode non connecté**, comme dans le cas du courrier, le destinataire reçoit le message petit à petit (la taille du message est indéterminée) et de façon désordonnée.

Le serveur reçoit les données grâce à la fonction *recvfrom()*. La fin de la connexion se fait grâce à la fonction *close()*. Voici le schéma d'une communication en mode Connecté :

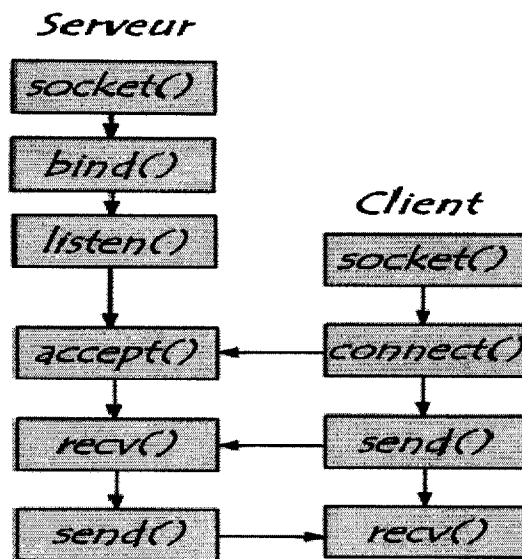


Figure 3.1 déroulement d'une communication en mode connecte

Voici le schéma d'une communication en mode non connecté:

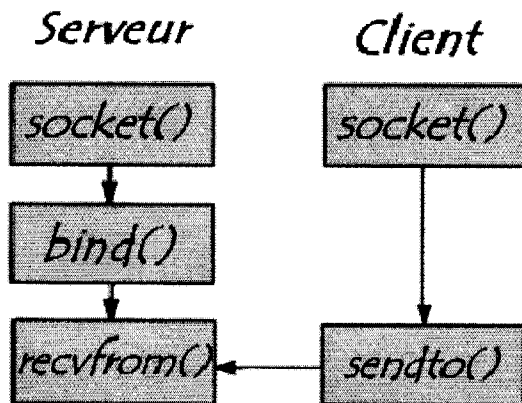


Figure 3.2 déroulement d'une communication en mode non connecte

4. Les fonctions des sockets

a. La fonction `socket()`

La création d'un socket se fait grâce à la fonction `socket()` :

`intsocket(famille, type, protocole)`

- **famille** représente la famille de protocole utilisé (*AF_INET* pour TCP/IP utilisant une adresse Internet sur 4 octets : l'adresse IP ainsi qu'un numéro de port afin de pouvoir avoir plusieurs sockets sur une même machine, *AF_UNIX* pour les communications UNIX en local sur une même machine)
- **type** indique le type de service (orienté connexion ou non). Dans le cas d'un service orienté connexion (c'est généralement le cas), l'argument *type* doit prendre la valeur *SOCK_STREAM* (communication par flot de données). Dans le cas contraire (protocole UDP) le paramètre *type* doit alors valoir *SOCK_DGRAM* (utilisation de datagrammes, blocs de données)
- **protocole** permet de spécifier un protocole permettant de fournir le service désiré. Dans le cas de la suite TCP/IP il n'est pas utile, on le mettra ainsi toujours à 0

b. La fonction `bind()`

Après création du socket, il s'agit de le lier à un point de communication défini par une adresse et un port, c'est le rôle de la fonction `bind()` :

`bind(intdescripteur,sockaddrlocaladdr,intaddrlen)`

- **descripteur** représente le descripteur du socket nouvellement créé
- **localaddr** est une structure qui spécifie l'adresse locale à travers laquelle le programme doit communiquer
- **addrlen** indique la taille du champ *localaddr*. On utilise généralement `sizeof(localaddr)`.

c. La fonction `listen()`

La fonction `listen()` permet de mettre un socket en attente de connexion.

La fonction `listen()` ne s'utilise qu'en mode connecté (donc avec le protocole TCP)

`int listen(intsocket,int backlog)`

- **socket** représente le socket précédemment ouvert
- **backlog** représente le nombre maximal de connexions pouvant être mises en attente

La fonction *listen()* retourne la valeur *SOCKET_ERROR* en cas de problème, sinon elle retourne 0.

d. La fonction *accept()*

La fonction *accept()* permet la connexion en acceptant un appel :

```
int accept(int socket, struct sockaddr * addr, int * addrlen)
```

- **socket** représente le socket précédemment ouvert (le socket local)
- **addr** représente un tampon destiné à stocker l'adresse de l'appelant
- **addrlen** représente la taille de l'adresse de l'appelant

La fonction *accept()* retourne un identificateur du socket de réponse. Si une erreur intervient la fonction *accept()* retourne la valeur *INVALID_SOCKET*.

e. La fonction *connect()*

La fonction *connect()* permet d'établir une connexion avec un serveur :

```
int connect(int socket, struct sockaddr * addr, int * addrlen)
```

- **socket** représente le socket précédemment ouvert (le socket à utiliser)
- **addr** représente l'adresse de l'hôte à contacter. Pour établir une connexion, le client ne nécessite pas de faire un *bind()*
- **addrlen** représente la taille de l'adresse de l'hôte à contacter

La fonction *connect()* retourne 0 si la connexion s'est bien déroulée, sinon -1.

f. La fonction *recv()*

La fonction *recv()* permet de lire dans un socket en mode connecté (TCP) :

```
int recv(int socket, char * buffer, int len, int flags)
```

- **socket** représente le socket précédemment ouvert
- **buffer** représente un tampon qui recevra les octets en provenance du client
- **len** indique le nombre d'octets à lire
- **flags** correspond au type de lecture à adopter :

- le flag *MSG_PEEK* indiquera que les données lues ne sont pas retirées de la queue de réception
- le flag *MSG_OOB* indiquera que les données urgentes (*Out Of Band*) doivent être lues
- le flag *0* indique une lecture normale

La fonction *recv()* renvoie le nombre d'octets lus. De plus cette fonction bloque le processus jusqu'à ce qu'elle reçoive des données.

g. La fonction *send()*

La fonction *send()* permet d'écrire dans un socket (envoyer des données) en mode connecté (TCP) :

```
int send(int socket, char * buffer, int len, int flags)
```

- **socket** représente le socket précédemment ouvert
- **buffer** représente un tampon contenant les octets à envoyer au client
- **len** indique le nombre d'octets à envoyer
- **flags** correspond au type d'envoi à adopter :
 - le flag *MSG_DONTROUTE* indiquera que les données ne routeront pas
 - le flag *MSG_OOB* indiquera que les données urgentes (*Out Of Band*) doivent être envoyées
 - le flag *0* indique un envoi normal

La fonction *send()* renvoie le nombre d'octets effectivement envoyés.

h. Les fonctions *close()* et *shutdown()*

La fonction *close()* permet la fermeture d'un socket en permettant au système d'envoyer les données restantes (pour TCP) :

```
int close(int socket)
```

La fonction *shutdown()* permet la fermeture d'un socket dans un des deux sens (pour une connexion full-duplex) :

```
int shutdown(int socket, int how)
```

- Si *how* est égal à 0, le socket est fermé en réception
- Si *how* est égal à 1, le socket est fermé en émission
- Si *how* est égal à 2, le socket est fermé dans les deux sens

close() comme *shutdown()* retournent -1 en cas d'erreur, 0 si la fermeture se déroule bien.

5. Architecture Client/serveur en java avec les sockets

a. Introduction

- Dans cette partie, nous allons apprendre tout ce qu'il faut savoir pour programmation réseau. Nous allons apprendre ce que sont les sockets et comment les débiter en manipuler, vous allez découvrir la création d'un client et celle d'un serveur.

b. Le client

Dans cette partie, nous allons maintenant traiter de la conception et la création d'un client en réseau avec les sockets. On va apprendre ce que sont les sockets, leur utilité et comment les utiliser, on va aussi apprendre à envoyer un mail avec lesdits sockets.

c. Les Sockets

Un Socket est une sorte de point D'ancrage pour les protocoles de transmission de données comme TCP/IP . Les Socket s ont des objets permettant la gestion de deux flux de données: un flux d'entrée (InputStream), garantissant la réception des données, et un flux de sortie (OutputStream), servant à envoyer les données. En Java, nous distinguerons deux types de Socket : les Socket simples (dits "clients") et les Socket serveurs. Un Socket client est tout simplement un Socket qui va se connecter sur un Socket serveur pour lui demander d'effectuer des tâches. Netscape utilise des Sockets clients par exemple...

Mais avant de passer à la pratique, analysons la classe Socket du package java.net de Java. Cette classe contient divers constructeurs dont seul un nous intéresse:

Socket(String host, int port)

Le constructeur du Socket attend deux arguments: une chaîne de caractères et un entier. Le premier argument définit l'adresse IP du serveur sur lequel nous désirons nous connecter. Cette adresse peut prendre la forme classique X.X.X.X (par exemple 127.0.0.1 pour votre propre machine) ou vous pouvez utiliser un nom (localhost est équivalent à 127.0.0.1). Le deuxième argument permet de définir le port sur lequel nous allons nous connecter. En effet, une même machine est tout à fait susceptible d'héberger plusieurs serveurs logiciels.

d. D'Autres usages

Ainsi, en utilisant les flux d'entrée et de sortie, il est possible, et ce très facilement, de faire communiquer deux logiciels. L'utilisation d'un Socket client se retrouve dans énormément d'applications comme ICQ, IRC, les browsers Web ou même les jeux !! De la sorte, rien ne vous empêche de créer votre propre client pour un autre serveur... cela ne vous demandera que de connaître le "protocole" du serveur convoité. Faites attention cependant aux flux que vous utilisez. Ici, nous avons encapsulé nos flux dans des objets rendant plus pratique l'envoi et la réception de chaînes de caractères (on aurait pu arguer en faveur de l'objet PrintWriter plutôt que OutputStreamWriter, mais la méthode println () de PrintWriter envoie le caractère de fin de ligne "\n" alors que SMTP attend "\r\n"). Or, certains serveurs, notamment les serveurs de jeux, sont extrêmement exigeants en termes de vitesse de réception et d'envoi. Dans ce genre de cas, il ne faut surtout pas utiliser des Strings mais plutôt des entiers (type int) voir des primitifs de type byte. Si vous vous retrouvez dans cette situation, conservez tout simplement les flux de base du Socket que vous récupérerez ainsi:

```
InputStream in = socket.getInputStream();
```

```
OutputStream out = socket.getOutputStream();
```

L'utilisation conjointe des méthodes read() (qui renvoie un entier) et write (byte) couvriront alors tous vos besoins.

6. Le serveur

a. Introduction

Avant d'aborder l'aspect technique du sujet, intéressons nous tout d'abord au fonctionnement général d'un serveur logiciel. Nous savons que les clients contiennent des routines ouvrant une connexion sur le port d'un serveur pour y demander et récupérer des informations. Un serveur logiciel aura donc pour but d'ouvrir un port sur la machine hôte et de gérer tous les clients se connectant à ce port. Cette gestion peut varier d'un type de serveur à un autre: simple réponse (un serveur HTTP), distribution aux autres clients (par exemple pour un jeu), mise en relation directe des clients (ICQ ou IRC en mode DCC), et ainsi de suite...

b. Le Socket serveur

A l'instar des Socket que nous nommerons "simples", c'est à dire les sockets clients, les sockets serveurs proposent un flux d'entrée et un flux de sortie. Pour employer un socket serveur en Java, il suffit d'utiliser une instance de la classe `ServerSocket` du package `java.net`. La classe `ServerSocket` offre trois constructeurs parmi lesquels un seul nous intéresse:

`ServerSocket(int port)`

le constructeur du `ServerSocket` n'attend qu'un seul et unique argument: un entier. Cet entier joue un peu le même rôle que pour les sockets simples. Il permet de spécifier un port. Mais dans ce cas, cet entier définira le numéro d'identification du port à créer. Toutes les applications réseaux utilisant des ports spécifiques.

7. Fonctions sur les sockets

Sommaire

- `socket_accept` — Accepte une connexion sur une socket
- `socket_bind` — Lie un nom à une socket
- `socket_clear_error` — Efface toutes les erreurs précédemment générées par une socket
- `socket_close` — Ferme une socket
- `socket_connect` — Crée une connexion sur une socket
- `socket_create_listen` — Ouvre une socket sur un port pour accepter les connexions
- `socket_create_pair` — Crée une paire de sockets identiques et les stocke dans un tableau
- `socket_create` — Crée une socket
- `socket_get_option` — Lit les options de la socket
- `socket_getpeername` — Interroge l'autre extrémité de la communication
- `socket_getsockname` — Interroge la socket locale
- `socket_last_error` — Lit la dernière erreur générée par une socket
- `socket_listen` — Attend une connexion sur une socket
- `socket_read` — Lit des données d'une socket
- `socket_recv` — Reçoit des données d'une socket connectée
- `socket_recvfrom` — Reçoit des données d'une socket, connectée ou pas
- `socket_select` — Exécute l'appel système `select()` un tableau de sockets avec une durée d'expiration
- `socket_send` — Envoie des données à une socket connectée
- `socket_sendto` — Envoie un message à une socket, qu'elle soit connectée ou pas
- `socket_set_block` — Met la socket en mode bloquant
- `socket_set_nonblock` — Sélectionne le mode non bloquant d'un pointeur de fichier
- `socket_set_option` — Modifie les options de socket
- `socket_shutdown` — Éteint une socket en lecture et/ou écriture
- `socket_strerror` — Retourne une chaîne décrivant un message d'erreur
- `socket_write` — Écrit dans une socket

8. Conclusion

Dans ce chapitre, nous avons présenté comment programmer avec les sockets. Un exemple de communication entre client et serveur a été fourni pour clarifier le fonctionnement et la communication en utilisant les sockets.

Le chapitre suivant sera consacré à la conception et la réalisation de notre application destiné au contrôle de trafic flux du réseau informatique.

Chapitre 4

Développement De L'application

1. Introduction

Il y a plusieurs langages de programmations réseaux. Dans ce travail, nous avons opté à travailler avec JPCAP à cause de sa simplicité, de plus il contient des bibliothèques complètes pour les applications réseaux qui suffit de les implémenter et exécuter.

Jpcap est applicable dans C et Java., il est basé sur Winpcap qui est un logiciel de filtrage intégré de nombreuses applications tel Windump. Il permet d'accéder aux couches réseaux à bas niveau du système, a fin d'en obtenir les statistiques ou de capturer à distance et de transmettre les paquets.

Grâce à sa librairie et ses contrôleurs, les développeurs ont pu développer des programmes de surveillance réseau, de détection d'intrusions, d'analyse de protocole ou de trafic, et bien d'autres encore.

2. Application de JPCAP

Ce qu'on peut développer avec JPCAP :

a. La sécurité

JPCP est un outil puissant dans le développement de :

- L'analyse des menaces potentielles ou réelles ;
- identification et l'analyse des vulnérabilités (audit, contrôle qualité...);
- évaluation des risques et la détermination du niveau de risque admissible.
- Elle se réalise par :
 - l'intégration d'outils et de services de sécurité système ou réseau (audit, contrôle d'accès, identification, logiciel antivirus, systèmes experts, noyau de sécurité...);
 - la validation logiciel/système (techniques formelles, tests statiques et dynamiques, etc.) ;
 - l'évaluation et la certification des systèmes et des produits.

b. Scanneur de port

Un scanneur est un programme qui balaye une plage de ports TCP ou UDP sur un ensemble de machines, afin d'établir la liste des couples machine/services ouverts. Le scan horizontal consiste à scanner un port sur un ensemble de machines, alors que le scan vertical consiste à scanner une plage de ports sur une même machine.

c. LES ANALYSEURS

Un analyseur est un programme qui capte tout le trafic qui circule dans le réseau. En partie, cette capture permet de déterminer la source et le destinataire de chaque trame (information) circulé dans le réseau. D'autre partie, elle détermine le protocole suivi par chaque trame.

d. LES SYSTEMES DE DETECTION D'INTRUSION(IDS)

Lorsqu'une attaque se produit, elle est invisible du point de vue humain puisqu'elle provoque seulement des modifications des schémas magnétiques des disques ou des schémas électroniques de la mémoire. Un système de détection d'intrusion, ou IDS (Intrusion Detection System), est un logiciel, et parfois un matériel, intéressant à plus d'un titre :

- Il consigne dans des journaux d'événements les données relatives aux activités de l'hôte et du réseau surveillé.
- Il fournit des outils automatiques pour générer des alarmes et des requêtes.
- Il propose également des outils de rapport pour aider les administrateurs à analyser les données de façon interactive pendant et après un incident.

Un IDS fonctionne comme une camera de sécurité dans un bâtiment. Il détecte des intrusions en temps réel et enregistre des bandes qui peuvent être étudiées après un incident. Mais ne remplace pas les dispositifs de protection visant empêcher les attaques. C'est un élément, parmi d'autres, de l'architecture de sécurité.

e. Moniteur de comportement

Un moniteur de comportement est un programme résidant que l'utilisateur charge à partir d'un fichier et qui reste alors actif à l'arrière plan, surveillant tout comportement inhabituel. C'est un programme qui permet de contrôler les actions suivantes :

- Les tentatives d'ouverture en lecture/ écriture des fichiers COM ou EXE
- Les tentatives d'écriture sur les secteurs de partition et de démarrage
- Les tentatives pour devenir résidant

f. Contrôleur d'intégrité

Les contrôleurs d'intégrités signalent toute modification intervenue dans un fichier. Schématiquement un contrôleur d'intégrité construit un fichier contenant les noms de tous les fichiers présents sur le disque auxquels sont associées quelques caractéristiques (Par exemple : la taille, la date et l'heure de la dernière modification).

3. Les algorithmes de l'application développée à analyser le flux :

Les étapes de notre algorithme sont comme suit :

- Obtenir la liste des interfaces réseau ;
- Ouvrir l'interface réseau ;
- Capturer les paquets d'interface réseau ;
- Filtrer les paquets capturés ;
- Sauvegarder les paquets capturés dans un fichier ;
- Lire les paquets à partir un fichier ;
- Transmettre les paquets vers une autre interface réseau.

a. Obtenir la liste des interfaces du réseau

Dans un premier temps, il faut obtenir la liste d'interface réseau, c'est-à-dire, sache tous les ports (nœud) structurés le réseau. (Scanner les ports)

b. Ouvrir l'interface réseau

Une fois obtenir la liste d'interface, on sélectionne le port à analyser et on établit un canal de transmission.

c. Capturer les paquets d'interface réseau

On capture les paquets à partir de l'interface sélectionné. Et on affiche les informations suivantes :

- Port source : indique le numéro de port du nœud d'origine (émetteur).
- Port de destination : indique le numéro de port du nœud de destination (récepteur).
- Protocole : identifie le type de protocole de la couche transport qui va recevoir le datagramme (par exemple, TCP ou UDP).
- Numéro de séquence : identifie la position du segment de données dans le flux des données déjà envoyées.
- Numéro de confirmation ou d'accusé de réception des données par un message envoyé à l'émetteur.
- Taille de fenêtre coulissante : indique combien de blocs de données l'ordinateur récepteur peut accepter.
- Données : comprend les données envoyées à l'origine par le nœud source.

d. Filtrer les paquets capturés

C'est un filtrage qui opère au niveau de la couche réseau et transport du modèle OSI. Cela consiste à accorder ou refuser le passage de paquet en se basant sur:

- L'adresse IP Source/Destination.
- Le numéro de port Source/Destination.
- Et bien sur les protocoles de niveau 3 ou 4.

Des instructions avec Jpcap permettent de filtrer les paquets capturés selon plusieurs facteurs (par exemple protocole, source d'origine...)

e. Sauvegarder les paquets capturés dans un fichier

La sauvegarde est un élément essentiel et même vital à celle-ci et pourtant ne fait pas l'objet d'assez d'attention dans beaucoup de cas. Nul n'est à l'abri d'une mauvaise manipulation, ou plus gravement d'un crash informatique provoquant la perte intégrale des données accumulées.

Ceci affecte sérieusement le fonctionnement même de réseau qui prend du temps à se relever d'un tel incident ou qui dans le pire des cas doit fermer ses portes.

Il est donc primordial d'établir une solution de sauvegarde afin de garantir la pérennité de l'information et d'assurer la continuité des activités du réseau.

Certaines questions doivent se poser afin de choisir un plan de sauvegarde :

- Volume des données à sauvegarder
- Type des données à sauvegarder
- Fréquence à laquelle les données doivent être sauvegardées
- Période de temps admis pour la sauvegarde
- Durée admise pour la restauration d'une sauvegarde
- Qualité de la sauvegarde
- Durée pendant laquelle les données doivent être sauvegardées
- Durée de vie des sauvegardes
- Environnement (à quel endroit seront effectuées les sauvegardes)
- Budget de l'entreprise

Le principe est de créer un espace de stockage à partir de plusieurs unités de disques. Ceci offre une plus grande rapidité de lecture/écriture.

Avec Jpcap, on peut sauvegarder les paquets capturés et filtrés dans un fichier sous forme binaire ainsi que pouvant les récupérer pour d'autre application.

Cet algorithme, fonctionne en deux étapes ; la première créer et ouvrir un nouveau fichier dans le but de sauvegarder les paquets. La deuxième consiste à fermer ce fichier.

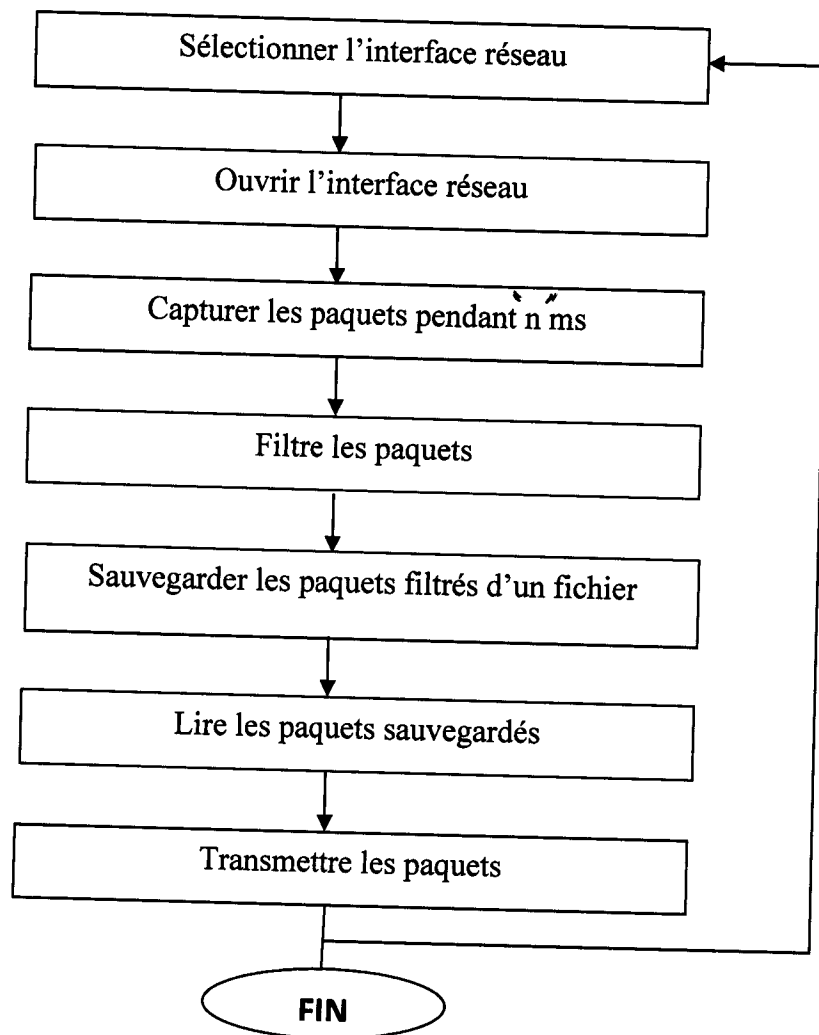
f. Lire Les paquets sauvegardés à partir d'un fichier

Des instructions de Jpcap permettent de lire les paquets sauvegardés à partir des fichiers sauvegardés.

g. Transmettre les paquets vers une autre interface réseau

Une fois les paquets récupérés et filtré et sauvegardés on peut les retransmettre vers un autre port.

L'organigramme suivant résume les étapes de l'algorithme ci-dessus :



4. Présentation de la simulation

Notre simulation est développée dans l'environnement NetBeans qui est :

a. Langage « NetBeans »

↓ Définition du « NetBeans »

NetBeans est un environnement de programmation permettant de développer des applications pour Windows et même pour Linux, c'est un outil moderne qui fait appel une conception visuelle des applications, à la programmation objet. De plus, il prend en charge la saisie automatique d'une partie du code source.

↓ Caractéristique de « NetBeans »

- Programmation objet.
- Outils visuels bidirectionnels.
- Compilateur produisant du code natif.
- Traitement complet des exceptions.
- Possibilité de créer des exécutables et des DLL.
- Bibliothèque de composants extensibles.
- Débogueur graphique intégré.

↓ Principes de développement en « NetBeans »

NetBeans fait évidemment partie de la famille de la programmation constructive. Les propriétés, entre autre, des objets sont immédiatement et toujours visibles aux programmeurs. Pour construire l'interface d'une application, il suffit avec NetBeans de placer des objets sur une fiche (fenêtre) et les personnaliser en modifiant éventuellement leurs propriétés et en leurs attachant des instructions liées à des événements données.

↓ Le concept de classe

Une classe est le support de l'encapsulation : c'est un ensemble de données et de fonctions regroupées dans une même entité. Une classe est une description abstraite

d'un objet. Les fonctions qui opèrent sur les données sont appelées des méthodes. Instancier une classe consiste à créer un objet sur son modèle. Entre classe et objet il y a, en quelque sorte, le même rapport qu'entre type et variable.

NetBeans est un langage orienté objet : tout appartient à une classe sauf les variables de type primitives. Pour accéder à une classe il faut en déclarer une instance de classe ou objet.

Une classe comporte sa déclaration, des variables et la définition de ses méthodes. Une classe se compose en deux parties : un en-tête et un corps. Le corps peut être divisé en 2 sections : la déclaration des données et des constantes et la définition des méthodes. Les méthodes et les données sont pourvues d'attributs de visibilité qui gère leur accessibilité par les composants hors de la classe.

✚ Les Objets

Les objets contiennent des attributs et des méthodes. Les attributs sont des variables ou des objets nécessaires au fonctionnement de l'objet. En NetBeans, une application est un objet. La classe est la description d'un objet. Un objet est une instance d'une classe.

Pour chaque instance d'une classe, le code est le même, seules les données sont différentes à chaque objet.

✚ Les méthodes

Les méthodes sont des fonctions qui implémentent les traitements de la classe.

b. La partie programmation

✚ Obtenir la liste des interfaces réseaux

Avant capturer les paquets d'interface réseau, il faut tout d'abord obtenir la liste des interfaces réseaux en utilisant la méthode « `getDeviceList()` » de la classe « `JpcapCaptor` » avec le retour à l'objet « `NetworkInterface` »

Le programme suivant permet d'obtenir la liste des interfaces réseaux et leurs adresse (l'adresse IP et MAC).

```

//Obtenir la liste des interface réseau
NetworkInterface[] Réseauliste = JpcapCaptor.getDeviceList();

//Pour chaque interface
for (int i = 0; i < Réseauliste.length; i++) {
    //Afficher le nom et la description de chaque interface
    System.out.println(i+": "+Réseauliste[i].name + "(" + devices[i].description+"");

    //Afficher la description et le type de liaison
    System.out.println(" datalink: "+Réseauliste[i].datalink_name + "(" +
devices[i].datalink_description+"");

    //Afficher l'adresse MAC
    System.out.print(" MAC address:");
    for (byte b :Réseauliste[i].mac_address)
        System.out.print(Integer.toHexString(b&0xff) + ":");
    System.out.println();

    //Afficher l'adresse IP, le masque et l'adresse de diffusion
    for (NetworkInterfaceAddress a : Réseauliste[i].addresses)
        System.out.println(" address:"+a.address + " " + a.subnet + " "+ a.broadcast);
}

```

Le résultat est le suivant

Sur Windows

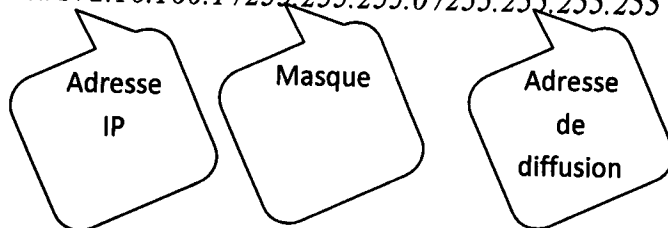
0: \Device\NPF_{C3F5996D-FB82-4311-A205-25B7761897B9}(VMware Virtual Ethernet Adapter)

data link: EN10MB(Ethernet)

adresse MAC: 0:50:56:c0:0:1:

adresse:/fe80:0:0:0:3451:e274:322a:fd9f null

addresses:/172.16.160.1 /255.255.255.0 /255.255.255.255



✚ Ouvrir l'interface réseau

Une fois obtenir la liste des interfaces réseaux, on peut choisir une interface réseau et l'ouvrir en utilisant la méthode « OpenDevice() » de la classe « JpcapCaptor ».

Le programme suivant explique comment ouvrir une interface réseau.

```
NetworkInterface[] Réseauliste = JpcapCaptor.getDeviceList();
int index=...; // déterminer l'interface réseau qu'on veut ouvrir
//ouvrir l'interface réseau(NetworkInterface intrface, int snaplen, boolean promics,
int to_ms)
JpcapCaptor captor=JpcapCaptor.openDevice(device[index], 65535, false, 20);
```

✚ Capturer les paquets d'interface réseau

Une fois ouvrir l'interface réseau, on peut capturer ses paquets.

Il y a deux techniques de capture les paquets en utilisant la classe « JpcapCaptor » sont : callback et la capture un par un.

On choisit la deuxième technique : « la capture un par un »

On utilise la méthode « getPacket() ».

Le programme suivant permet de capturer 10 paquets un par un.

```
JpcapCaptor captor=JpcapCaptor.openDevice(device[index], 65535, false, 20);

for(int i=0;i<10;i++){
    //capture les paquets et les afficher
    System.out.println(captor.getPacket());
}

captor.close();
```

✚ Filtre les paquets capturés

Avec Jpcap, on peut filtrer les paquets capturés selon des parametre (protocole par exemple).

Le programme suivant est un exemple de filtrage les paquets capturés selon le protocole TCP/IPV4 tel que permet de conserver seulement les paquets de ce protocole et les délivrer vers autre application.

```
JpcapCaptor captor=JpcapCaptor.openDevice(device[index], 65535, false, 20);
//filtrer les paquets capturés de protocole TCP/IP
captor.setFilter("ip and tcp", true);
```

✚ Sauvegarde les paquets dans un fichier

On peut sauvegarder les paquets dans un fichier sous forme binaire.

Il faut d'abord ouvrir un fichier en utilisant la méthode « OpenDumpFile » de la classe « JpcapWriter ». En suite, on sauvegarde les paquets de ce fichier en utilisant la méthode « WritePacket() » de la classe « JpcapWriter ». Et en fin,

On ferme le fichier en utilisant la methode « close() » de la classe « JpcapWriter ».

Le programme suivant est un exemple simple de capture et sauvegarde les 10 paquets capturés premiers.

```

JpcapCaptor captor=JpcapCaptor.openDevice(device[index], 65535, false, 20);
//ouvrir un fichier pour sauvegarder les paquets
JpcapWriter writer=JpcapWriter.openDumpFile(captor, "yourfilename");
for(int i=0;i<10;i++){
  //capture les paquets
  Packet packet=captor.getPacket();
  //save it into the opened file
  writer.writePacket(packet);
}
writer.close();

```

✚ Lire les paquets sauvegardés dans fichier

Avec Jpcap, on peut lire les paquets sauvegardés dans un fichier par ouvrir le fichier en utilisant la méthode «OpenFile()» de la classe «JpcapCaptor».

Le programme suivant est un exemple de lire et d'afficher des paquets sauvegardés dans un fichier.

```

//ouvrir le fichier pour lire les paquets
JpcapCaptor captor=JpcapCaptor.openFile("yourfilename");
while(true){
  //lire les paquets à partir d'un fichier
  Packet packet=captor.getPacket();
  // Si des erreurs sont détectés; la capture s'arrete.
  if(packet==null || packet==Packet.EOF) break;
  //sinon, afficher les paquets
  System.out.println(packet);
}
captor.close();

```

✦ Transmettre des paquets via d'interface réseau

On peut transmettre des paquets via d'interface réseau.

On a besoin d'ouvrir le fichier qui contient les paquets à transmettre en utilisant la méthode « Open Device() » de la classe « JpcapSender ». Une fois ouvrir le fichier, on appelle à la méthode « SenderPacket() » de la classe « JpcapSender » pour transmettre les paquets.

Le programme suivant est un exemple de transmettre des paquets de protocole TCP/IPV4 via d'interface réseau.

```
//ouvrir l'interface réseau pour transmettre ses paquets
JpcapSender sender=JpcapSender.openDevice(devices[index]);

//Créer le paquet TCP avec l'adresse du port et tous les drapeaux nécessaire
TCPpacket p=new
TCPpacket(12,34,56,78,false,false,false,false,true,true,true,true,10,10);

//specifier l'entete du IPv4
p.setIPv4Parameter(0,false,false,false,0,false,false,false,0,1010101,100,IPpacket.IP
PROTO_TCP,

InetAddress.getByName("www.univtlemcen.dz"),InetAddress.getByName("www.go
ogle.com"));

//insérer le champ de données dans le paquet
p.data=("data").getBytes();

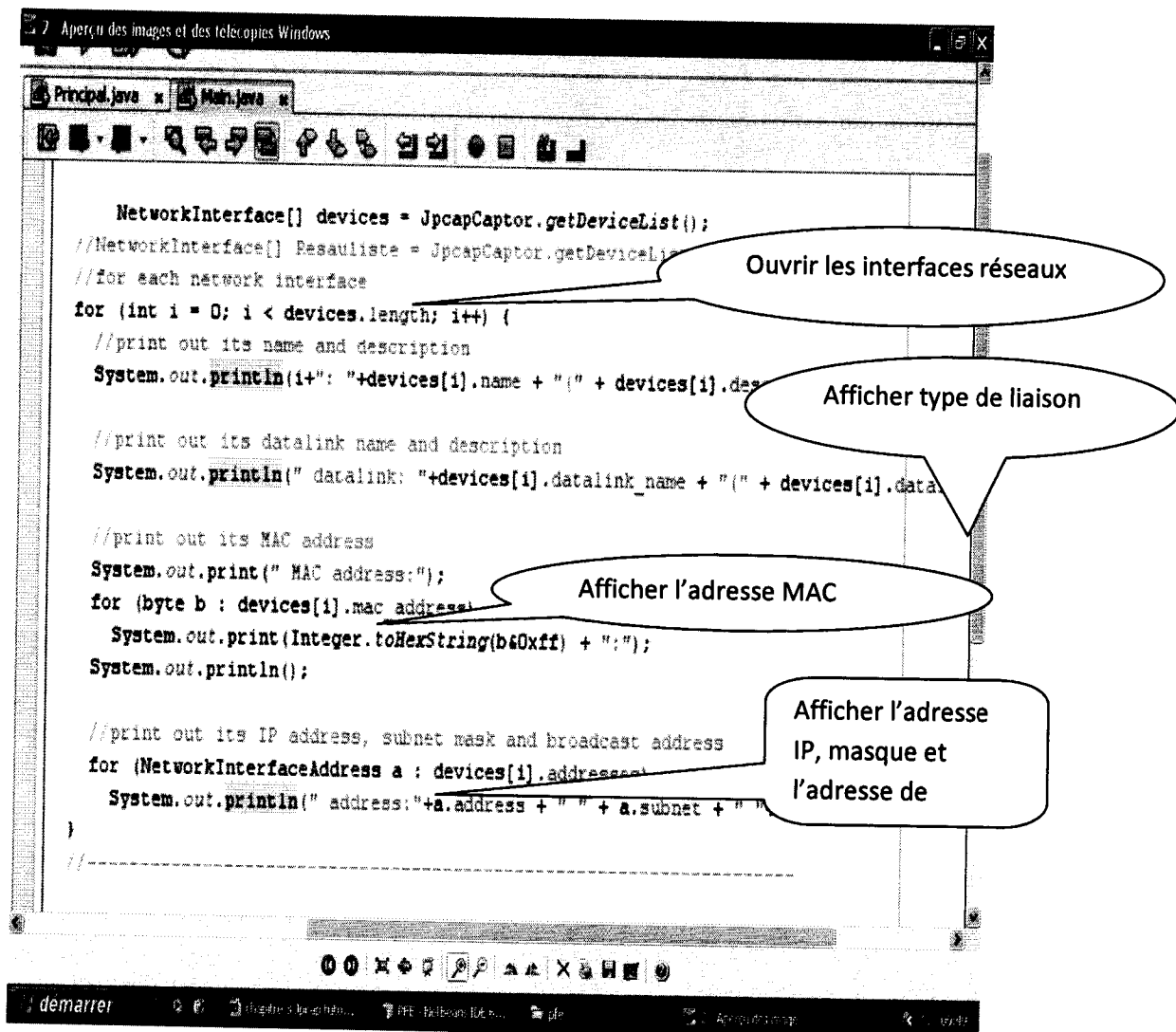
//Créer la trame ethernet
EthernetPacket ether=new EthernetPacket();
//Specifiez le type de liaison dans l'entete IP
ether.frameType=EthernetPacket.ETHERTYPE_IP;
//specifiez l'adresse MAC de la source et du destinataire
ether.src_mac=new byte[] {(byte)0,(byte)1,(byte)2,(byte)3,(byte)4,(byte)5};
ether.dst_mac=new byte[] {(byte)0,(byte)6,(byte)7,(byte)8,(byte)9,(byte)10};

//specifiez la liaison de donnée dans le paquet p comme Ether
p.dataLink=ether;

//envoyer le paquet p
sender.sendPacket(p);

sender.close();
```

La simulation de ce programme a abouti aux résultats suivants



The screenshot shows a Java IDE window titled 'Aperçu des images et des télécopies Windows'. The code in the editor is as follows:

```
NetworkInterface[] devices = JpcapCaptor.getDeviceList();
//NetworkInterface[] Pasauliste = JpcapCaptor.getDeviceList();
//for each network interface
for (int i = 0; i < devices.length; i++) {
    //print out its name and description
    System.out.println(i+": "+devices[i].name + "(" + devices[i].des

//print out its datalink name and description
    System.out.println(" datalink: "+devices[i].datalink_name + "(" + devices[i].data

//print out its MAC address
    System.out.print(" MAC address:");
    for (byte b : devices[i].mac address)
        System.out.print(Integer.toHexString(b&0xff) + ":");
    System.out.println();

//print out its IP address, subnet mask and broadcast address
    for (NetworkInterfaceAddress a : devices[i].addresses)
        System.out.println(" address: "+a.address + " " + a.subnet + " ")
}
//-----
```

Four callouts are present:

- Ouvrir les interfaces réseaux**: points to the `for (int i = 0; i < devices.length; i++)` loop.
- Afficher type de liaison**: points to the `System.out.println(" datalink: "+devices[i].datalink_name + "(" + devices[i].data` line.
- Afficher l'adresse MAC**: points to the `System.out.print(" MAC address:");` and the `for (byte b : devices[i].mac address)` loop.
- Afficher l'adresse IP, masque et l'adresse de**: points to the `for (NetworkInterfaceAddress a : devices[i].addresses)` loop.

Fig 4.1 : Ouvrir les Interfaces Réseaux

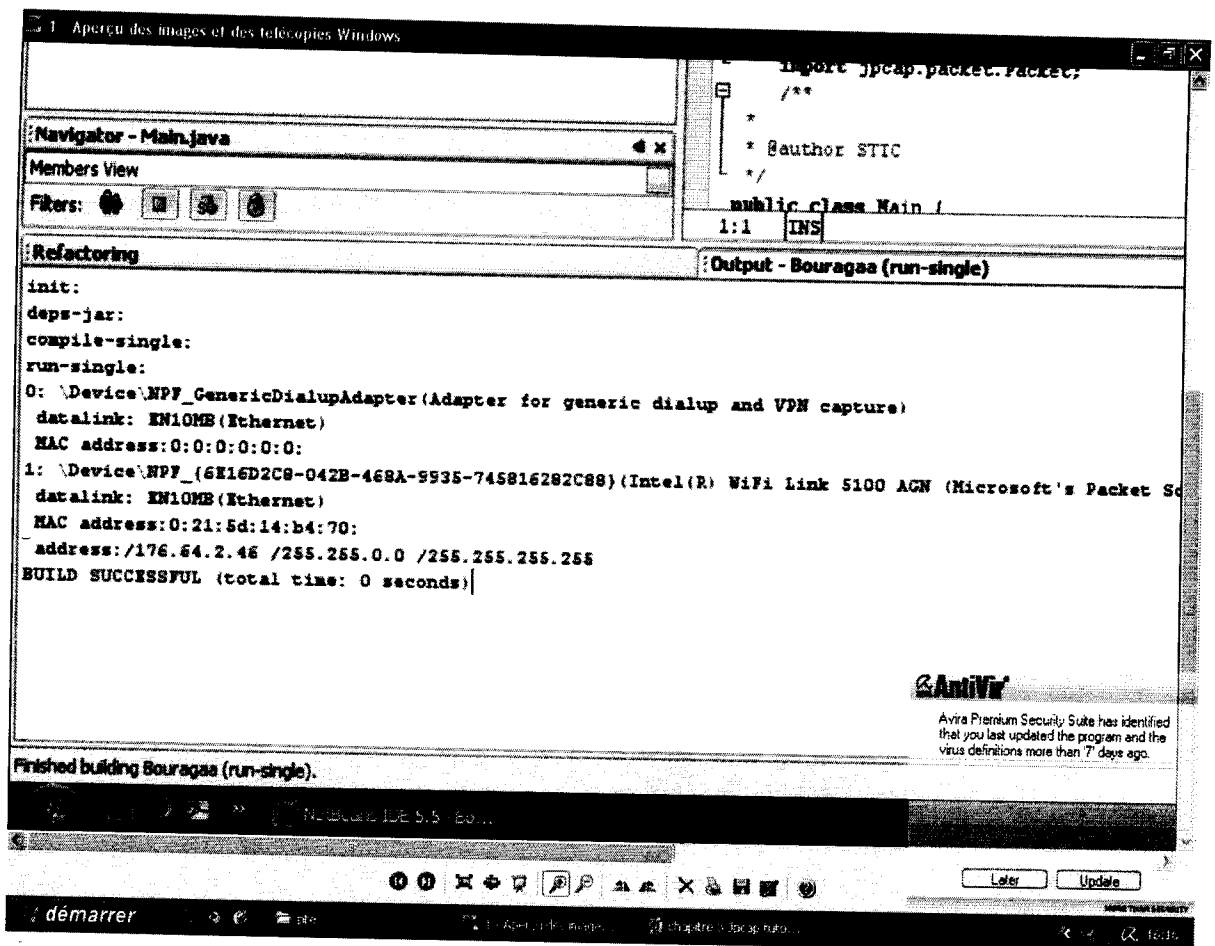


Fig 4.2 : Résultat d'ouvrir les interfaces réseau.

Cette partie affiche les interfaces réseaux et donne leurs descriptions et respective addresses (MAC, IP, masque, l'adresse de diffusion).

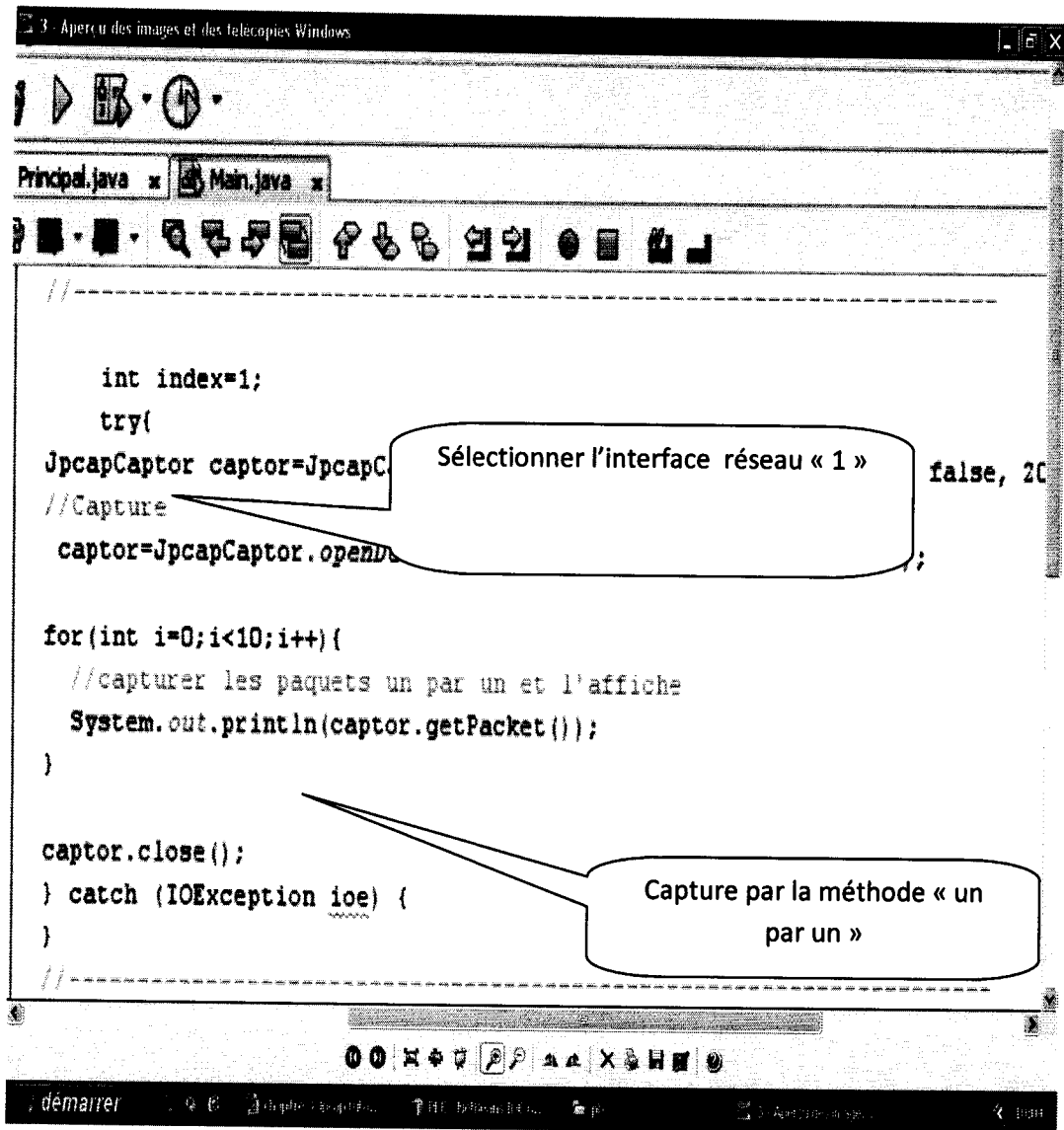


Fig 4.3: La sélection de l'interface 1 et capture les paquets

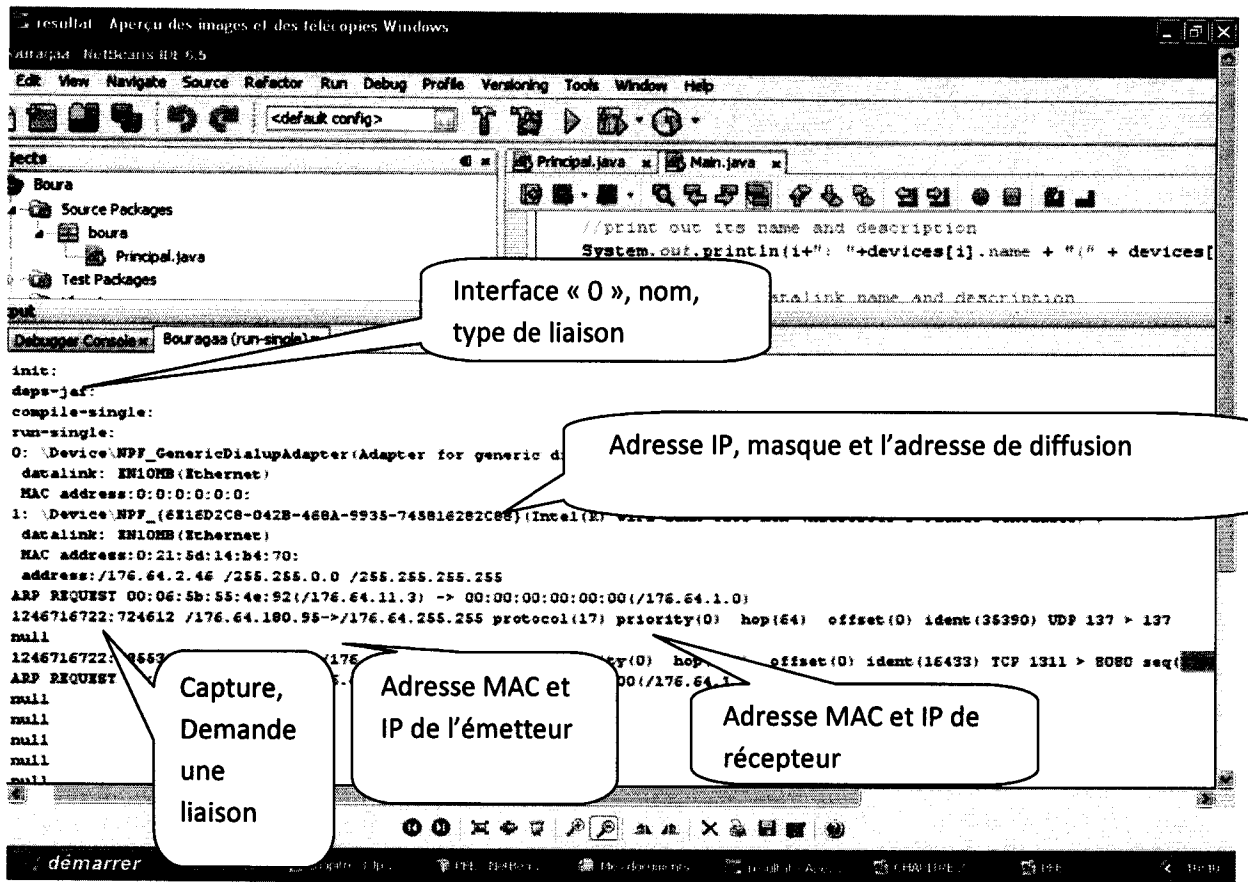


Fig 4.4 : Les résultats et leur interprétation

Comme mon PC n'est pas configure dans le réseau local du labo, je n'ai pas pu exécuter la requête ci-dessous, ou je demande l'adresse MAC d'un poste qui a l'adresse IP: 176.64.1.0), le résultat est nul car cet adresse IP n'est affecté à une adresse MAC:

```
ARP REQUEST 00:06:5b:55:4e:92(/176.64.11.3) 00:00:00:00:00:00(/176.64.1.0)
Signifie:
```

Je suis l'hôte «00:06:5b:55:4e:92», Est-ce que l'hôte possédant l'adresse IP 176.64.1.0 peut me retourner son adresse physique ?". La traduction de cette requête saisie grâce à Ethereal.

5. Conclusion

Dans ce chapitre, nous avons présenté l'API JPCAP permettant d'analyser le trafic réseau. Nous avons présenté également les résultats de l'exécution de notre application sur un réseau LAN ce qui nous a permis de contrôler et d'analyser tout le flux qui y circule.

Conclusion Générale

Conclusion générale

Ce présent travail consiste à développer une application permettant de contrôler et d'analyser tout le flux qui circule dans un réseaux TCP/IP. L'analyse de flux permet d'identifier les entités en cour de communication et ainsi le type des informations échangées. Cela permet de protéger ou bien de minimiser le risque des attaques qu'un réseau peut subir.

Pour protéger notre réseau, la mise en place d'un système de sécurité est indispensable.

L'application que nous avons développé joue un rôle d'un par feu et peut être implémentée dans une passerelle qui relie le réseau au monde extérieur. En connaissant les adresses IP sources et destinations, l'administrateur réseau décide de laisser ou ne pas laisser passer le flux, ainsi il aura la possibilité de détecter tout attaque sur le réseau.

Pour offrir un niveau de sécurité élevé dans un réseau, l'application que nous avons développé peu être utilisée avec autres systèmes de sécurité.

Références bibliographiques :

- [1] C. SERVIN, « Réseaux et Télécoms », Dunod, 2003.
- [2] P. SHIELDS et al, « Windows 2000 Server », Eyrolles, 2003.
- [3] Microsoft, « Réseau et système d'exploitation Microsoft Windows 2000 : notions fondamentales », 2000.
- [4] ED TITTEL, « Réseaux », Dunod, 2003.
- [5] S. LOHIER, D. PRESENT, « Transmissions et réseaux », Dunod,, 2003.
- [6] <http://www.alcove.fr> (15/05/2010)
- [7] <http://www.commentcamarche.net/contents/internet/tcpip.php3> (20/05/2010)
- [8] <http://www.africacomputing.org/pcours69.html> (22/05/2010)
- [9] <http://www.africacomputing.org/pcours69.html> (25/05/2010)
- [10] <http://docs.sun.com> (25/05/2010)
- [11] <http://www.commentcamarche.net/contents/internet/tcpip.php3> (29/05/2010)
- [12] <http://www.commentcamarche.net/contents/internet/tcpip.php3> (29/05/2010)
- [13] <http://gfx.developpez.com/tutoriel/java/network/> (29/05/2010)
- [14] www.php.net/manual/fr/function.socket-bind.php (29/05/2010)

Liste de figures et tableaux:

Liste de figure

Figure 1.1 paire torsadée	8
Figure 1.2 câble coaxial	8
Figure 1.3 câble à fibre optique.....	8
Figure 1.4 transmission asynchrone	10
Figure 1.5 extension réseau en utilisant un répéteur.....	14
Figure 1.6 extension réseau en utilisant un pont.....	14
Figure 1.7 extension réseau en utilisant un routeur.....	15
Figure 1.8 extension réseau en utilisant un routeur.....	16
Figure 1.9 topologie bus.....	16
Figure 1.10 topologie étoile.....	17
Figure 1.11 topologie anneau.....	17
Figure 3.1 déroulement d une communication en mode connecte	30
Figure 3.2 déroulement d une communication en mode non connecte	30
Fig 4.1 : Ouvrir les Interfaces Réseaux.....	51
Fig 4.2 : Résultat d'ouvrir les interfaces réseau.....	52
Fig 4.3 : La sélection de l'interface 1 et capture les paquets.....	53
Fig 4.4 : Les résultats et leur interprétation.....	54

Liste des tableaux

Tab 1.1 modèle OSI.....18

Tab 2.1 protocole TCP/IP.....21

Tab 3.1 les sockets dans l'OSI.....28