Université de Ain Temouchent –Belhadj Bouchaib
Faculté des Sciences et de la Technologie

Département: Mathématiques & Informatique

Laboratoire: Recherche en Informatique Industrielle et Réseaux (RIIR)
Oran 1 Ahmed Ben Bella

Polycopié pédagogique

Dr. A. BENZERBADJ
Maître de Conférences Rang B

Architecture des Réseaux

Cours destiné aux étudiants
de:

Master : 1ère année RID & CYSIA

Année: 2024-2025

Préface

Ce polycopié se veut être une aide pédagogique destinée à approfondir la compréhension des fondements des réseaux des étudiants en Master I RID (Réseaux et Ingénierie des Données) et Master I CYSIA (Cyber-Sécurité et Intelligence Artificielle). Nous tenons à rappeler que les étudiants (es) de ces deux Master ont déjà croisé le chemin des réseaux en L2 et par conséquent ils ont acquis des connaissances dans ce domaine complexe qui ne pourrait être présenté exhaustivement dans un polycopié.

Les cours de ce polycopié ont été élaborés en s'inspirant de multiples sources documentaires, notamment des polycopiés, des livres tels que [1], des sites web, des vidéos, etc.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Introduction Générale

Objectifs :

Ce polycopié de cours sur l'architecture des réseaux a pour objectif de fournir une base solide en matière de conception, d'organisation et de fonctionnement des réseaux informatiques. Il vient renforcer les connaissances déjà acquises en deuxième année de licence (L2) dans ce domaine en évolution continue.

Public visé :

Ce cours s'adresse principalement aux étudiants de Master 1, spécialités RID et CYSIA, qui souhaitent approfondir leurs connaissances en réseaux informatiques.

Requis :

Ce cours nécessite une bonne compréhension des concepts fondamentaux en réseaux informatiques, tels que les modèles OSI et TCP/IP, acquis lors des cours de deuxième année de licence (L2). Les étudiants doivent aussi avoir une bonne maîtrise de l'adressage IPv4 et être familiers avec des outils d'analyse et de simulation de réseaux tels que Wireshark, Packet Tracer ou GNS3, afin de bien suivre le contenu du cours.

Structure du polycopié :

À travers ce document, nous explorerons les différents concepts clés, notamment les modèles OSI et TCP/IP, le routage, la fragmentation des paquets, ainsi que les protocoles tels que TCP, IPv_4, IPv_6, Ethernet, ARP, RARP, ICMP et DHCP. Nous aborderons également le concept de NAT (Network Address Translation) et les VLANs.

Ce polycopié est structuré en 12 chapitres, chacun axé sur un thème précis parmi ceux énumérés ci-dessus. Cette organisation modulaire facilite, à notre avis, l'apprentissage progressif et l'intégration des différents éléments constitutifs de la matière "Architecture des Réseaux".

Une Conclusion générale clôture le polycopié.

Chapitre 1

Adressage IPv_4

1.1 Introduction

Les adresses IPv_4, dites aussi adresses logiques, permettent d'identifier de manière unique chaque périphérique du réseau. Elles permettent un routage efficace des données à travers le réseau mondial. Ces adresses, représentées sous forme de 4 octets séparés par des points (notation décimale pointée), constituent le langage universel qui permet aux appareils de se localiser mutuellement et de communiquer. L'épuisement progressif des adresses IPv_4, dû à la croissance exponentielle des dispositifs connectés, a donné lieu à l'exploration de plusieurs solutions, entre autres l'adressage sans classes, le VLSM et la transition vers IPv_6.

1.2 Rappel sur les modèles en couches

1.2.1 Modèle OSI

Le modèle OSI (Open Systems Interconnexion, en Français interconnexion de systèmes ouverts) est constitué de 7 couches distinctes. Dans chacune de ces couches opèrent un certain nombre de protocoles. Le modèle en question a été créé en 1978 par l'organisation internationale pour la standardisation (en Anglais, International Organization for Standardization (ISO)). Pour se rappeler les 7 couches du modèle OSI, **de bas en haut** et **de haut en bas** en Anglais, il faut se rappeler respectivement des deux phrases suivantes :

1. **P**lease **D**o **N**ot **T**hrow **S**ausage **P**izza **A**way, ce qui signifie : "S'il vous plaît, ne jetez pas les saucisses de pizza" où P représente *physique* (physical en Anglais), D *liaison de données* (data link en Anglais), N *réseau* (network en Anglais), T *transport*, S *session*, P *présentation* (presentation en Anglais) et A *application*.
2. **A**ll **P**eople **S**eem **T**o **N**eed **D**ata **P**rocessing, ce qui signifie : "Tout le monde a besoin du traitement de données" où A représente *application*, P *présentation* (presentation en Anglais), S *session*, T *transport*, N *réseau* (network en Anglais), D *liaison de données* (data link en Anglais) et P *physique* (Physical en Anglais).

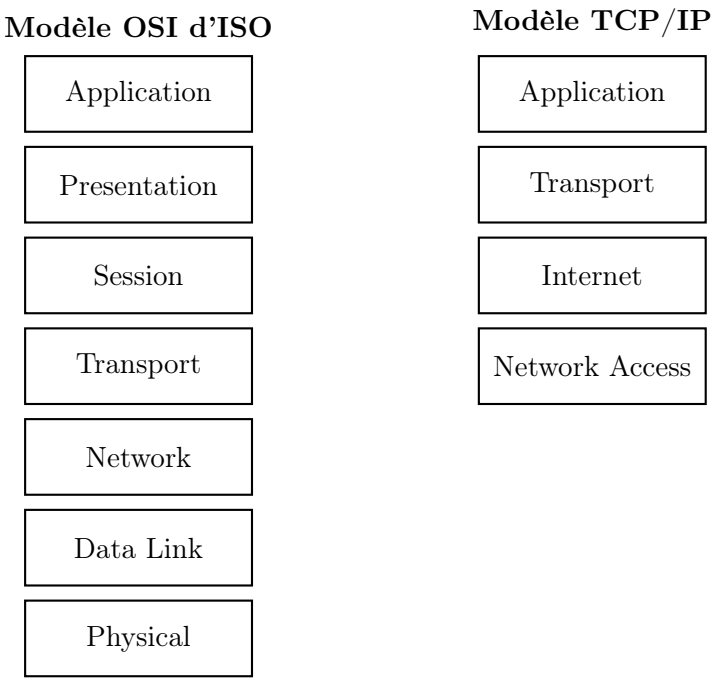


FIGURE 1.1 – Comparison entre le modèle OSI d'ISO et le Modèle TCP/IP

Les couches basses du modèle OSI s'intéressent au transport de l'information, tandis que les couches hautes correspondent à leur traitement. Une couche définit des fonctionnalités qui sont réalisées par un protocole associé à la couche. Chaque couche rend un service à la couche située au-dessus. Autrement dit, chaque couche se sert de la couche sous-jacente pour réaliser sa fonction.

1.2.2 Modèle TCP/IP

Le modèle TCP [1]/IP [2] a été créé dans les années 1970 par le département de la défense des États Unis d'Amérique, plus précisément par l'agence DARPA [3]. C'est d'ailleurs pour cette raison qu'une autre appellation lui est associée, à savoir DoD [4] Model. Le modèle en question est constitué de 4 quatre couches seulement.

1.3 Adressage Internet Protocol version 4 (IPv_4)

— Une adresse IPv_4, dite aussi adresse *logique*, est un numéro d'identification constitué de quatre octets, qui est attribué de façon permanente ou provisoire à chaque carte réseau utilisant le protocole IPv_4.

1. Transmission Control Protocol
2. Internet Protocol
3. Defense Advanced Research Projects Agency
4. Department of Defense

— Une adresse IPv_4 permet d'identifier chaque hôte connecté à un réseau informatique utilisant le protocole IPv_4.

— L'adresse IPv_4 est assignée soit manuellement par l'administrateur du réseau local, soit automatiquement via le protocole DHCP [5]. Si un composant dispose de plusieurs interfaces, chacune de ces interfaces dispose d'une adresse IPv_4 spécifique.

— Les adresses IPv_4 sont généralement écrites sous forme décimale pointée (dotted-decimal notation), c.-à-d. quatre entiers de 0 à 255 chacun, séparés par des points. Par exemple :

  — L'adresse IPv_4 10.0.0.1 est sous forme décimale pointée.

  — L'adresse IPv_4 précédente est notée en binaire : 00001010.00000000.00000000.00000001

— La suite des 4 octets d'une adresse IPv_4 est divisée en deux parties :

  — Net-Id : désigne l'adresse d'un réseau (*Network Identifier*). le Net-Id est obtenu en assignant 0 à chaque bit de la partie constituant le Host-Id.

  — Host-Id : désigne l'adresse d'une machine hôte (*Host Identifier*) sur le réseau désigné par Net-Id. le Host-Id est obtenu en assignant 0 à chaque bit de la partie constituant le Net-Id.

1.4 Classes d'adresses IPv_4

— Les adresses IPv_4 sont réparties en cinq classes. Les champs Net-Id et Host-Id ont des longueurs variables qui dépendent de la classe de l'adresse IPv_4. Il est à noter que cette répartition en classes est devenue par la suite obsolète en raison de la pénurie d'adresses IPv_4 qui a résulté de la croissance rapide de l'utilisation d'internet. Le système de répartition en classes a été remplacé par l'adressage sans classes (CIDR [6]) au milieu des années 90 (voir Section 1.8) :

1. Classe A (grands réseaux) : Ce sont les adresses dont le bit de poids fort est à 0. Le Net-Id est codé sur le premier octet, et les trois derniers octets représentent le Host-Id.

   — **Plage : de 1.0.0.0 à 126.255.255.255**. J'ai délibéremment exclu les réseaux 0.0.0.0 et 127.0.0.0 dont les adresses représentent respectivement l'absence d'une adresse IP ou la route par défaut et le réseau des adresses dites de boucle locale (en Anglais, loopback addresses).

2. Classe B (réseaux moyens) : Ce sont les adresses dont les deux bits de poids fort sont 10. Le Net-Id est codé sur 2 octets, et le Host-Id sur les deux autres octets.

   — **Plage : de 128.0.0.0 à 191.255.255.255**.

3. Classe C (petits réseaux) : Ce sont les adresses dont les trois bits de poids forts sont 110. Le Net-Id est alors codé sur les 3 premiers octets et le Host-Id sur le dernier octet.

5. Dynamic Host Configuration Protocol
6. Classless Inter-Domain Routing

— **Plage : de 192.0.0.0 à 223.255.255.255**.

4. Classe D (multicast [7]) : Ce sont les adresses dont les quatre bits de poids forts sont 1110. Les 28 bits qui restent désignent dans ce cas un groupe de multicast. Les adresses appartenant à cette classe sont utilisée pour l'envoi de messages en multicast. Par exemple l'adresse 224.0.0.9 est utilisée par le protocole RIPv2 [8].

   — **Plage : de 224.0.0.0 à 239.255.255.255**.

5. Classe E (expérimentale) : Ce sont les adresses dont les quatre bits de poids fort sont 1111. C'est une classe réservée pour les expérimentations.

   — **Plage : de 240.0.0.0 à 247.255.255.255**.

Les Figures 1.2 et 2.5, ainsi que le Tableau 2.1 illustrent tout ce qui a été expliqué ci-dessus.

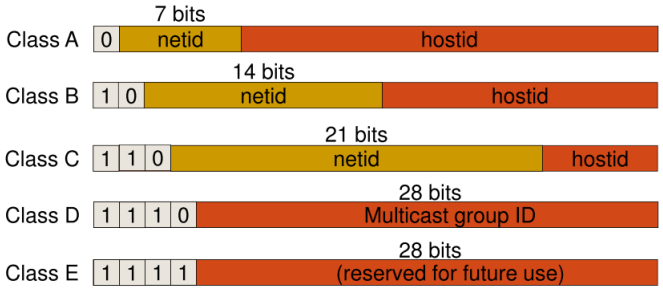


FIGURE 1.2 – Bits de poids fort du premier octet des classes A, B, C, D et E [2].

TABLEAU 1.1 – Répartition en classes de l'espace d'adressage IPv_4.

Classe	Bits de poids fort du 1^{er} octet	Plage décimale du 1^{er} octet	Adresse réseau et hôte (R=Réseau, H=Hôte)	Masque S/réseau par défaut	Nombre de réseaux	Nombre d'hôtes par réseau (adresses utilisables)
A	0	1-126	R.H.H.H	255.0.0.0	126	16 777 214 (2^{24}-2)
B	10	128-191	R.R.H.H	255.255.0.0	16.384	65 534 (2^{16}-2)
C	110	192-223	R.R.R.H	255.255.255.0	2.097.152	254 (2^8-2)
D	1110	224-239	Prévue pour faire du "multicast" ou multipoint			.
E	1111	240-247	Réservée à un usage futur			

1.5 Adresses IPv_4 réservées

Certaines adresses IPv_4 sont réservées pour des usages particuliers :

7. L'adressage multicast est employé pour s'adresser en une seule fois à un groupe d'hôtes
8. Routing Information Protocol version 2

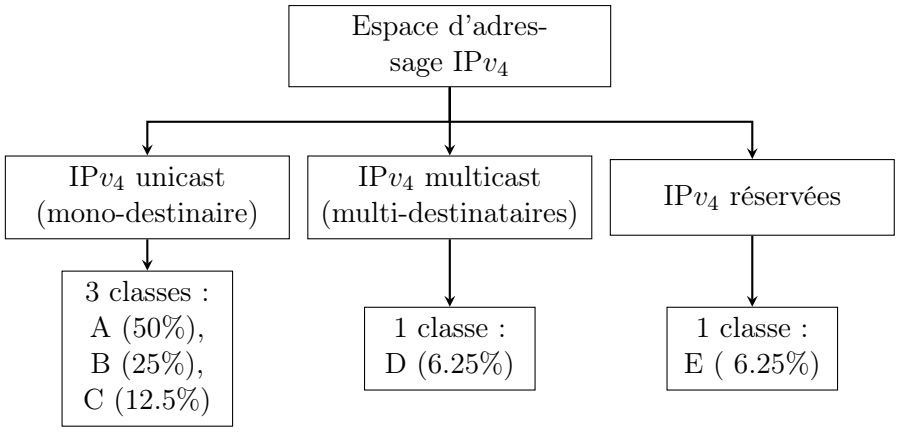


FIGURE 1.3 – Répartition des classes par type de communication.

1. La plage d'adresses **APIPA** [9] est la plage allant de 169.254.0.1 à 169.254.255.254. Il s'agit d'une plage d'adresses utilisées lorsqu'une interface configurée en tant que client DHCP n'obtient pas de réponse d'un serveur DHCP. L'objectif initial est de permettre à des hôtes dont le réseau ne dispose pas d'un serveur DHCP de communiquer entre eux. Il est à noter que les adresses APIPA ne sont utilisées qu'en cas de problème :

   — APIPA est donc un processus qui permet à un système d'exploitation de s'attribuer automatiquement une adresse 169.254.x.y/16 lorsque le serveur DHCP est hors service ou injoignable. La plupart du temps, les cartes réseaux sont configurées pour emettre à leur montage des reqêtes pour établir une communication avec un serveur DHCP. Cependant, quand la carte ne parvient pas à trouver une réponse à ses requêtes, elle choisit alors de se paramétrer en adressage APIPA.

   — Les plages d'adresses APIPA ne sont pas routables sur Internet et sont exclusivement dédiées à des communications locales.

   — C'est un réseau privé qui n'est routable ni sur internet ni ailleurs. Lorsqu'un hôte dispose d'une adresse APIPA, il ne pourra communiquer qu'avec d'autres hôtes configurés en APIPA (communication au niveau du même switch).

   — Pour que deux hôtes ne s'affectent la même adresse APIPA, une fois qu'un hôte s'est affecté une adresse APIPA (en choisissant deux chiffres aléaroires x et y), il envoie une requête ARP [10] (request ARP) aux autres hôtes pour leur demander s'ils n'ont pas la même adresse. Si oui, il choisit deux autres nombres aléatoires x et y et il s'assure qu'ils ne sont pas déjà pris (request ARP), etc.

   — Le service APIPA continue de vérifier la présence d'un serveur DHCP dans les environs et ce toutes les 5 minutes. Il réemet une requête de type DHCP Discover en espérant avoir une réponse d'un serveur. S'il obtient une réponse, le service DHCP affectera une adresse IP à la carte réseau et remplacera l'IPv_4 APIPA en place.

9. Automatic Private IPv_4 Addressing
10. Address Resolution Protocol

— Notons que pour désactiver/activer APIPA sous Linux(Debian) :
   (a) Modifier le fichier /etc/default/avahi-daemon, en mettant AVAHI_DAEMON_DETECT_LOCAL à 0
   (b) Redémarrer le daemon avahi via la commande : /etc/init.d/avahi-daemon restart

2. Le réseau **127.0.0.0** est réservé pour les tests de boucle locale (en Anglais, loopback) avec notamment l'adresse IP 127.0.0.1 qui est l'adresse "localhost", c.-à-d., de boucle locale de votre machine :
   — Les adresses commençant de 127.0.0.1 à 127.255.255.255 sont réservées pour le bouclage (loopback). Elles permettent de tester la pile de protocoles TCP/IP locale d'une machine sans passer par un réseau, c.-à-d., un paquet envoyé sur l'interface loopback est envoyé vers la machine elle même sans être transmis sur le réseau.
   — Par exemple, sur votre machine, l'adresse de loopback (127.0.0.1), correspond à une interface fictive présente sur toutes les machines. L'interface de loopback (lo ou lo0) permet à la machine de s'envoyer à elle-même des paquets. On peut ainsi utiliser des applications réseau sans disposer d'interface physique, ou sans qu'elle soit reliée au réseau. On peut par exemple installer les programmes client et serveur d'une application dans la même machine pour aller faire une démonstration chez un client, ou encore travailler chez soi à la conception d'un site Internet, etc.
3. **0.0.0.0** : valeur qui indique soit l'absence d'une adresse IPv_4, soit la route par défaut.
4. **255.255.255.255** : adresse de diffusion limitée (limited broadcast). Dans ce type de broadcast, le routeur du réseau local (LAN [11]) ne transmet le paquet diffusé à aucun de ses autres segments de réseau connectés. Ainsi, la diffusion est limitée au sein du LAN.
5. **L'adresse de broadcast associée à un réseau qui correspond à une partie hôte de l'adresse en question dont tous les bits sont à 1**. Il s'agit de ce qu'on appelle le diffusion dirigée ou ciblée (direted/targeted diffusion) où la diffusion est envoyée d'un réseau à un autre réseau sous forme de paquet de datagramme IPv_4 monodiffusion. Lorsque le paquet de diffusion dirigée envoyé en monodiffusion depuis un réseau atteint le réseau de destination, il sera diffusé sur le réseau de destination.

1.6 Adresses IPv_4 privées et publiques

L'adressage privé peut être utilisé librement par n'importe quel administrateur ou utilisateur au sein de son réseau local. Au contraire, l'adressage public est soumis à des restrictions de déclaration et d'enregistrement de l'adresse IP auprès d'un organisme spécialisé, en l'occurrence l'IANA [12]. C'est ce que effectuent les FAI [13] en acquérant une plage d'adresses IPv_4 pour leurs abonnés.

11. Local Area Network
12. Internet Assigned Numbers Authority
13. Fournisseur d'accès Internet, en Anglais ISP (Internet Service Provider) : opérateur qui commercialise des accès à Internet.

1.6.1 Adresses IPv_4 privées

Les adresses IPv_4 privées (RFC[14]1918) sont non routables vers Internet. Le Tableau 9.1 illustre les différentes plages d'adresses IPv_4 privées.

Tableau 1.2 – Plages d'adresses IPv_4 privées.

Classe	Plage d'adresses privées
A	de 10.0.0.0 à 10.255.255.255 (1 réseau de la classe A)
B	de 172.16.0.0 à 172.31.255.255 (16 réseaux de la classe B)
C	de 192.168.1.0 à 192.168.255.255 (255 réseaux de la classe C)

1.6.2 Adresses IPv_4 publiques

Contrairement aux adresses IPv_4 privées, les adresses IPv_4 publiques ne sont pas utilisées dans un réseau local mais uniquement sur internet. Le Tableau 9.2 illustre les différentes plages d'adresses IPv_4 publiques. Elles représentent toutes les adresses IP des classes A, B et C qui ne font pas partie des plages d'adresses privées de ces classes (voir Tableau 9.1) ou des deux exceptions de la classe A que nous avons mentionnées dans le Section 1.5. Il est à noter qu'une adresse IPv_4 publique est une adresse qui est unique au niveau mondial.

Tableau 1.3 – Plages d'adresses IPv_4 publiques.

Classe	Plage d'adresses publique
A	1.0.0.0 - 9.255.255.255 11.0.0.0 - 126-255.255.255
B	128.0.0.0 - 169.253.255.255 169.255.0.0 à 172.15.255.255 172.32.0.0 - 191-255.255.255
C	192.0.0.0- 192.167.255.255 à 192.169.0.0 - 223-255.255.255

1.7 Masque sous-réseau

Le masque sous-réseau est une suite de quatre octets dont les bits correspondant à la partie Net-Id sont à 1, et ceux de la partie Host-Id sont à 0. En faisant un "*et logique*" bit à bit entre une adresse IPv_4 et le masque associé, on obtient l'adresse du réseau auquel elle appartient. La paire

14. Request For Comments

(adresse IPv_4, masque) peut être représentée en notation CIDR par l'adresse IPv_4 suivie d'une barre oblique "/" et le nombre de bits à 1 dans la notation binaire du masque de sous-réseau.

— Le masque sous réseau associé à l'adresse 192.168.1.1 est 255.255.255.0 puisqu'il s'agit d'une adresse IPv_4 de la classe C.

— En notation CIDR, on indiquera la paire (adresse IPv_4, masque) comme suit : 192.168.1.1/24.

— Un "*et logique*" entre 192.168.1.1 et 255.255.255.0 donne l'adresse réseau 192.168.1.0 auquel appartient l'adresse IPv_4.

Nous tenons enfin à noter que :

— Les termes masque réseau et masque sous réseau font référence à la même chose, c'est-à-dire à une série de bits utilisée pour séparer la partie réseau de la partie hôte dans une adresse IPv_4.

— Le découpage par classe permet de donner un seul masque pour tous les sous-réseaux, ce qui signifie que tous les sous-réseaux vont avoir le même nombre de machines ce qui n'est pas toujours optimal et entraîne un gaspillage au niveau des adresses machines et au niveau du nombre de sous-réseaux.

1.8 Adressage sans classes (CIDR)

CIDR est l'acronyme de Classless Inter Domain Routing (en Français, Routage Inter-domaines sans classes, RFC**1518** et **1519**). Il a été proposé à partir de 1994. Il s'agit d'une approche sans classes où on ne considère plus une adresse comme appartenant implicitement à l'une des trois classes A, B ou C. En CIDR, on associe explicitement à toute adresse IPv_4 un masque qui définit le préfixe qui caractérise le réseau auquel correspond cette adresse. Les adresses de réseaux sont donc toujours utilisées avec leur préfixe qui peut être de taille arbitraire (par exemple /10, /17, /21).

Le *CIDR* est actuellement le système de gestion et d'allocation d'adresses IPv_4 le plus répandu et optimisé. Il a été conçu pour remplacer l'adressage par classes. Son objectif est :

— Économiser les adresses IPv_4 : En effet, avec la croissance rapide d'internet, les adresses IPv_4 publiques commenaçient à manquer. En effet, une pénurie d'adresses IPv_4 s'est déclarée dès les années 1990.

— Faciliter le routage : L'utilisation de protocoles **classfull** (c.-à-d., les protocoles utilisant l'adressage par classes) et l'expansion vertigineuse de l'Internet ont entraîné une augmentation similaire de la taille des tables de routage sur les routeurs des FAI. Dans le but de réduire la taille des tables de routage ainsi que le trafic réseau, le principe du CIDR a été mis en place.

Pour résumer, nous dirons que :

— Le CIDR permet de diviser les réseaux en sous-réseaux de toute taille, ce qui permet une meilleure utilisation de l'espace d'adressage et réduit le gaspillage.

— Le CIDR, également connu sous le nom de supernetting, simplifie le routage en agrégeant les adresses IPv_4 en blocs plus grands, ce qui réduit la taille des tables de routage et améliore l'efficacité du routage sur Internet.

— les adresses IPv_4 utilisant l'adressage CIDR sont appelées *classless* addresses.

— Les adresses IPv_4 *classfull*, désignent les adresses qui utilisent l'adressage par classes.

1.9 Subnetting

Nous allons traiter le subnetting à travers l'exemple suivant :

Soit le réseau d'adresse 134.214.0.0/16. On veut le découper en 4 sous-réseaux. Pour chaque sous-réseau, on veut obtenir le masque, l'adresse du sous-réseau et l'adresse de broadcast.

Rappelons tout d'abord ce qui suit :

— Pour obtenir l'adresse réseau, tous les bits du Host-Id sont positionnés à 0

— Pour obtenir l'adresse de broadcast, tous les bits du Host-Id sont positionnés à 1.

Solution :

1. On veut découper le réseau initial en 4 sous réseaux. Sachant que $4 \leq 2^2$, le masque de chaque sous-réseau est obtenu en ajoutant 2 bits à 1 au masque initial. Notons que les deux bits en question ont été emprûntés à la partie Host-Id. Le masque initial étant 255.255.0.0 (16 bits à 1 suivis de 16 bits à 0). Le nouveau masque est : 255.255.192.0 ($16 + 2 = 18$ bits à 1 (/18 en notation CIDR) suivis de 14 bits à 0). La partie Host-Id, initialement constituée de 16 bits (l'adresse réseau considérée étant une adresse de classe B), est maintenant constituée de 14 bits après lui avoir emprûnté 2 bits pour créer 4 sous-réseaux du réseau initial, à savoir 134.214.0.0/16.
2. On va maintenant déterminer l'adresse de chaque sous-réseau. Pour cela, nous allons positionner tous les bits du Host-Id à 0 :
   — 134.214.(00000000).0, soit @(sous-réseau 1) est 134.214.0.0/18
   — 134.214.(01000000).0, soit @(sous-réseau 2) est 134.214.64.0/18
   — 134.214.(10000000).0, soit @(sous-réseau 3) est 134.214.128.0/18
   — 134.214.(11000000).0, soit @(sous-réseau 4) est 134.214.192.0/18
3. Pour l'obtention des adresses de broadcast de chaque sous-réseau, on positionne tous les bits du Host-Id à 1 :
   — 134.214.(00111111).255, soit @(broadcast du sous-réseau 1) est 134.214.63.255/18
   — 134.214.(01111111).255, soit @(broadcast du sous-réseau 2) est 134.214.127.255/18
   — 134.214.(10111111).255, soit @(broadcast du sous-réseau 3) est 134.214.191.255/18
   — 134.214.(11111111).255, soit @(broadcast du sous-réseau 4) est 134.214.255.255/18

Exercice :

Un réseau de classe B dispose du masque de sous-réseau 255.255.248.0.

1. Combien de bits ont-ils été pris de la partie Host-Id pour créer des sous-réseaux ?
2. En déduire le nombre de sous-réseaux possibles.
3. Combien de bits restent-ils à Host-Id ?
4. En déduire le nombre de machines possibles pour chacun des sous-réseaux.

1.10 Supernetting

Le supernetting (appelé aussi route summarization, route agrégation ou résumé de routes) permet d'agréger les adresses IPv_4 en une seule adresse, ce qui permet de réduire la taille des tables de routage et le trafic réseau (moins d'annonces de routes par les protocoles de routage dynamiques, ce qui permet d'economiser la bande passante).

Il est à noter qu'avant de procéder à l'agrégation des routes, on doit s'assurer que les règles suivantes sont vérifiées :

1. Tous les réseaux doivent être contigus.
2. La taille de bloc de chaque réseau doit être la même (c.-à-d., tous les réseaux doivent avoir la même taille) et la taille totale du supernet doit être sous la forme 2^n.
3. Le premier identifiant de réseau (Net-Id) doit être exactement divisible par la taille totale du supernet. Notez que le premier identifiant de réseau signifie le plus petit identifiant.

Exemple : Considérons les 4 routes suivantes à agréger :

— 200.1.0.0 / 24 (255.255.255.0)

— 200.1.1.0 / 24 (255.255.255.0)

— 200.1.2.0 / 24 (255.255.255.0)

— 200.1.3.0 / 24 (255.255.255.0)

On va tout d'abord vérifier si les 3 règles mentionnées ci-dessus sont vérifiées :

1. Les 4 réseaux, sont-ils contigus ? → Oui : (200.1.0.0, 200.1.1.0, 200.1.2.0, 200.1.3.0). En effet, la plage du premier réseau est de 200.1.0.0 à 200.1.0.255. Si vous ajoutez 1 à la dernière adresse IP du premier réseau, c.-à-d., 200.1.0.255 + 0.0.0.1, vous obtiendrez le prochain identifiant de réseau qui est 200.1.1.0. En faisant la même opération avec les 3 autres réseaux, vous pouvez vérifier que tous les réseaux sont contigus.
2. Les 4 réseaux, sont-ils de même taille ? → Oui : La taille de bloc de chaque réseau est de 2^8 (le Host-Id=8bits). La taille totale du supernet est-elle sous la forme 2^n ? → Oui : Nous avons 4 blocs, c.-à-d., $4 * 2^8 = 2^2 * 2^8 = 2^{10}$.
3. Le premier identifiant de réseau, est-il exactement divisible par la taille totale du supernet ? → Oui :
   — La taille totale du supernet est $4 * 2^8 = 2^2 * 2^8 = 2^{10}$ IPv_4 adresses.
   — Le premier identifiant de réseau, en l'occurrence 200.1.0.0/24 est bien divisible par 2^{10} car ses 10 derniers bits, qui representent le reste de la division de ce premier

identifiant de réseau par 2^{10}, sont égals à 00.00000000. Par conséquent, à chaque fois qu'on veut prouver que la première adresse IPv_4 est exactement divisible par la taille du supernet, nous pouvons vérifier les n derniers bits s'ils sont à 0 ou non.

Pour déterminer le masque du supernet, on regarde les bits en commun :

— 200.1.0.0 / 24 → 11001000.00000001.00000000.00000000
— 200.1.1.0 / 24 → 11001000.00000001.00000001.00000000
— 200.1.2.0 / 24 → 11001000.00000001.00000010.00000000
— 200.1.3.0 / 24 → 11001000.00000001.00000011.00000000

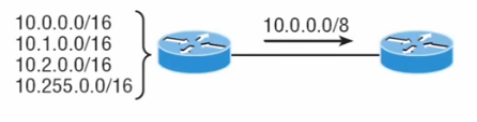
En comptant les bits en commun (bits en bleu), on peut affirmer que le masque du supernet est /22.

En ce qui concerne le Net-Id, ce dernier est obtenu en mettant à 0 tout les bits non communs, puis en lisant les adresses résultantes. En général, nous choisirions l'adresse la plus basse ou la première dans la séquence

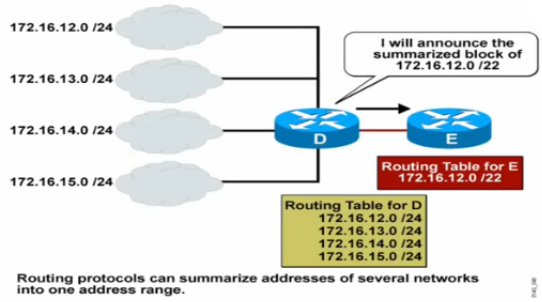
— 200.1.0.0 / 24 → 11001000.00000001.00000000.00000000
— 200.1.1.0 / 24 → 11001000.00000001.00000000.00000000
— 200.1.2.0 / 24 → 11001000.00000001.00000000.00000000
— 200.1.3.0 / 24 → 11001000.00000001.00000000.00000000

Le Net-Id est donc 200.1.0.0 et la route agrégée est par conséquent : 200.1.0.0/22

Le Tableau 1.4 illustre les étapes de calcul de la route agrégée tandis que La Figure 1.4 montre deux exemples d'utilisation de la route agrégée.



(a) Résumé de routes :Exemple 1



(b) Résumé de routes :Exemple 2

FIGURE 1.4 – Exemples d'utilisation de résumé de routes [3]

Pour trouver les adresses agrégées dans une route donnée, vous pouvez utiliser la méthode suivante :

TABLEAU 1.4 – Comment calculer le résumé de routes.

Adresse IPv_4	Notation binaire
172.20.0.0/16	10101100 . 00010100 . 00000000 . 00000000
172.21.0.0/16	10101100 . 00010101 . 00000000 . 00000000
172.22.0.0/16	10101100 . 00010110 . 00000000 . 00000000
172.23.0.0/16	10101100 . 00010111 . 00000000 . 00000000
	Les réseaux sont contigus, de tailles égales, la taile du supernet est sous la forme 2^n et le premier (Net-Id) est divisible par la taille totale du supernet
	Le nombre de bits correspondants (bits communs) est égal à 14
	Copier les bits correspondants et mettre les bits restants à zero pour determiner l'adresse réseau résumée (Typiquement, vous choisiriez l'adresse la plus basse ou la première dans la séquence.
172.20.0.0/14	10101100 . 00010100 . 00000000 . 00000000

— Déterminez la plage d'adresses IP qui tombent dans la route agrégée. Calculez les adresses réseau et de diffusion pour la route donnée.

— Identifiez tous les sous-réseaux spécifiques englobés par la route agrégée.

Par exemple, si vous avez la route agrégée 172.16.12.0/22, vous pouvez trouver les sous-réseaux spécifiques englobés par cette route en calculant la plage d'adresses IP, l'adresse réseau et de diffusion, ainsi que les sous-réseaux spécifiques dans cette plage. En utilisant la route 172.16.12.0/22 comme exemple, la plage d'adresses IP dans cette route va de 172.16.12.0 à 172.16.15.255. L'adresse réseau est 172.16.12.0, et l'adresse de diffusion est 172.16.15.255. Les sous-réseaux spécifiques dans cette route agrégée sont 172.16.12.0/24, 172.16.13.0/24, 172.16.14.0/24 et 172.16.15.0/24.

Avant de clore cette Section sur le supernetting, nous tenons à rappeler les deux formules suivantes :

— Nbre [15] de réseaux du supernet = $2^{(\text{Nbre de 1 dans le masque par défaut - Nbre de 1 dans le masque du supernet})}$

— Nbre d'adresses IPv_4 dans le supernet = $2^{(\text{Nbre de 0 dans le masque du supernet})}$

Aussi, nous tenons à rappeler ce qui suit :

— Une des conséquences des 3 règles que nous avons énumérées ci-dessus concernant la réalisation d'une agrégation de routes, est que nous ne pouvons combiner que 2, 4, 8, 16 réseaux etc., c.-à-d., un nombre de réseaux en puissance de 2. Nous ne pouvons pas combiner 3, 10 réseaux etc.

15. Nombre

— Vous pouvez obtenir l'identifiant de réseau de la route agrégée en prenant les bits communs des adresses IP données et en mettant les bits restants à 0. Typiquement, vous choisiriez l'adresse la plus basse ou la première dans la séquence.

— Vous pouvez déterminer l'identifiant de réseau de la route agrégée en prenant le plus petit identifiant parmi les adresses IP données.

— Vous pouvez obtenir l'identifiant de réseau de la route agrégée en effectuant une opération logique "ET" bit à bit entre les adresses IP données.

Rappelez vous aussi que le nombre de 1 dans le masque de supernet est toujours inférieur au nombre de 1 dans le masque initial.

Exercice :

Trouver la route agrégée des réseaux suivants :

— 200.1.2.0/25

— 200.1.2.128/26

— 200.1.2.192/26

1.11 Variable Length Subnet Mask (VLSM)

Nous tenons tout d'abord à rappeler que pour pallier au manque d'adresses IPv_4, plusieurs solutions ont été envisagées. Le VLSM (en Français, Masques sous réseaux à longueur variable) en fait partie :

— Subnetting en 85

— Classless Inter Domain Routing

— Variable Lenght Subnet Mask

— Network address Translation (NAT)

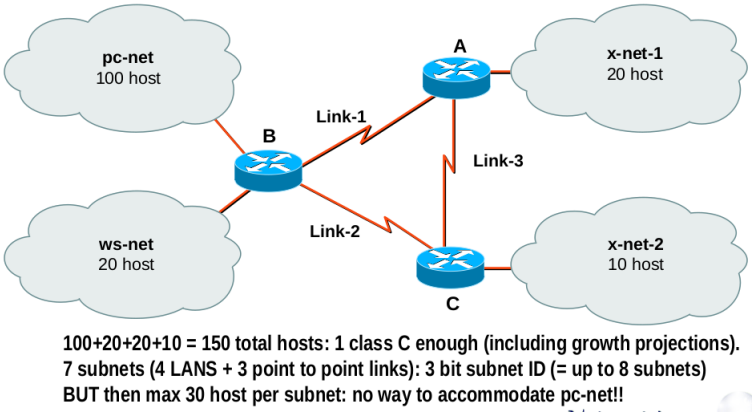
— Solution ultime : IPv6 (128 bits) :

On parle de VLSM quand un réseau est divisé en sous-réseaux de différentes tailles, ce qui permet une meilleure utilisation des adresses disponibles. Notons que certains protocoles dynamiques, à l'image de BGP [16], OSPF [17], IS-IS [18], EIGRP [19] et RIPv2 [20] supportent le VLSM car ils indiquent toujours un masque réseau associé à une route annoncée. Les protocoles tels que RIPv1, IGRP [21] et EGP [22] ne supportent pas le VLSM.

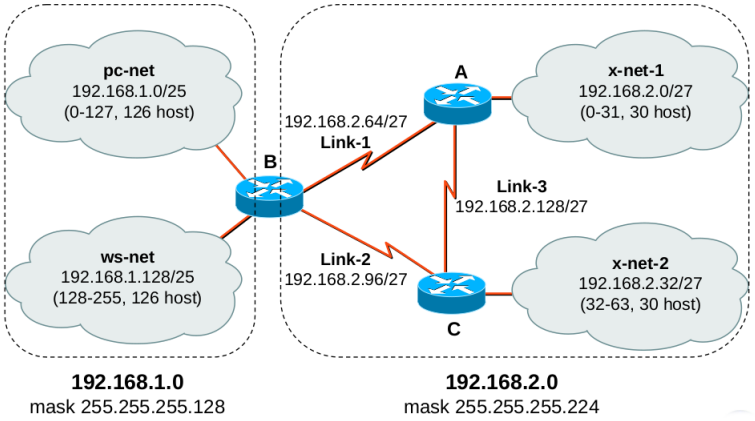
La Figure 1.5 illustre l'intérêt de résoudre un problème typique en utilisant la technique du VLSM.

La Figure 1.6 illustre le découpage approprié au problème présenté dans la Figure 1.5a.

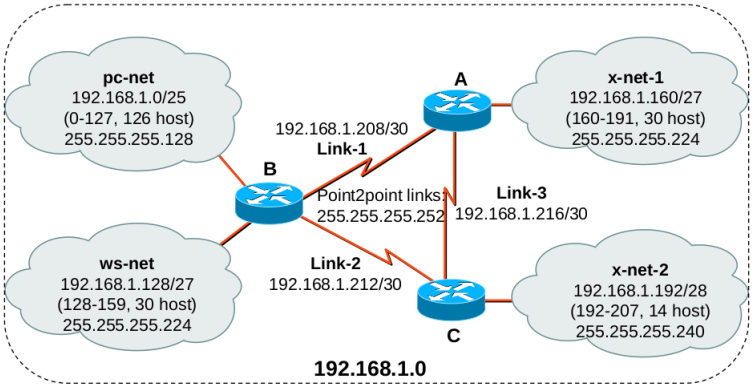
16. Border Gateway Protocol
17. Open Shortest Path First
18. Intermediate system to Intermediate System
19. Enhanced Interior Gateway Routing Protocol
20. Routing Information Protocol Version 2
21. Interior Gateway Routing Protocol
22. Exterior Gateway Protocol



(a) Problème typique



(b) Solution sans VLSM du problème : 2 adresses de classes C sont nécessaires



(c) Solution avec VLSM du problème : 1 adresse de classes C est suffisante

FIGURE 1.5 – Utilisation du VLSM pour la résolution d'un problème typique [4].

Exercice :

Soit l'adresse réseau IPv_4 : 195.167.46.0/24. On veut découper ce réseau en trois sous-réseaux comme illustré dans la Figure 1.7.

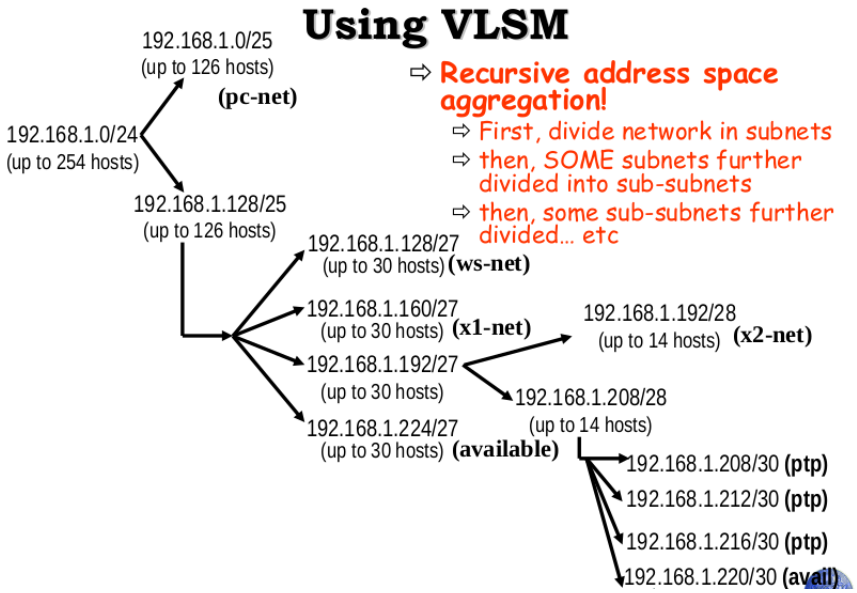


FIGURE 1.6 – Découpage proposé pour résoudre le problème décrit dans la Figure 1.5a [4].

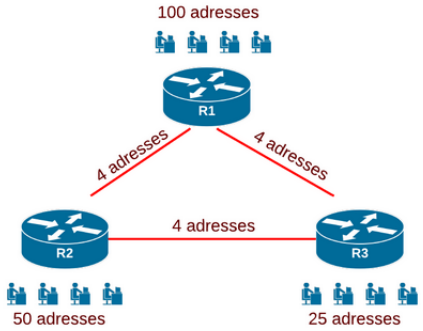


FIGURE 1.7 – Réseau à découper en utilisant la technique du VLSM.

— Sous-Réseau 1 (S/R 1) : 100 hôtes
— Sous-Réseau 2 (S/R 2) : 50 hôtes
— Sous-Réseau 3 (S/R 3) : 25 hôtes.

On a donc un bloc de 256 adresses (/24). On propose ici de le découper de la manière suivante :

— un bloc /25 de 128 adresses pour le LAN de R1 ($2^7 - 2 >= 100\,;\, 126 + 2$)
— un bloc /26 de 64 adresses pour le LAN de R2, ($2^6 - 2 >= 50\,;\, 62 + 2$)
— un bloc /27 de 32 adresses pour le LAN de R3. , ($2^5 - 2 >= 50\,;\, 30 + 2$)
— Dans le reste, on prendra trois blocs /30 de 4 adresses pour adresser les connexions point-à-point entre les routeurs.

Notons que dans ce genre de problème, on repère le sous-réseau qui a le plus grand nombre d'hôtes et on commence avec.

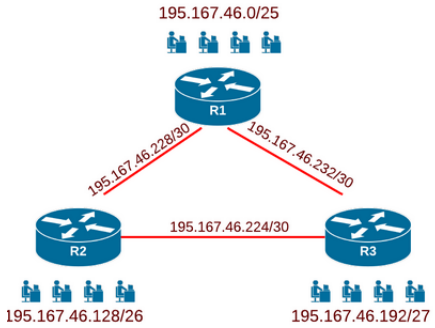


FIGURE 1.8 – Aperçu du découpage du réseau de la Figure 1.7, en utilisant la technique du VLSM.

Pour clore cette Section, nous tenons à rappeler ce qui suit :

— Quand vous avez trop d'adresses IP par sous-réseau, vous devriez empêcher l'assignation des adresses inutilisées, pour empêcher qu'un intrus puisse s'octroyer une adresse IP libre.

— La RFC950 recommande de ne pas utiliser les sous-réseaux dont les bits correspondant à la nouvelle partie du masque sont tous à 0 ou tous à 1, et ce pour éviter les erreurs d'interprétation par les protocoles de routage dits classful tel que RIPv1. En effet, ce type de protocole ne véhicule aucune information sur le masque. Par exemple, l'adresse du sous-réseau 192.168.1.0 où les 3 bits emprûntés de la partie hôte sont à zéro, peut être considérée comme l'adresse réseau de 2 réseaux différents : le réseau 192.168.1.0/24 et 192.168.1.0/27. De la même façon, l'adresse de diffusion 192.168.1.255 où les 3 bits emprûntés de la partie hôte sont à 1, est la même pour 2 réseaux différents : 192.168.1.0/255 ou 192.168.1.224/27.

  Depuis la publication du document RFC950, en 1985, les protocoles de routage qui servent à échanger les tables d'adresses de réseaux connectés entre routeurs ont évolué. Tous les protocoles contemporains sont conformes aux règles de routage inter-domaine sans classe (CIDR). Les protocoles tels que RIPv2, OSPF et BGP intègrent le traitement des masques de sous-réseaux. Ils peuvent même regrouper ces sous-réseaux pour optimiser le nombre des entrées des tables de routage. Pour appuyer cet argument, le document RFC1878 de 1995 spécifie clairement que la pratique d'exclusion des sous-réseaux all-zeros et all-ones est obsolète.

1.12 Conclusion

Ce Chapitre nous a permis, d'une part, de mettre en évidence le fait que l'adressage IPv_4 constitue le fondement essentiel de la connectivité et de la communication au sein des réseaux informatiques. D'autre part, nous avons mis l'accent sur un des défis majeurs auxquels est confronté le système IPv_4, à savoir l'épuisement progressif des adresses IPv_4 en raison de la croissance exponentielle des dispositifs connectés. Pour faire face à cette problématique, plusieurs solutions ont été envisagées, telles que l'adressage sans classes, le VLSM et la transition vers IPv_6.

Tableau 1.5 – Détail du découpage du réseau de la Figure 1.7 en utilisant la technique du VLSM.

S/R	Nbre hôtes	@ S/R	Plage	@ diffusion
S/R 1	100	195.167.46.0/25	195.167.46.1/25 - 195.167.46.126/25	195.167.46.127/25
S/R 2	50	195.167.46.128/26	195.167.46.129/26 - 195.167.46.190/26	195.167.46.191/26
S/R 3	25	195.167.46.192/27	195.167.46.193/27 - 195.167.46.222/27	195.167.46.223/27
R2-R3	4	195.167.46.224/30	195.167.46.225/30 - 195.167.46.226/30	195.167.46.227/30
R1-R2	4	195.167.46.228/30	195.167.46.229/30 - 195.167.46.230/30	195.167.46.231/30
R1-R3	4	195.167.46.232/30	195.167.46.233/30 - 195.167.46.234/30	195.167.46.235/30

Chapitre 2

Routage

2.1 Introduction

Le routage dans les réseaux consiste à trouver les chemins de moindre coût pour acheminer les données vers leurs destinations. Les routeurs sont comme des guides qui aident ces données à atteindre leurs destinations de la manière la plus efficace possible. Dans ce Chapitre, nous allons découvrir comment tout cela fonctionne, et comment les données sont "guidées" à travers les réseaux informatiques.

2.2 Routage

La fonction principale d'un routeur est de déplacer un paquet IP d'un réseau à un autre réseau, et pour cela le routeur a besoin de ce qui suit :

— L'adresse de destination

— Les itinéraires possibles

— Le meilleur itinéraire

FIGURE 2.1 – Routage [5]

Il est à noter que [6] :

— Un réseau de routeurs interconnectés et de systèmes connexes gérés par une administration de réseau commune est appelé système autonome (en Anglais, Autonomous System (AS)) ou domaine de routage (voir Figure 2.2).

— L'internet est composé de milliers de systèmes autonomes répartis sur toute la planète.

La notion d'AS est abordée avec plus de détail dans la Section 2.9.

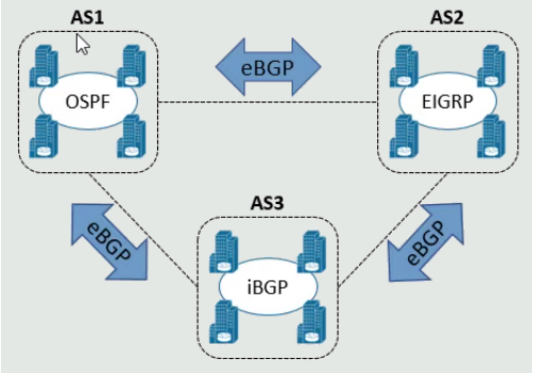


FIGURE 2.2 – Systèmes Autonomes (iBGP : Interior Gateway BGP, eBGP : Exterior Gateway BGP) [6]

2.3 Types de routage

Il existe deux types de routage :

— Routage statiques

— Routage dynamique

2.3.1 Routage statique

Dans ce type de routage, l'administrateur réseau configure **manuellement** les informations sur le(s) réseau(x) distant(s). Il est à noter que si un réseau est directement [1] connecté à un routeur, ce dernier le reconnaît automatiquement sans que l'administrateur n'intervienne.

2.3.1.1 Avantages

— Economie de la bande passante : Etant donné qu'aucune information concernant les routes n'est échangée par les routeurs pour qu'ils soient à jour, la bande passante n'est pas encombrée avec des messages d'échange d'information et de routage.

— Pas de surcharge sur le processeur du routeur : La charge est plutôt sur l'administrateur.

— Connaissance du chemin à l'avance : L'administrateur ayant configuré l'ensemble de la topologie saura exactement par où transitent les paquets pour aller d'un réseau à un autre. Ceci pourrait faciliter la compréhension des erreurs eventuelles lors de la transmission des paquets.

1. Un réseau directement connecté est un réseau directement attaché à l'une des interfaces réseau du routeur.

— Sécurité : Contrairement aux protocoles de routage dynamiques, le routage statique ne diffuse pas d'information concernant les routes sur le réseau puisque les routes sont manuellement configurées par l'administrateur.

2.3.1.2 Inconvénients

— Si un réseau est ajouté à la topologie initiale, l'administrateur doit lui ajouter une route sur tous les routeurs.

— Non réalisable dans les grands réseaux : Imaginons, par exemple, un administrateur qui va configurer un certain nombre de routeurs n (par exemple, $n \geq 8$) un à un au sein du réseau afin d'y indiquer les routes distantes.

2.3.1.3 Route statique par défaut

Un réseau candidat par défaut (en Anglais, candidate default), aussi appelé route par défaut, est une entrée de table de routage qui dirige les paquets vers un saut suivant défini, lorsqu'il n'y a pas d'entrée explicite pour le réseau de destination. Ce type de route est utilisé par exemple pour rediriger les paquets d'un réseau LAN vers Internet (voir Figure 2.3).

Tout paquet qu'un routeur reçoit n'ayant pas d'entrée explicite ou implicite (réseau candidat par défaut ou route par défaut) dans la table de routage est détruit et un message ICMP [2] "Network Unreachable" est alors envoyé par le routeur à la station source [32].

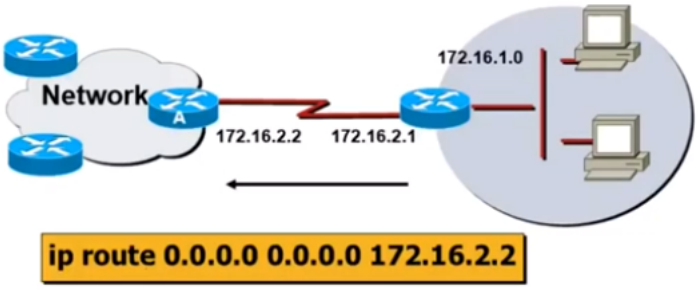


FIGURE 2.3 – Configuration de la route par défaut [5].

2.3.1.4 Configuration d'une route statique sous PacketTracer

1. Ajout d'une route statique sous PacketTracer de CISCO :
   — Router(config)# *ip route* network-address subnet-mask exit-interface
   Par exemple, pour le cas du routeur R1 de la Figure 2.4, on saisit la commande suivante :
   — Router(config)# *ip route* 192.168.3.0 255.255.255.0 192.168.2.1
2. Ajout d'une route statique par défaut sous PacketTracer de CISCO :

2. Internet Control Message Protocol

— Router(config)# *ip route* 0.0.0.0 0.0.0.0 exit-interface

Par exemple, pour le cas du routeur R1 de la Figure 2.3, on saisit la commande suivante :

— Router(config)# *ip route* 0.0.0.0 0.0.0.0 172.16.2.2

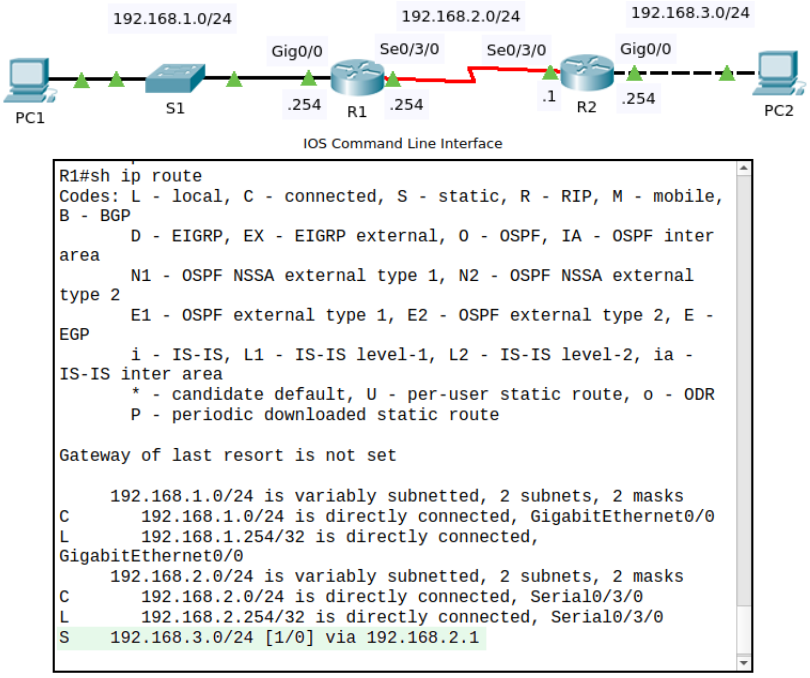


FIGURE 2.4 – Configuration des routes statiques.

2.3.2 Routage dynamique

Dans ce type de routage, les informations sont apprises à partir d'autres routeurs et les protocoles de routage ajustent automatiquement les routes. Parmi les protocoles de routage dynamique courants, on citera :

— RIP (v1 & v2) [3]

— EIGRP [4]

— OSPF [5]

— IS-IS [6]

— BGP [7]

3. Routing Information Protocol version 1 & version 2
4. Enhanced Interior Gateway Routing Protocol
5. Open Shortest Path First
6. Intermediate System-to-Intermediate System
7. Border Gateway Protocol

2.3.2.1 Avantages

— Partage de manière dynamique des informations sur les réseaux distants entre routeurs.

— Découverte des réseaux distants et mise à jour des tables de routage dès qu'une topologie est modifiée.

— Recherche du meilleur chemin (et du meilleur chemin de secours) vers chaque destination.

2.3.2.2 Inconvénients

— Consommation du temps processeur et de la bande passante du lien réseau

— Problèmes de sécurité (voir Section 2.3.1.1, aspect sécurité d'un protocole de routage statique)

2.3.2.3 Classification des protocoles de routage dynamique

La Figure 2.5 illustre une classification des protocoles de routage dynamique selon leur type, à savoir IGP [8] ou EGP [9], et l'algorithme utilisé pour la sélection des chemins, c.-à-d., vecteur de distance, état de liens, hybride ou vecteur de chemin.

La Figure 2.6 étend la classification au mode d'adressage utilisé par ces protocoles (classful ou classless).

Il est à noter que :

— Un protocole IGP est un type de protocole de routage utilisé pour échanger des informations sur les tables de routage entre les routeurs au sein d'un système autonome (AS) (voir Figure 2.2).

— Un protocole EGP est utilisé pour échanger des informations de routage entre les systèmes autonomes (ASs) et s'appuie sur les protocoles IGPs pour résoudre les routes au sein d'un système autonome (AS) (voir Figure 2.2).

2.3.2.4 Algorithmes de selection des routes utilisés par les IGPs et EGPs

Les protocoles IGPs et EGPs utilisent différents algorithmes pour la sélection des routes tel qu'illustré dans le Tableau 2.1.

A- Algorithme à vecteur de distance [6]

— Les protocoles de routage à vecteur de distance tel que RIP, annoncent les routes sous forme de vecteurs de distance, où la distance est une métrique tel que le nombre de sauts, et le vecteur est l'adresse IP du routeur du prochain saut utilisé pour atteindre la destination, comme l'illustrent les Figures 2.7 et 2.8 .

— Lorsqu'un routeur reçoit des informations de routage d'un voisin, il les stocke dans une base de données de routage locale au fur et à mesure qu'il les reçoit. Ensuite, il utilise un algorithme de type vecteur de distance tel que l'algorithme de Bellman-Ford ou celui

8. Interior Gateway Protocol
9. Exterior Gateway Protocol

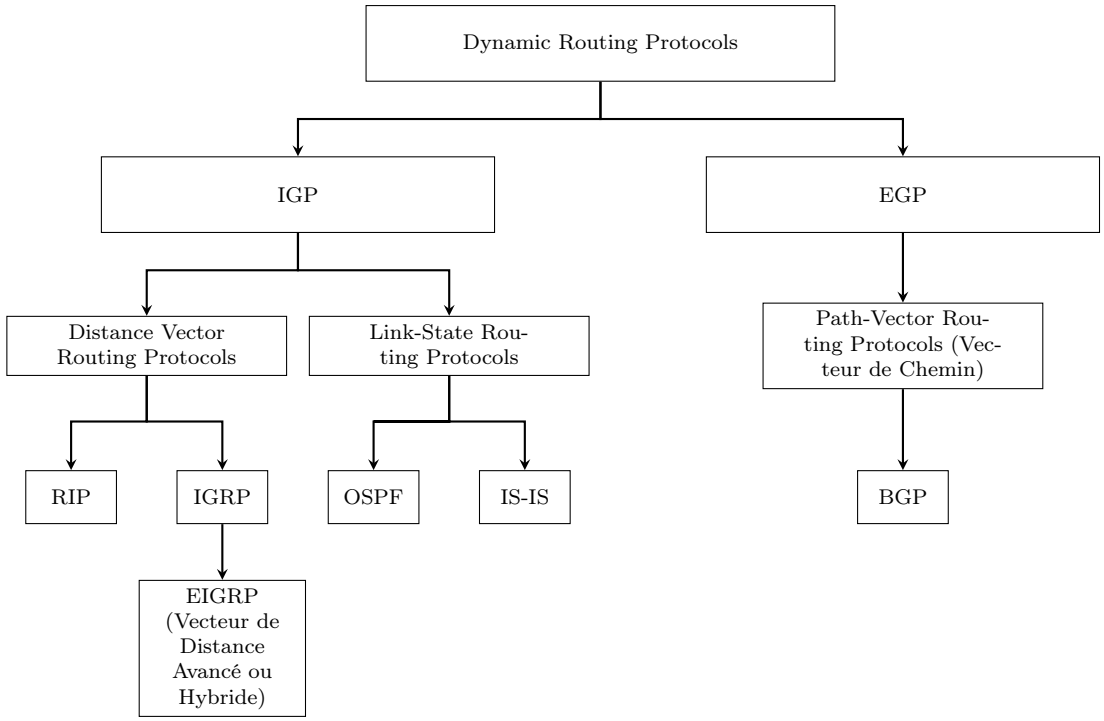


FIGURE 2.5 – Classification des protocoles de routage dynamique selon le type et l'algorithme utilisé.

de Ford-Fulkerson pour déterminer quels sont les meilleurs chemins sans boucle (loop-free paths) vers chaque destination atteignable.

— Lorsque les meilleurs chemins sont déterminés, ils sont installés dans la table de routage et sont annoncés à chaque routeur voisin.

— Les routeurs exécutant le protocole à vecteur de distance annoncent les informations de routage à leurs voisins "selon leur propre point de vue", c.-à-d., modifiées par rapport aux routes originales reçues (après avoir exécuté Bellman-Ford ou tout autre algorithme du même type). Par conséquent, un protocole à vecteur de distance ne dispose pas d'une carte complète de l'ensemble du réseau, mais sa base de données indique qu'un routeur voisin sait comment atteindre le réseau de destination et à quelle distance il se trouve de ce réseau.

— Les protocoles à vecteur de distance nécessitent moins de ressources CPU [10] et mémoire et peuvent par conséquent fonctionner sur des routeurs d'entrée de gamme.

— Les protocoles à vecteur de distance selectionnent les chemins sur la base du nombre de sauts uniquement. Ils ne tiennent pas compte d'autres métriques telle que par exemple la qualité des liens (bandwidth, delai, RSSI [11], etc.).

Remarques [7]

Dans un protocole de routage à vecteur de distance :

10. Central Processing Unit
11. Received Signal Strength Indicator

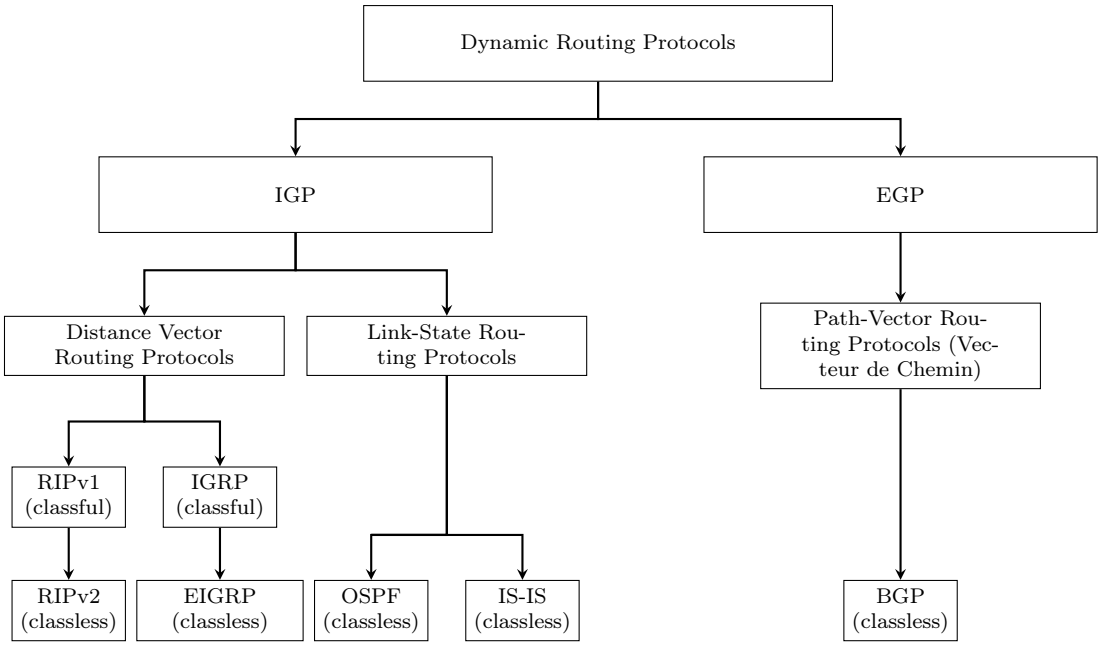


FIGURE 2.6 – Classification des protocoles de routage dynamique selon le type, l'algorithme utilisé et le mode d'adressage.

TABLEAU 2.1 – Protocoles de routage et algorithmes de selection de chemin.

Protocole de routage	Algorithme de routage
RIP v1 & v2	Vecteur de distance
OSPF / IS-IS	Etat de liens
EIGRP	Vecteur de distance avancé ou hybride
BGP	Vecteur de Chemin

— La route est définie comme une direction associée à une distance.

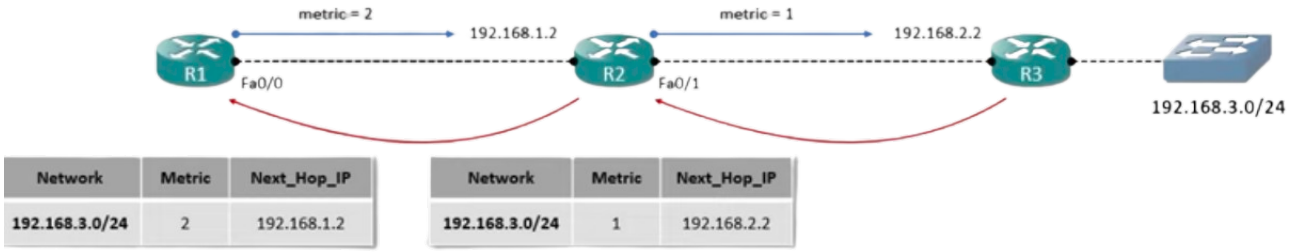
— Chaque routeur du réseau dépend de ses voisins pour apprendre les routes et contribue à l'apprentissage de ses voisins en leur annonçant les routes qu'il connaît. L'information de topologie circule de proche en proche sur le réseau, ce qui fait parfois appeler ce type de routage "routage par la rumeur".

B- Algorithme à vecteur de distance avancé ou hybride [6]

— DUAL [12] est un algorithme de type vecteur de distance amélioré qui est utilisé par EIGRP par exemple pour calculer le plus court chemin vers une destination.

— Parmi les améliorations introduites dans l'algorithme DUAL par rapport à d'autres algorithmes à vecteur de distance, on citera :

12. Diffusing Update ALgorithm



Network	Metric	Next_Hop_IP
192.168.3.0/24	2	192.168.1.2

Network	Metric	Next_Hop_IP
192.168.3.0/24	1	192.168.2.2

FIGURE 2.7 – Algorithme à vecteur de distance : Exemple 1 [6]

.

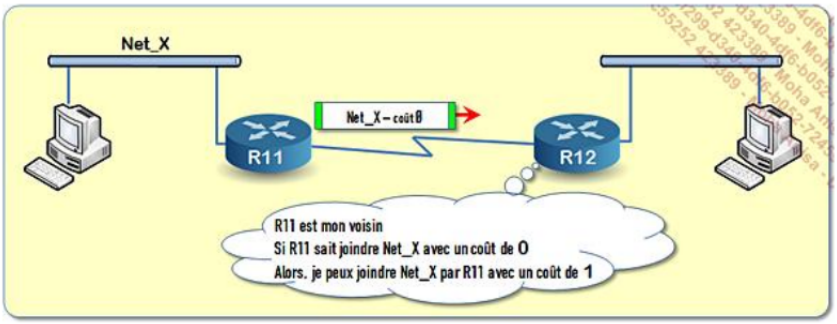


FIGURE 2.8 – Algorithme à vecteur de distance : Exemple 2 [7]

.

— Temps de convergence rapide pour les changements dans la topologie du réseau.

— L'envoi des mises à jour uniquement lorsqu'il y a un changement dans la topologie du réseau. Il n'y a pas d'envoi de mises à jour complètes de la table de routage de façon périodique, comme le font les protocoles à vecteur de distance tel que RIP par exemple qui envoie toute la table de routage toutes les 30 secondes (période par défaut).

— Utilisation des "hello" périodiquement pour la formation des relations de voisinage comme le font les protocoles à état de liaison tel que OSPF.

— Utilisation de la bande passante, le délai, la fiabilité, la charge et la taille du MTU [13] au lieu du nombre de sauts pour calculer le chemin. Par exemple, EIGRP se base sur la bande passante (bandwidth) et le délai (delay) (voir Figure 2.9).

— possibilité d'équilibrer la charge du trafic sur des chemins de coûts égaux ou inégaux.

C- Algorithme à état de liens [6]

— OSPF et IS-IS constituent deux exemples de ce type de protocoles, couramment utilisés dans les réseaux d'entreprises et des fournisseurs de services.

13. Maximum Transmission Unit

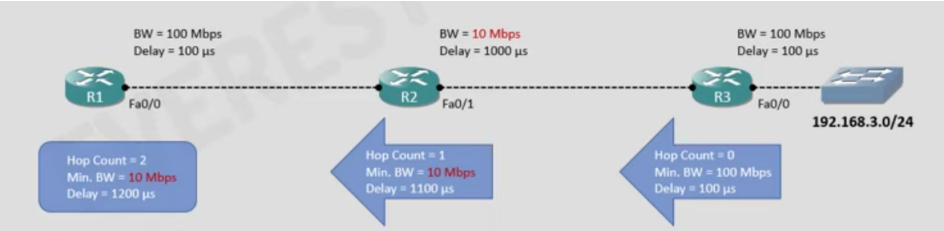


FIGURE 2.9 – EIGRP [6].

— Les annonces OSPF sont appelées LSAs [14] tandis que les messages d'annonce utilisés par IS-IS sont appelés LSPs [15].

— Lorsqu'un routeur reçoit une annonce d'un voisin, il stocke les informations dans une base de données locale appelée LSDB [16] et annonce les informations sur l'état des liens à chacun de ses routeurs voisins exactement telles qu'elles ont été reçues.

— Les informations sur l'état des liens sont essentiellement diffusées à travers le réseau, sans changement d'un routeur à l'autre, exactement comme le routeur d'origine les a annoncées. Cela permet à tous les routeurs du réseau de disposer d'une carte synchrone et identique de la topologie du réseau (Les routeurs R1, R2 et R3 de la Figure 2.10 connaissent chacun la topologie complète du réseau.)

— À l'aide de la carte complète du réseau, chaque routeur du réseau exécute ensuite l'algorithme SPF [17] de Dijkstra pour calculer les meilleurs plus courts chemins sans boucle. L'algorithme link-state remplit ensuite la table de routage avec ces informations.

— Puisque les routeurs sur lesquels s'exécute un protocole de routage à état de liens construisent chacun une carte complète de la topologie du réseau, ils doivent donc disposer de plus de ressources, en termes de CPU et de mémoire, que les routeurs qui exécutent un protocole à vecteur de distance.

— Les protocoles à état de liens sont moins sujets aux boucles de routage et prennent de meilleures décisions en matière de chemins.

D- Algorithme à vecteur de chemin (path vector) [8]

— BGP externe qui est le protocole utilisé pour le routage externe (routage inter-AS), utilise ce type d'algorithme.

— Un routeur sur lequel est installé BGP externe n'annonce pas une métrique à ses voisins, comme dans le cas des protocoles à vecteur de distance, mais le chemin complet qu'a suivi l'annonce avant d'atteindre le récepteur. Par exemple, dans la topologie de la Figure 2.12, le routeur S annonce S, le routeur A annonce A-S, le routeur B annonce B-A-S. Le next-hop est défini comme étant le nœud en tête du chemin.

14. Link-state Advertisements
15. Link-State packets
16. Link-State Data Base
17. Shortest Path First

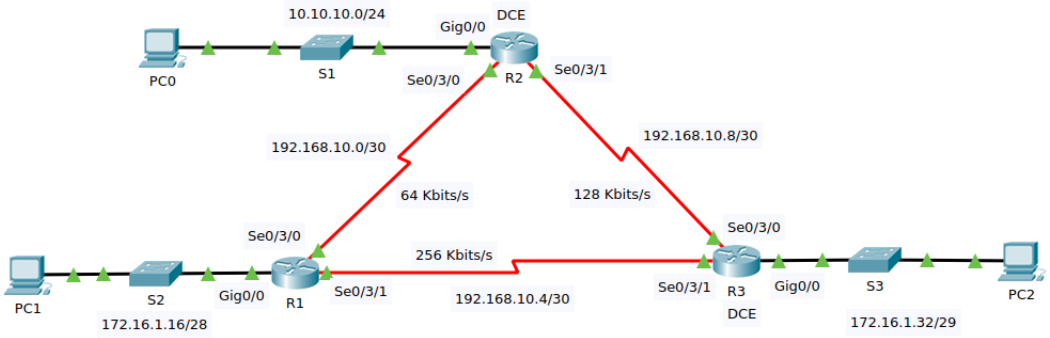


FIGURE 2.10 – OSPF.

IOS Command Line Interface

```
Router#
Router#
Router#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 1 subnets
O       10.10.10.0/24 [110/1172] via 192.168.10.6, 01:07:44, Serial0/3/1
     172.16.0.0/16 is variably subnetted, 3 subnets, 3 masks
C       172.16.1.16/28 is directly connected, GigabitEthernet0/0
L       172.16.1.17/32 is directly connected, GigabitEthernet0/0
O       172.16.1.32/29 [110/391] via 192.168.10.6, 01:21:31, Serial0/3/1
     192.168.10.0/24 is variably subnetted, 5 subnets, 2 masks
C       192.168.10.0/30 is directly connected, Serial0/3/0
L       192.168.10.1/32 is directly connected, Serial0/3/0
C       192.168.10.4/30 is directly connected, Serial0/3/1
L       192.168.10.5/32 is directly connected, Serial0/3/1
O       192.168.10.8/30 [110/1171] via 192.168.10.6, 01:07:44, Serial0/3/1

Router#
Router#
```

Ctrl+F6 to exit CLI focus

Copy Paste

FIGURE 2.11 – Table de routage du routeur R1 correspondante à la topologie de la Figure 2.10.

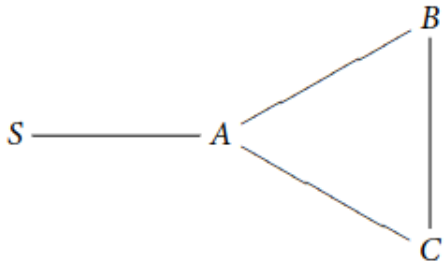


FIGURE 2.12 – Topologie réseau [8].

— Pour éviter les boucles, chaque routeur, avant d'accepter une annonce, vérifie s'il est présent dans le chemin annoncé ; si c'est le cas, il ignore l'annonce. Une annonce est préférable à une autre si le chemin qu'elle porte est plus court. C'est la longueur du chemin qui sert de métrique.

2.4 Table de routage

La table de routage est l'élément central d'un routeur. Elle est utilisée par la fonction de routage pour déterminer le meilleur chemin pour chaque destination connue du routeur. Elle inclut :

— Des routes directement connectées.

— Des routes statiques configurées par l'administrateur.

— Des routes qui ont été apprises par le biais d'un protocole de routage dynamique.

2.4.1 Champs d'une table de routage

Une table de routage possède généralement les champs suivants :

— Destination : réseau à joindre + masque

— Prochain saut (passerelle) : l'adresse IP du prochain routeur sur le chemin pour atteindre le réseau de destination.

— Métrique

— Distance Administrative (DA)

— Moyen d'apprentissage de la route

2.4.2 Métrique

Dans un protocole de routage, la métrique est une mesure de la "distance" qui sépare un routeur d'un réseau de destination. Elle peut correspondre :

— Au nombre de sauts IP nécessaires pour atteindre le réseau destination, comme dans le protocole RIP

— A un coût numérique qui dépend de la bande passante des liens traversés, comme dans le protocole OSPF

— Au résultat d'un calcul plus complexe, qui tient compte de la charge, du délai, du MTU, etc.

Une métrique dont la valeur est la plus petite reflète le meilleur chemin.

2.4.3 Distance Administrative

— La Distance Administrative (DA) est une valeur numérique entière comprise entre 0 et 255 indiquant l'origine de la route (route connectée, route statique, route apprise via un protocole de routage dynamique tel que RIP, OSPF, etc.).

— La DA permet d'indiquer la préférence d'une route par rapport à une autre. En effet, plus la valeur de la DA est petite, meilleure est la route.

Tableau 2.2 – DA : Exemples.

Type de route	DA
Route directement connectée	0 (une confiance absolue)
Route statique	1 (c'est l'administrateur qui configure la route. On considère qu'il sait ce qu'il fait)
RIP	120
OSPF	110
IGRP	100 (normalement abandonné au profit de EIGRP)
IS-IS	115
BGP interne	200
BGP externe	20
EIGRP interne	90
EIGRP externe	170
Non fiable	255 (source non fiable, la route n'est pas installée dans la table de routage)

Remarques :

1. Si la DA est de 255, le routeur ne croit pas à la source de ce routage et n'installe pas le routage dans la table de routage.

2. On pourra toujours modifier la DA d'un protocole via la commande *distance* dans le mode de sous-configuration du processus de routage. Par exemple, afin que le routeur R1 privilégie les routages RIP par rapport aux routages IGRP, on procède à la configuration suivante :
   - R1(config)#router rip
   - R1(config-router)#distance 90
3. Supposons que nous avons les deux routes statiques suivantes :
   - R1(config)#ip route 192.168.0.0 255.255.255.0 1.1.1.1 name ROUTE1
   - R1(config)#ip route 192.168.0.0 255.255.0.0 2.2.2.1 name ROUTE2

   Si R1 reçoit un paquet à router vers l'adresse 192.168.0.1, il l'envoie à travers la route ROUTE1 car le masque /24 est plus restrictif que le masque /16 de la ROUTE2.
4. Si maintenant les distances administratives sont les mêmes et les masques aussi, on va employer le *load balancing* (en fraçais, la répartition de charge). Le procédé en question consiste à répartir les paquets sur les différents chemins possibles.

2.4.4 Moyen d'apprentissage d'une route

Il Indique la méthode d'apprentissage pour chaque entrée dans la table de routage tel qu'illustré dans le Tableau 2.3

Tableau 2.3 – Moyens d'apprentissage.

Code	Moyen d'apprentissage
C	Directement connectée
S	Statique
R	RIP
O	OSPF
I	IGRP
*	Route par défaut (candidate default)
D	EIGRP
EX	EIGRP externe
B	BGP

2.4.5 Remplissage d'une table de routage (routage statique)

Une table de routage d'un hôte se construit comme suit :

1. Indiquer les réseaux auxquels l'hôte est connecté
2. Indiquer la route par défaut
3. Indiquer tous les autres réseaux que l'hôte ne peux pas encore joindre avec les deux étapes précédentes

Exemple :

Soit le réseau de la Figure 2.13. Le Tableau 2.4 énumère toutes les routes conformément aux trois règles de remplissage d'une table de routage citées ci-dessus. Il s'agit d'une 1re solution où nous avons considéré une route par défaut. Le Tableau 2.5 représente quant à lui la 2e solution où nous ne considérons pas une route par défaut du moment qu'il n'y a pas un accès à internet.

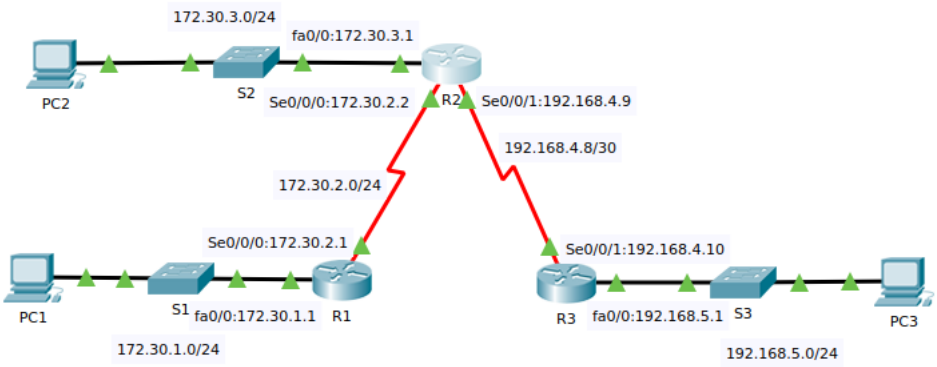


FIGURE 2.13 – Topologie réseau

TABLEAU 2.4 – Table de routage du Routeur R2 : Première solution.

Moyen d'apprentissage	Réseau à joindre	Masque	Passerelle	DA	Métrique
C	172.30.2.0	255.255.255.0	172.30.2.2	-	-
C	172.30.3.0	255.255.255.0	172.30.3.1	-	-
C	192.168.4.8	255.255.255.252	192.168.4.9	-	-
*S**	0.0.0.0	0.0.0.0	192.168.4.10	1	0
S	172.30.1.0	255.255.255.0	172.30.2.1	1	0

TABLEAU 2.5 – Table de routage du Routeur R2 : Deuxième solution.

Moyen d'apprentissage	Réseau à joindre	Masque	Passerelle	DA	Métrique
C	172.30.2.0	255.255.255.0	172.30.2.2	-	-
C	172.30.3.0	255.255.255.0	172.30.3.1	-	-
C	192.168.4.8	255.255.255.252	192.168.4.9	-	-
S	192.168.5.0	255.255.255.0	192.168.4.10	1	0
S	172.30.1.0	255.255.255.0	172.30.2.1	1	0

2.5 Protocole de routage RIP

— Le protocole RIP est un protocole de routage IGP à vecteur de distance.

— C'est un protocole open standard (open standard industry protocol), c.-à-d., il peut être utilisé avec des appareils non CISCO, y compris Juniper et d'autres équipements réseau de divers fournisseurs.
— Etant donné qu'il s'agit d'un protocole IGP, il ne peut être utilisé que pour effectuer un routage entre réseaux au sein d'un même système autonome.
— En règle générale, il convient à un réseau de petite taille.
— Il prend en charge un maximum de 15 sauts dans un chemin.
— Il utilise la métrique du nombre de sauts pour calculer le meilleur chemin d'une source à un réseau de destination.
— Il envoie des mises à jour de routage (toute la table de routage) toutes les 30 secondes et lorsque le réseau change (une route est down par exemple ou un nouveau réseau apparait, les routeurs n'attendent pas les 30 secondes pour s'échanger les tables de routage. C'est ce qu'on appelle mise à jour déclenchée ou en Anglais the **triggered update**.
— Il utilise des paquets de diffusion UDP et le port 520 pour échanger des informations de routage.
— La valeur de la distance administrative (DA) du protocole RIP est 120.
— Il existe trois versions du protocoles RIP : RIPv1, RIPv2 et RIPng qui une amélioration de RIP version 2 (RIPv2) destiné à prendre en charge les réseaux IPv_6. Il est à noter que RIPng n'est pas destiné à être utilisé avec les réseaux IPv_4.

2.6 Différents types de temporisateurs utilisés par le protocole RIP

Le Tableau 2.6 répertorie les différents types de temporisateurs utilisés par le protocole RIP pour optimiser les performances du réseau.

Remarque : Si tous les routeurs utilisaient des temporisateurs Update Timer paramétrés avec la même valeur de 30 secondes par exemple, il se produirait au bout d'un certain temps un phénomène de synchronisation de leurs annonces RIP. Pour éviter ce phénomène qui conduirait à des rafales de paquets et des risques de congestion cycliques, les valeurs effectives des minuteurs sont perturbées aléatoirement de 0 à 5 secondes.

Commandes pour implémenter RIP sous PacketTracer

— Router(config)# router rip → Active le processus RIP sur le routeur
— Router(config-router)# version 2 → Active la version 2 du RIP

2.7 RIPv1 vs. RIPv2

Les fonctionnalités de base des protocoles RIPv1 et RIPv2, telles que le nombre de sauts et la métrique, restent les mêmes. Cependant, RIPv2 est une version améliorée de RIPv1. RIPv2 fournit plus de fonctionnalités que RIPv1. Le Tableau 2.7 répertorie les principales différences entre RIPv1 et RIPv2 :

TABLEAU 2.6 – Différents types de temporisateurs utilisés par le protocole RIP.

Temporisateur	Objectif	Valeur par défaut (secondes)	Description
Temporisateur d'envoi (Update Timer)	Contrôle la fréquence d'envoi des mises à jour.	30	Spécifie l'intervalle entre les mises à jour périodiques. Chaque routeur envoie sa table de routage complète à ses voisins à cet intervalle.
Temporisateur d'invalidation (Invalid Timer)	Marque une route comme invalide en l'absence de mises à jour	180	Si un routeur ne reçoit pas de mise à jour pour une route dans ce délai, il marque la route comme invalide, indiquant que la destination n'est plus atteignable.
Temporisateur de stabilisation (Hold-down Timer)	Empêche l'utilisation de routes invalides	180	Après qu'une route ait été marquée invalide, le routeur attend le temporisateur de stabilisation avant d'accepter de nouvelles mises à jour pour la même destination afin d'éviter des changements rapides.
Temporisateur de suppression (Flush Timer)	Supprime une route inatteignable	240	Après qu'une route ait été marquée invalide, le routeur attend le temporisateur de suppression avant de supprimer la route de sa table de routage.

2.8 Protocole de routage OSPF

Il s'agit d'un protocole standard (open standard industry protocol), c.à-d., non propriétaire et par conséquent peut fonctionner sur un routeur CISCO, Juniper, Huawei etc. Il est basé sur l'algorithme Shortest Path First (SPF) ou ce qu'on désigne souvent par algorithme de Dijkstra. Il existe deux version d'OSPF, à savoir OSPFv2 (supporte uniquement IPv4) et OSPFv3 (supporte uniquement IPv6). Le Tableau 2.8 enumère quelques caractéristiques du protocole OSPF.

Tableau 2.7 – RIPv1 vs. RIPv2

Caractéristique	RIPv1	RIPv2
Famille de protocoles	IGP	IGP
Système d'adressage	Classful (Le masque de sous-réseau n'est pas transmis dans les mises à jour de routage)	Classless (Le masque de sous-réseau est transmis dans les mises à jour de routage)
Intervalle de mise à jour	30 secondes	30 secondes
Algorithme de base	Bellman-Ford	Bellman-Ford
Adresse de mise à jour du routage	Diffusion (Broadcast) (255.255.255.255)	Multicast (224.0.0.9)
Protocole et port	UDP 520	UDP 520
Unité de métrique	Saut (jusqu'à 15)	Saut (jusqu'à 15)
Élément de mise à jour	Table entière	Table entière
VLSM[a]	Ne supporte pas	Supporte
CIDR[b]	Ne supporte pas	Supporte
Authentification	Ne supporte pas	Supporte l'authentification
Réseaux discontinus (non contigus)	Ne supporte pas	Supporte

[a] Variable Length Subnet Mask
[b] Classless Inter Domain Routing

Exemple :

— Dès que le protocole OSPF est activé sur les routeurs de la Figure 2.14, chacun d'eux va envoyer un message "*hello*" pour découvrir leurs voisins directs.

— Chaque routeur va envoyer un message LSA [18] aux routeurs qui sont connectés à ses interfaces. Un message LSA contient des informations sur le routeur ID, les interfaces du routeur (fa0/0, gig0/0, se0/1/0, etc) et les adresses IPs configurées sur ces interfaces.

— Il est à noter que les routeurs, sur lesquels est configuré le protocole OSPF, ne s'échangent pas les tables de routage en entier comme le font les routeurs avec RIP. Ils vont s'échanger les message LSA.

— Chaque routeur qui réçoit un message LSA, il le sauvegarde dans une base de données

18. Link State Advertisement

TABLEAU 2.8 – Caractéristiques du protocole OSPF.

Famille de protocoles	IGP
Système d'adressage	Classless
Algorithme de base	Dijkstra
Adresse de mise à jour du routage IPv4	Multicast (224.0.0.5)
Adresse de mise à jour du routage IPv6	Multicast (FF02 : :5)
Unité de métrique	Coût=$\frac{10^8}{BandWidth}$
Nombre de sauts	Illimité
VLSM	Supporte
Réseaux non continus (non contigus)	Supporte
Protocole de transport	N'utilise ni UDP ni TCP. Le numéro du protcole utilisé : 59(hexa) 89 (Déc.)
Route summarization	Supporte
Authentification	Supporte
Loop Free Topology	Oui (car chaque routeur a une carte complète de la topologie globale du réseau

appellée LSDB [19] ou *Topological DataBase*, et le forward à son tour aux routeurs qui lui sont connectés.

— En se basant sur le LSDB, chaque routeur va créer un SPF [20] tree ayant comme racine le routeur lui même, c.-à-d., une sorte d'une carte de la topologie globale du réseau.

— En se basant sur l'arbre SPF créé, chaque routeur va construire sa table de routage.

— Il est à noter que chaque routeur possède trois types de table :

  — Table des voisins (LSDB)

  — Table de topologie (SPF tree)

  — Table de routage

19. Link State DataBase
20. Shortest Path First

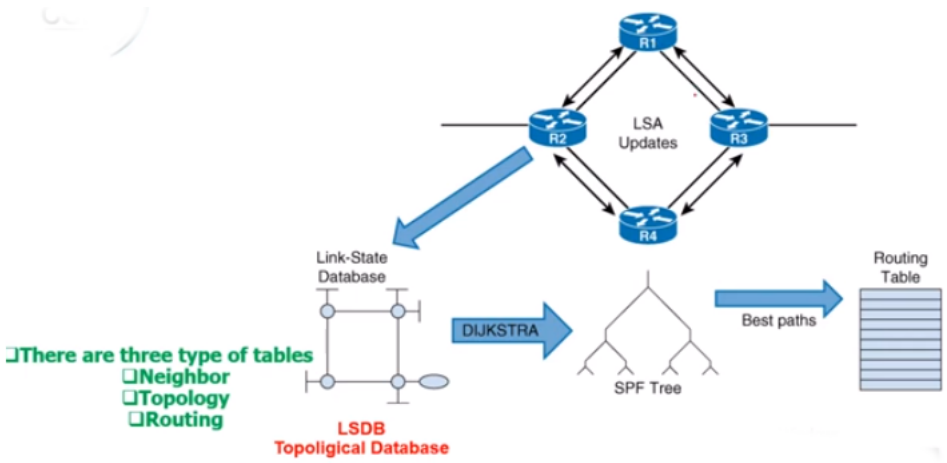


FIGURE 2.14 – OSPF : Fonctionnement [5]

2.9 Systèmes autonomes

L'Internet mondial est constitué d'un ensemble de systèmes autonomes (Autonomous Systems (AS)) interconnectés :

- Un système autonome (AS) est un ensemble de routeurs et de réseaux sous la même administration
- Les systèmes autonomes sont identifiés par des numéros de 32 bits, appelés numéros AS (ASN) (à l'origine, l'espace des numéros était limité à 16 bits, mais il a été augmenté à 32 bits)
- Les paquets IP sont transmis entre des systèmes autonomes sur des chemins établis par un protocole de passerelle extérieure (EGP)
- Au sein d'un système autonome, les paquets IP sont transmis sur des chemins établis par un protocole de passerelle intérieure (IGP)

2.10 Conclusion

Ce chapitre a exploré en profondeur le concept fondamental du routage dans les réseaux informatiques. Nous avons notamment expliqué les deux types de routage, à savoir le routage statique et le routage dynamique, et avons expliqué comment ces méthodes sont utilisées pour diriger le trafic à travers les réseaux. Nous avons aussi mis l'accent sur l'importance des tables de routage dans la prise de décisions de routage efficaces. A la fin du Chapitre, nous avons abordé la notion des systèmes autonomes.

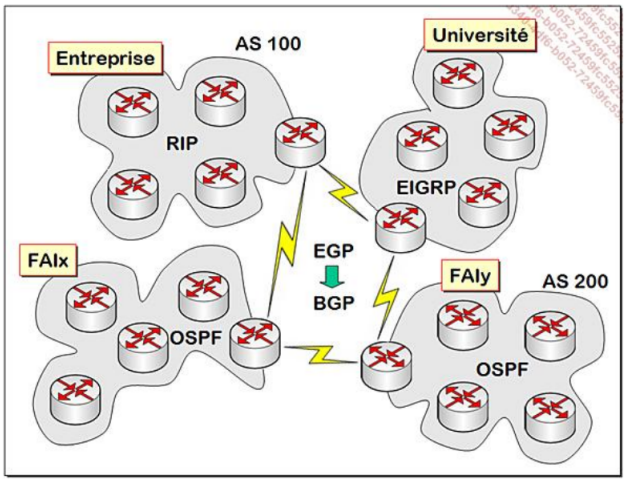


FIGURE 2.15 – Systèmes autonomes, IGP et EGP [7].

Chapitre 3

Protocole IPv_4

3.1 Introduction

Le protocole IPv_4, abréviation de "Internet Protocol version 4", est l'un des protocoles de communication fondamentaux permettant le fonctionnement d'Internet. Il utilise une adresse IP, composée de quatre chiffres décimaux, allant chacun de 0 à 255, séparés par des points (notation décimale pointée), pour identifier de manière unique chaque nœud sur un réseau. L'epuisement imminent des adresses IPv_4 disponibles a conduit à l'adoption progressive d'IPv_6, une version plus récente et évolutive du protocole. Malgré cela, IPv_4 reste largement utilisé dans le paysage numérique actuel.

3.2 Protocole IPv_4 (Internet Protocol v_4)

IPv_4 est un protocole de la couche 2 du modèle TCP/IP [1] (couche 3 du modèle OSI [2]), permet aux couches supérieures de faire abstraction de l'ensemble des réseaux (locaux et étendus) qu'il faut parfois traverser pour acheminer un paquet dans le réseau. Les couches supérieures n'ont en effet pas à se soucier de la route parfois compliquée que les paquets doivent emprunter. Elles voient la liaison comme une liaison directe entre la machine émettrice et la machine réceptrice.

Notons aussi que pour s'affranchir de la longueur du datagramme admissible par les réseaux parcourus (selon la MTU [3]), le protocole IPv_4 assure la fragmentation et le réassemblage des datagrammes. Ce mécanisme est appelé mécanisme de fragmentation que nous aborderons dans le prochain Chapitre.

3.3 Fonctionnement du protocole IPv_4

Les données constituant le datagramme à émettre sont fournies par la couche de transport. Deux scénarios sont possibles :

1. Transmission Control Protocol/Internet Protocol
2. Open Systems Interconnexion
3. Maximum Transmission Unit

TABLEAU 3.1 – Exemples de valeurs de MTU.

Réseau	MTU (Octets)
X.25	576
Ethernet	1500
Point to Point Protocol Over Ethernet (PPPoE)	1500 (peut être réduite à 1492 octets)
Wi-Fi (802.11)	2304 à 7935 (peut varier selon les variantes)
Fiber Distributed Data Interface (FDDI)	4352
High-level Data Link Control (HDLC)	1500 (la valeur peut varier selon les implémentations)
Frame Relay	1600 (peut varier selon les implémentations)

1. Si la station destinataire se trouve sur le même réseau que la station émettrice, le datagramme est alors envoyé directement vers la station destinataire.
2. Si la station destinataire se trouve sur un autre réseau, la couche IPv_4 envoie le datagramme vers une passerelle (routeur). Cette passerelle se charge d'acheminer le datagramme vers la station destinataire.

3.4 Caratéristiques du protocole IPv_4

— **Non Connecté** : IPv_4 est un protocole de couche réseau sans connexion, ce qui signifie qu'il ne crée pas de connexion préalable entre les expéditeurs et les destinataires avant de transmettre des données.

— **Non Sécurisé** : IPv_4 n'inclût pas de mécanismes de sécurité intégrés. Il n'assure pas la confidentialité ni l'intégrité des données lors de leur transmission.

— **Best Effort** : IPv_4 suit le principe du "best effort", ce qui signifie qu'il fait de son mieux pour livrer les paquets à destination, mais il n'offre aucune garantie absolue de livraison. Un paquet peut être perdu par exemple à cause de la saturation des files d'attente et ni l'émetteur ni le récepteur ne sont au courant.

— **Routage** : IPv_4 utilise des tables de routage pour déterminer le chemin optimal des paquets à travers le réseau. Le routage peut être statique ou dynamique.

— **ARP** [4] : IPv_4 utilise ARP pour résoudre les adresses IP en adresses MAC dans les réseaux locaux.

— **Fragmentation et Réassemblage** : Lorsque des paquets sont trop grands pour être transmis sur un lien, IPv_4 prend en charge la fragmentation des paquets à l'émetteur et leur réassemblage au récepteur. Nous tenons à souligner que chaque réseau de niveau 2 (OSI) possède sa propre valeur de MTU qui spécifie la taille maximale que le média réseau peut transporter en un seul PDU [5](voir Tableau 4.1) .

Remarque Importante :

4. Address Resolution Protocol
5. Protocol Data Unit

— Le service de base offert par IPv_4 est l'émission et la réception de paquets de données appelés datagrammes. Ce service est dit "non fiable" dans la mesure où la perte (ou l'altération) d'un paquet pendant son transport n'est pas résolue. Aucun mécanisme permettant de récupérer ces erreurs n'existe dans la couche IPv_4.

— L'exception IPv_4 : Généralement, les protocoles contiennent eux-mêmes les mécanismes nécessaires pour signaler les erreurs qui sont de leur ressort. IPv_4 fait exception. En effet, le protocole ICMP [6] a été défini pour les situations d'anomalies.

3.5 Datagramme IPv_4

Un datagramme consiste en deux parties :

— Entête (header)

— Données ou charge utile (en Anglais, payload)

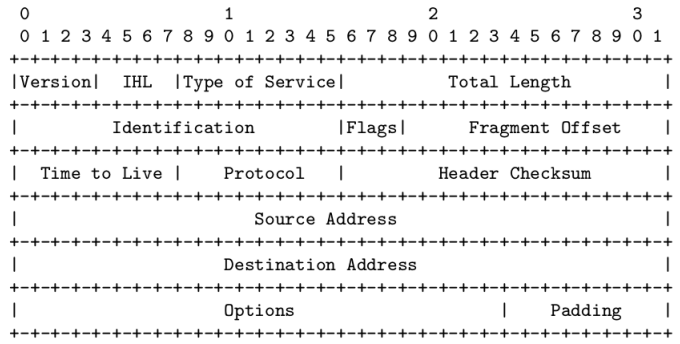


FIGURE 3.1 – Entête d'un datagramme IPv_4 (RFC 791).

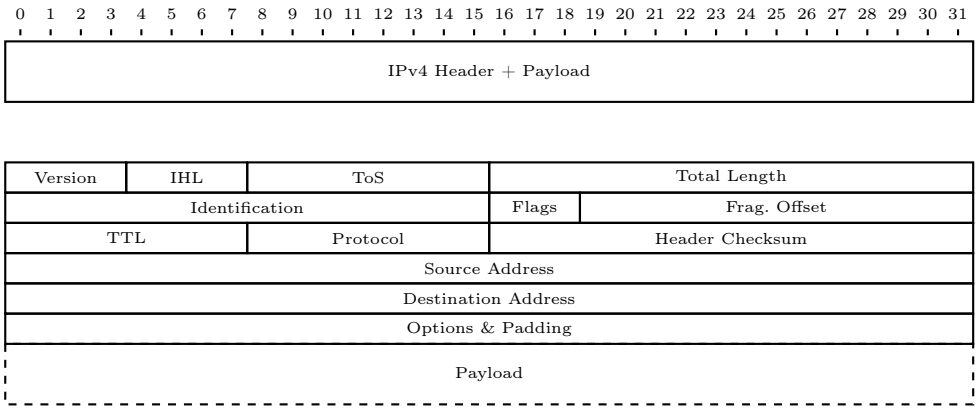


FIGURE 3.2 – Entête+Charge Utile d'un datagramme IPv_4

6. Internet Control Message Protocol

— **Version** (**4 bits**) : Le numéro de version IP, en général 4 pour IPv_4.

— **IPv_4 Header Length (IHL)** (**4bits**) : est la longueur de l'entête du datagramme exprimée en nombre entier de mots de 32 bits chacun. Si le champs options n'est pas présent alors l'entête occupe 20 octets (5 mots).

— **Total Length** (**16 bits**) : Indique la taille totale en octets du datagramme IPv_4 (entête + charge utile). La taille maximale d'un datagramme IP = ($2^{16}-1$), soit 65635 octets. Notons que la taille effective des datagrammes ne doit pas dépasser la MTU (Maximum Transfert Unit) qui est taille maximale d'une trame obtenue par encapsulation du datagramme de la couche IPv_4 au niveau de la couche d'accès réseau.

— Le champ **Type of Service (TOS)** dans l'en-tête IPv_4 a évolué au fil du temps (voir Figure 3.3, et actuellement, il est actuellement divisé en deux parties principales :

  — **Differentiated Services Code Point (DSCP)** : Il s'agit de la partie la plus significative des 8 bits du champ TOS. Les 6 bits de poids fort sont utilisés pour le DSCP, qui permet de définir différents niveaux de qualité de service (QoS) pour le traitement des paquets.

  — **Explicit Congestion Notification (ECN)** : Les 2 bits de poids faible sont utilisés pour le champ ECN. Ces bits permettent aux routeurs de signaler la présence de congestion dans le réseau aux hôtes source et destination, afin d'ajuster leur comportement de transmission.

  Auparavant, ToS avait la structure suivante :

  — **Précédence (Precedence)** : Ce sous-champ définit la priorité du datagramme par rapport aux autres datagrammes. Il est composé de trois bits qui indiquent la priorité (par exemple, routine, prioritaire, immédiat) :

    — **Routine (000)** : Il s'agit du niveau de priorité le plus bas, indiquant un trafic routinier normal.

    — **Priority (001)** : Légèrement supérieur à la routine, utilisé pour le trafic prioritaire.

    — **Immediate (010)** : Utilisé pour un trafic immédiat qui nécessite une livraison rapide.

    — **Flash (011)** : Utilisé pour un trafic éclair qui doit être livré le plus rapidement possible.

    — **Flash Override (100)** : Similaire à Flash mais avec une priorité plus élevée.

    — **Critical (101)** : Utilisé pour un trafic critique qui nécessite une attention immédiate.

    — **Internetwork Control (110)** : Réservé pour le contrôle des réseaux interconnectés.

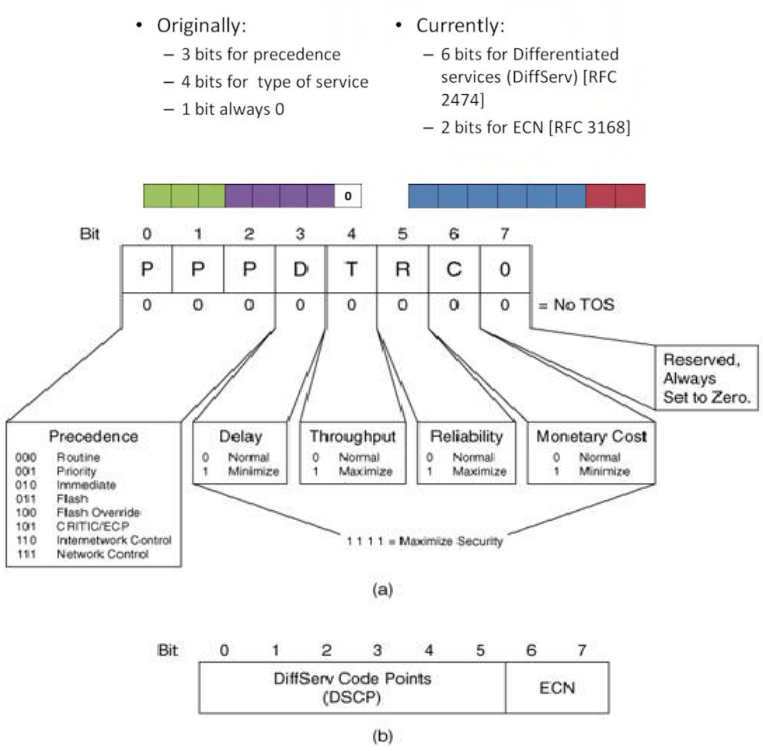


FIGURE 3.3 – Détail du champ ToS au fil du temps [9] [10]

- **Network Control (111)** : Réservé pour le trafic de contrôle du réseau, indiquant le niveau de priorité le plus élevé pour les fonctions critiques de contrôle du réseau nécessitant une attention et un traitement immédiats.

- **Delay** : Ce bit indique si le datagramme est sensible au délai. Si le bit est défini, cela signifie que le datagramme ne peut pas supporter de délai important.
- **Throughput** : Ce bit indique si le datagramme nécessite un débit élevé. S'il est défini, cela signifie que le datagramme nécessite un débit élevé.
- **Reliability** : Ce bit indique si le datagramme nécessite une transmission fiable. S'il est défini, cela signifie que le datagramme nécessite une transmission fiable.
- **Cost** : Ce bit indique le coût relatif du datagramme. S'il est défini, cela signifie que le datagramme est coûteux à transmettre.
- **Reserved** : Ce bit est réservé à des fins futures et doit être mis à zéro.

- Les trois champs, **Identification**, **Flags** et **Fragment Offset** interviennent dans le processus de fragmentation des datagrammes IPv_4 que nous verrons dans le prochain Chapitre :
  - **Identification (16 bits)** : Permet à la station destinataire de déterminer à quel datagramme appartient le fragment reçu. Tous les fragments d'un même datagramme contiennent la même valeur d'identification. Notons que les fragments d'un datagramme sont aussi des datagrammes.

  - **Flags (Drapeaux)(3 bits)** :
    - bit 0 : est **réservé**
    - bit 1 : appelé **DF** (Don't Fragment) "ne pas fragmenter". Si à 0 alors autorisation de fragmenter. Si à 1 alors ne pas fragmenter.
    - bit 2 : appelé **MF** (More Fragments) "encore des fragments". Si à 1 alors d'autres fragments arrivent. Si à 0 alors il s'agit du dernier fragment. Si le datagramme n'est pas fragmenté, le bit 2 est à 0.
- **Fragment Offset (13 bits)** : C'est le déplacement de pointeur de fragment qui sert de compteur (par groupe de huit octets) pour calculer le déplacement de la charge utile du fragment courant dans le datagramme d'origine. Ce champ contient 13 bit et par conséquent on a un de maximum 8192 fragments par datagramme.
- **TTL (8 bits)** : Le champ TTL sur 8 bits limite la durée de vie d'un paquet IP, l'empêchant de circuler indéfiniment dans le réseau. Il décompte le temps de séjour (en secondes) du datagramme dans le réseau avec une limite maximale de 255 secondes. Il doit être décrémenté d'une unité à chaque saut (lors de la traversée d'un routeur) et de plusieurs unités lorsqu'il est en file d'attente pendant un temps important dans un routeur. En pratique, il n'est décrémenté qu'à chaque saut.
- **Protocol (8 bits)** : Le champ Protocole sur 8 bits identifie le protocole de niveau supérieur utilisé dans la partie de données du datagramme IP, tel que TCP (6), UDP (17) ou ICMP (1), IGMP [7] (2) et OSPF (89) (voir Tableau 3.2).

TABLEAU 3.2 – Quelques valeurs courantes pour le champ Protocol.

Valeur du protocole (en Hexadécimal)	Valeur du protocole (en Décimal)	Nom
6	6	TCP
11	17	UDP
1	1	ICMP
2	2	IGMP
59	89	OSPF

- **Header Checksum (16 bits)** : champs calculé à partir de l'en-tête du datagramme pour assurer l'intégrité. Toute modification en chemin de la charge utile n'est pas détectée par IP. C'est au protocole de la couche supérieur de procéder à sa propre vérification d'intégrité des données et de demander une retransmission en cas de problème.
- **Source Address et Destination Address (32 bits chacune)** : Adresses IPv_4 de la machine source et de la machine destination respectivement.

7. Internet group management protocol

— **Options** : Le champ Options permet une fonctionnalité supplémentaire et est utilisé à diverses fins, telles que le marquage temporel, la sécurité et le débogage.

— **Padding** : Le champ Padding (en Français, bourrage ou remplissage) sert à aligner les options sur une limite de 32 bits.

3.6 Conclusion

Dans ce Chapitre, nous avons abordé le fonctionnement du protocole de couche réseau, en l'occurrence IP$v4$, mettant l'accent sur son rôle fondamental dans l'acheminement des données à travers l'Internet. Cette exploration du protocole IP$v4$ nous a permis de comprendre le format d'un datagramme IP$v4$ avec ses deux principales parties, à savoir l'en-tête et la charge utile.

Chapitre 4

Fragmentation des paquets IPv_4

4.1 Introduction

La fragmentation des paquets IPv_4 est un processus essentiel dans la transmission des données à travers un réseau. Elle assure que les données peuvent être transmises efficacement à travers des réseaux hétérogènes tout en respectant les contraintes de taille imposées par les différents équipements du réseau (la MTU [1]).

Lorsqu'un paquet est envoyé d'une source à une destination, il doit souvent traverser plusieurs réseaux intermédiaires avec des MTU différentes. Si la taille d'un paquet IPv_4 est plus grande que la MTU d'un réseau intermédiaire, le paquet doit être fragmenté en plusieurs fragments plus petits. Chaque fragment contient une partie des données du paquet d'origine et est envoyé individuellement. Une fois arrivés à destination, les fragments sont réassemblés pour reconstituer le paquet initial. Cependant, elle peut également entraîner des problèmes de performances et de sécurité. Par conséquent, il est important de gérer efficacement ce processus.

4.2 Maximum Transmission Unit (MTU)

— La MTU, ou Maximum Transmission Unit, représente la taille maximale d'un paquet de données qu'un protocole de la couche réseau peut transmettre sans nécessiter de fragmentation. Elle est généralement mesurée en octets. Autrement dit, la MTU spécifie la taille maximale que le média réseau peut transporter en un seul PDU [2]

— La MTU est une spécification importante pour assurer une transmission efficace des données à travers un réseau car elle détermine la taille maximale des paquets qui peuvent être envoyés d'un nœud à un autre sans être fragmentés.

— chaque type de réseau de couche 2, tel qu'Ethernet ou PPP, a sa propre MTU spécifique déterminée par les caractéristiques de cette technologie particulière (voir Tableau 4.1). Chacun de ces réseaux peut avoir des exigences différentes en termes de taille maximale

1. Maximum Transmission Unit
2. Protocol Data Unit

Tableau 4.1 – Exemples de valeurs de MTU.

Réseau	MTU (Octets)
X.25	576
Ethernet	1500
Point to Point Protocol Over Ethernet (PPPoE)	1500 (peut être réduite à 1492 octets)
Wi-Fi (802.11)	2304 à 7935 (peut varier selon les variantes)
Fiber Distributed Data Interface (FDDI)	4352
High-level Data Link Control (HDLC)	1500 (la valeur peut varier selon les implémentations)
Frame Relay	1600 (peut varier selon les implémentations)

des paquets qu'il peut transporter sans fragmentation, influençant ainsi la MTU associée à chaque réseau de couche 2.

Pour résumer, nous dirons que :

— La couche 2 (liaison de données) définit la MTU (Maximum Transmission Unit), qui représente la taille maximale d'un datagramme qu'elle peut accepter car c'est elle qui va encapsuler ce datagramme IP$v4$ dans une trame. Dans le cas du protocole Ethernet, la MTU standard est de 1500 octets.

— Lorsque la couche 3 (réseau) génère des datagrammes, elle veille à ce qu'ils ne dépassent pas la MTU définie par la couche 2.

— Si un datagramme est plus grand que la MTU, la couche 3 doit le fragmenter en plusieurs paquets plus petits, chacun respectant la taille maximale autorisée par la couche 2.

.

4.3 Fragmentation

— La fragmentation des données, processus consistant à diviser un paquet en fragments plus petits, peut se produire à divers endroits du réseau. Contrairement à la croyance courante, ce n'est pas exclusivement la responsabilité des routeurs.

— Les émetteurs, tels que les PC, peuvent également prendre en charge la fragmentation, notamment lorsqu'ils envoient des données dépassant la MTU du réseau.

— Les routeurs peuvent être impliqués dans le processus de fragmentation, notamment s'ils rencontrent des interfaces avec des MTU différentes le long du chemin de transmission.

— Si la MTU de la liaison ne permet de transporter le paquet entier alors envoyer le paquet en fragments tel que l'illustre la Figure 4.1.

— Le fait de fragmenter un datagramme revient à le diviser en plusieurs morceaux, chaque morceau ayant le **même format que le datagramme d'origine** comme le montre la Figure 4.2 où une machine dans un réseau ethernet souhaite envoyer un datagramme de 1600 octets. Sachant qu'une trame Ethernet pouvant transporter un datagramme de 1500

FIGURE 4.1 – Fragmentation d'un datagramme IP$v4$ [2].

octets (1518 - 18 octets d'en-tête Ethernet), il va falloir donc fragmenter ce datagramme en deux fragments de 1480 et 100 octets de payload (charge utile) respectivement et ayant chacun un en-tête de 20 octets.

— Chaque nouveau fragment possède un en-tête, qui reprend la plupart des informations de l'en-tête d'origine et le plus de données possible.

— Un fragment doit tenir dans une seule trame.

— La fragmentation est un mécanisme coûteux pour les routeurs.

— La fragmentation n'existe plus en IPv_6.

— **C'est à la destination finale de réaliser le réassemblage des fragments pour reconstituer le datagramme d'origine** (voir Figure 4.3). En effet, Le réassemblage des fragments se produit uniquement à la destination finale du datagramme. Les routeurs intermédiaires ne sont généralement pas responsables du réassemblage, à l'exception de certains dispositifs spécifiques tels que les pare-feu (firewalls). Les routeurs se contentent de router les fragments vers leur destination en fonction des informations contenues dans l'en-tête IPv_4, sans effectuer le réassemblage.

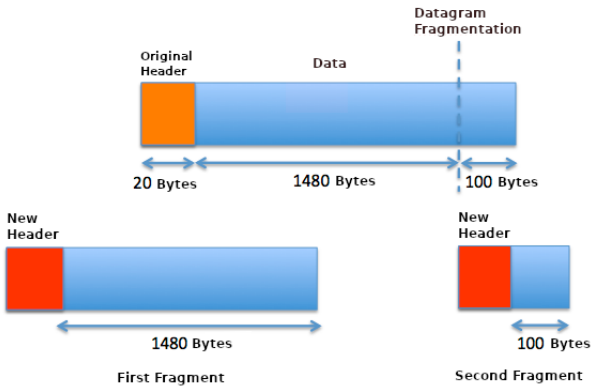


FIGURE 4.2 – Fragmentation d'un datagramme IPv_4

Remarque très importante

— La taille des données d'un datagramme IP$v4$ doit obligatoirement être un multiple de 8.

— Le dernier fragment provenant d'un datagramme fragmenté n'a pas besoin d'avoir une taille de données multiple de 8 car il n'y aura aucun fragment après lui et donc on n'a pas besoin d'avoir un nombre de données multiple de 8.

La Figure 4.4 illustre la remarque précédente.

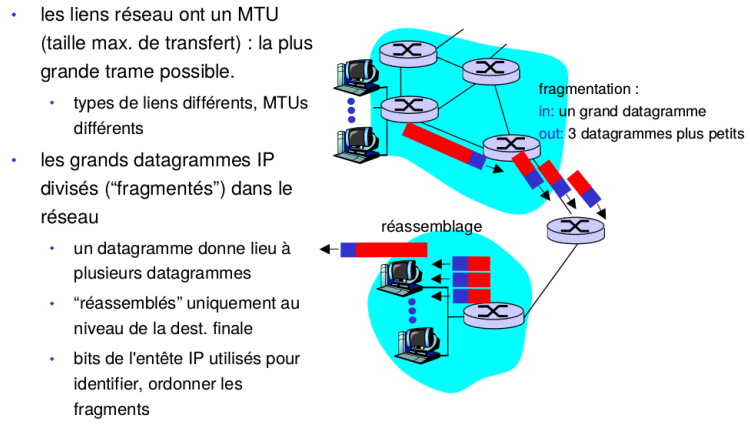


FIGURE 4.3 – Réassemblage des fragments d'un datagramme IPv_4 [11].

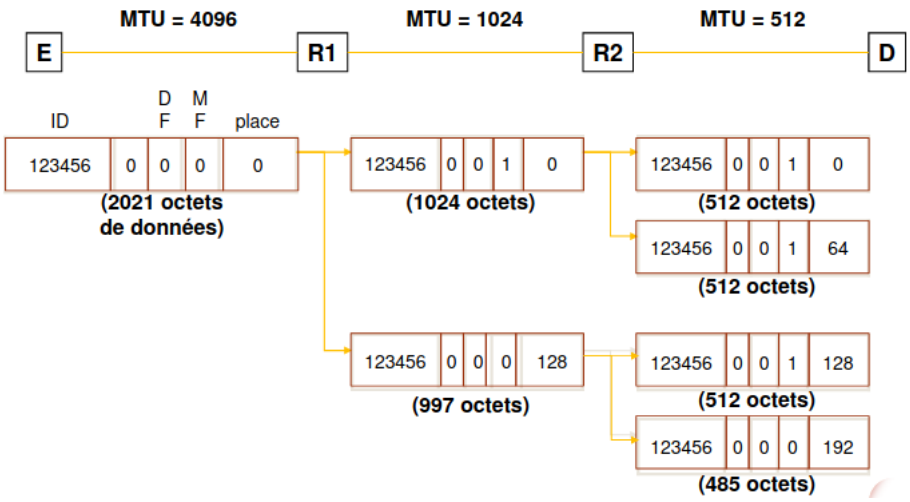


FIGURE 4.4 – Exemple de fragmentation [2]

.

4.4 Exemples

A travers les exemples de la Section courante, nous allons découvrir comment les fragments issus d'un datagramme vont être identifiés, et comment une machine qui va recevoir ces fragments va pouvoir les réassembler.

4.4.1 Exemple 1

La machine PC1 du réseau de la Figure 4.5 veut envoyer un datagramme de 5600 octets à la machine PC2. Les MTUs pour les réseaux 192.168.10.0/24 et 192.168.11.0/24 sont identiques et valent 1500 octets (réseau ethernet habituel). Nous allons donc devoir fragmenter.

Comment trouver le nombre de fragments et leurs tailles respectives :

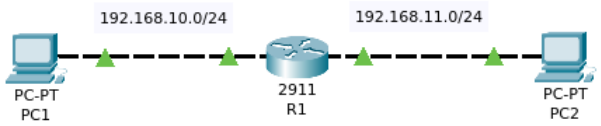


FIGURE 4.5 – Exemple 1

.

1. 1500 octets - 20 octets = 1480 octets. 1480 est un multiple de 8, 1480= 8*185 (sinon, ou aurait cherché le plus proche entier de 1480 qui est un multiple de 8).
2. 5600 octets - 20 octets = 5580 octets
3. $\frac{5580}{1480}$= 3.77027027. Donc, on aura 3 fragments de 1480 et un fragment de 5580-1480*3 = 1140

Le datagramme d'origine et les 4 fragments résultants :

1. Datagramme d'origine :
   - ID : 10254 (par exemple)
   - 0/DF/MF : 0/0/0
   - Charge Utile (CU) : 5580 octets
   - FO (Fragment Offset) : 0
2. 1^{er} fragment :
   - ID : 10254
   - 0/DF/MF : 0/0/1
   - Charge Utile (CU) : 1480 octets
   - FO (Fragment Offset) : 0
3. 2^{me} fragment :
   - ID : 10254
   - 0/DF/MF : 0/0/1
   - Charge Utile (CU) : 1480 octets
   - FO (Fragment Offset) : 1480/8=185
4. 3^{me} fragment :
   - ID : 10254
   - 0/DF/MF : 0/0/1
   - Charge Utile (CU) : 1480 octets
   - FO (Fragment Offset) : (1480*2)/8=370
5. 4^{me} fragment :
   - ID : 10254

— 0/DF/MF : 0/0/0

— Charge Utile (CU) : 1140 octets

— FO (Fragment Offset) : (1480*3)/8=555

Vérification des calculs effectués :

— Pour s'assurer que nous n'avons pas fait d'erreur de calcul, nous allons faire une petite vérification.

— Le Fragment Offset du dernier fragment représente toutes les données des fragments précédents. Si on y ajoute le nombre de ses données, on devrait retrouver le nombre de données total. Vérifions cela :

— (555*8)+1140=4440+1140=5580 octets, ce qui correspond bien.

— La valeur du Fragment Offset dépend de la taille des données contenues dans les fragments précédents et non dans celles contenues dans le fragment lui-même.

4.4.2 Exemple 2

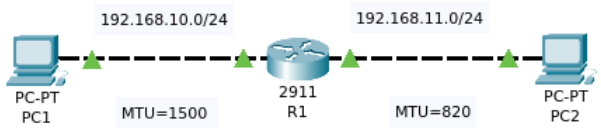


FIGURE 4.6 – Exemple 2 : même réseau que précédemment, mais la MTU sur le second réseau est de 820 octets.

— Le découpage va donc être, dans un premier temps, le même que le précédent.

— Mais une fois arrivés au niveau du routeur R1, la table de routage de celui-ci va lui dire de les envoyer sur le second réseau qui a une MTU de 820 octets. Le routeur R1 a deux choix :

1. Soit il réassemble les fragments et les découpe une nouvelle fois
2. Soit il découpe directement les fragments reçus en fragments plus petits

— C'est en fait la seconde solution qui est utilisée en réseau, car aucun matériel réseau n'a le droit de réassembler des fragments, sauf la machine destinatrice (et les firewalls parfois pour des raisons de sécurité).

— Le routeur R1 va donc découper chacun des fragments qu'il a reçu en fonction de la nouvelle MTU qui est égale à 820 octets. Les nouveaux fragments vont donc avoir un maximum de 800 octets de données.

Comment trouver le nombre de fragments et leurs tailles respectives :

1. 820 octets - 20 octets = 800 octets. 800 est un multiple de 8, 800= 8*100 (sinon, ou aurait cherché le plus proche entier de 800 qui est un multiple de 8).

2. $\frac{1480}{800}$= 1.85. Donc, on aura 1 fragment de 800 et un fragment de 1480-800 = 680. Ceci est valable pour les trois fragments de 1480 octets
3. $\frac{1140}{800}$= 1.425. Donc, on aura 1 fragment de 800 et un fragment de 1140-800 = 340.

Le datagramme d'origine et les 8 fragments résultants :

1. Datagramme d'origine :
   - ID : 10254 (par exemple)
   - 0/DF/MF : 0/0/1
   - Charge Utile (CU) : 1480 octets
   - FO (Fragment Offset) : 0

   (a) 1^{er} fragment :
   - ID : 10254
   - 0/DF/MF : 0/0/1
   - Charge Utile (CU) : 800 octets
   - FO (Fragment Offset) : 0

   (b) 2^{me} fragment :
   - ID : 10254
   - 0/DF/MF : 0/0/1
   - Charge Utile (CU) : 680 octets
   - FO (Fragment Offset) : 800/8=100

2. Datagramme d'origine :
   - ID : 10254 (e.g.)
   - 0/DF/MF : 0/0/1
   - Charge Utile (CU) : 1480 octets
   - FO (Fragment Offset) : 185

   (a) 1^{er} fragment :
   - ID : 10254
   - 0/DF/MF : 0/0/1
   - Charge Utile (CU) : 800 octets
   - FO (Fragment Offset) : 185 ou (800+680)/8=185

   (b) 2^{me} fragment :
   - ID : 10254
   - 0/DF/MF : 0/0/1
   - Charge Utile (CU) : 680 octets

— FO (Fragment Offset) :(185+800/8)=285 ou (2*800+680)/8=285

3. Datagramme d'origine :
   — ID : 10254 (e.g.)
   — 0/DF/MF : 0/0/1
   — Charge Utile (CU) : 1480 octets
   — FO (Fragment Offset) : 370

   (a) 1^{er} fragment :
      — ID : 10254
      — 0/DF/MF : 0/0/1
      — Charge Utile (CU) : 800 octets
      — FO (Fragment Offset) : 370 ou (2*800+680*2)/8=370

   (b) $2me$ fragment :
      — ID : 10254
      — 0/DF/MF : 0/0/1
      — Charge Utile (CU) : 680 octets
      — FO (Fragment Offset) :(370+800/8)=470 ou (3*800+680*2)/8=470

4. Datagramme d'origine :
   — ID : 10254 (e.g.)
   — 0/DF/MF : 0/0/0
   — Charge Utile (CU) : 1140 octets
   — FO (Fragment Offset) : 555

   (a) 1^{er} fragment :
      — ID : 10254
      — 0/DF/MF : 0/0/1
      — Charge Utie (CU) : 800 octets
      — FO (Fragment Offset) : 555 ou (3*800+680*3)/8=555

   (b) 2^{me} fragment :
      — ID : 10254
      — 0/DF/MF : 0/0/0
      — Charge Utile (CU) : 340 octets
      — FO (Fragment Offset) :(555+800/8)=655 ou (4*800+680*3)/8=655

Comment vérifier les calculs effectués :

Pour s'assurer que nous n'avons pas fait d'erreur de calcul, nous allons faire une petite vérification :

— Le Fragment Offset du dernier fragment représente toutes les données des fragments précédents. Si on y ajoute la taille de ses données, on devrait retrouver la taille totale des données :

  — (655*8)+340=5240+340=5580 octets, ce qui correspond bien.

— Nous tenons à rappeler que la valeur du Fragment Offset dépend de la taille des données contenues dans les fragments précédents et non dans celles contenues dans le fragment lui-même.

Remarque Importante :

Le flag More Fragment n'est positionné à 0 que pour le dernier fragment même si plusieurs fragmentations successives ont eu lieu.

4.5 Organigramme de transmission d'un paquet IPv_4

La Figure 4.7 détaille les différentes étapes de transmission d'un datagramme IPv_4, d'une machine à une autre.

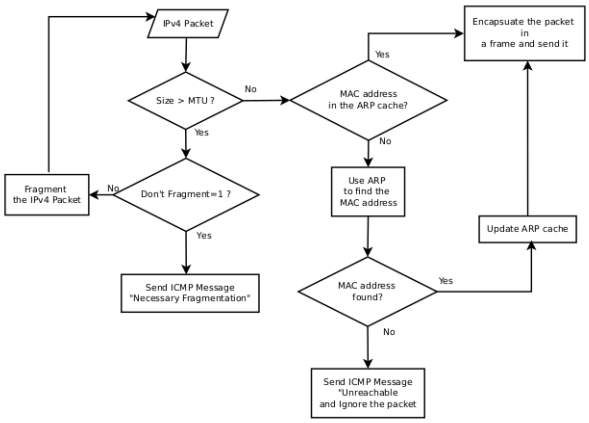


FIGURE 4.7 – Les étapes d'envoi d'un datagramme IPv_4.

4.6 Conclusion

Dans ce Chapitre, nous avons abordé une technique cruciale pour assurer la transmission fluide des données à travers des réseaux hétérogènes, en l'occurrence la fragmentation des datagramme IP$v4$. Cette dernière conssite en effet à diviser les datagrammes IP$v4$ en fragments pour s'adapter à la taille maximale des unités de transfert (MTU) des réseaux de niveau 2. Nous avons examiné les étapes du processus, y compris la division en fragments et le réassemblage ultérieur.

Chapitre 5

Trames ethernet

5.1 Introduction

La trame Ethernet est la base de la transmission des données sur les réseaux locaux (LAN [1]) utilisant la technologie Ethernet. Chaque périphérique sur le réseau a une interface réseau avec une adresse MAC unique, permettant ainsi l'acheminement efficace des données à travers le réseau. Notons que chaque type de réseau (Ethernet, HDLC [2], PPP [3], etc.) a son propre format de trame spécifique.

Dans ce Chapitre, nous allons explorer le format d'une trame Ethernet, comprendre ses composants essentiels, et découvrir comment elle assure le transfert de données dans un LAN.

5.2 Trames Ethernet

La structure de la trame Ethernet a été initialement définie par Xerox, Digital et Intel, formant le triumvirat fondateur. Plus tard, cette structure a été normalisée par l'IEEE (Institute of Electrical and Electronics Engineers). Ainsi, deux versions de trames Ethernet coexistent : la version originale du triumvirat (Ethernet II) et celle normalisée par l'IEEE (IEEE 802.3). Ethernet II est le format le plus couramment utilisé, tandis que IEEE 802.3 est un format alternatif. Les deux formats diffèrent légèrement dans leur structure d'en-tête comme l'illustrent les Figures 5.1 et 5.2.

1. Local Area Network
2. High Level Data Link Control
3. Point to Point Protocol

Préambule	**@ MAC Destination**	**@ MAC Source**	**Ether Type**	**Données**	**FCS**
8 octets	6 octets	6 octets	2 octets	46 à 1500 octets	4 octets

FIGURE 5.1 – Format d'une trame Ethernet II

Préambule	SFD	**@ MAC Destination**	**@ MAC Source**	**Long.**	**Données**	**FCS**
7 octets	1 octets	6 octets	6 octets	2 octets	46 à 1500 octets	4 octets

FIGURE 5.2 – Format d'une trame Ethernet IEEE 802.3

5.3 Ethernet II vs IEEE 802.3

— On distingue une trame Ethernet II d'une trame IEEE 802.3 selon la valeur champ Ether Type/Long. Si la valeur de ce champ < 1500, il s'agit d'une trame IEEE 802.3. Si cette valeur est > 1500, il s'agit alors d'une trame Ethernet II.

— Le format de trame IEEE 802.3 est principalement utilisé pour les échanges entre périphériques intermédiaires tels que les commutateurs, notamment pour des protocoles comme STP.

— En revanche, Ethernet II est le format de trame privilégié utilisé par les stations pour la transmission de données.

5.4 Description des principaux Champs d'une trame ethernet

Une trame Ethernet est une unité de données structurée utilisée dans les réseaux Ethernet pour la communication entre les périphériques. Elle comprend plusieurs champs qui sont utilisés pour le processus d'encapsulation et la transmission des données sur un réseau Ethernet, notamment le préambule, l'adresse MAC de destination, l'adresse MAC source, le type de protocole/Longueur, les données à transférer, et une somme de contrôle pour la détection d'erreurs (voir les Figures 5.1 et 5.2).

5.4.1 Préambule

Un préambule composé de séquences de bits spécifiques, indique le début imminent d'une trame. Il aide les récepteurs à synchroniser leurs horloges avec le signal émis. En effet :

— Lorsqu'une carte réseau envoie une trame Ethernet, elle n'envoie pas tout de suite le contenu de la trame :

  — Elle se synchronise avec la carte réceptrice en premier lieu. Pour cela, elle envoie une suite de zéros et de uns (7 octets valant $(10101010)_2$ ou $(AA)_{16}$ chacun).

  — Ensuite, elle indique qu'elle va commencer à envoyer une trame en envoyant 1 octet valant $(10101011)_2$ ou $(AB)_{16}$ (SFD[4] qui marque le début de la trame, c.-à-d., il indique à la carte réceptrice que le début de la trame va commencer). Autrement, **Le SFD** est un champ court qui marque la fin du préambule et le début du reste de la trame. Il indique le début formel de la trame.

  — La Figure 5.3 montre les 8 octets de synchronisation constituant le préambule d'une trame Ethernet II.

4. Start Frame Delimiter

— Enfin, elle commence à envoyer le contenu de la trame.

— Il est à noter que tout ce qui se passe avant est appellé préambule et n'est pas visible lors d'une analyse de trames avec un logiciel comme **Wireshark** par exemple.

1er octet	2e octet	3e octet	4e octet	5e octet	6e octet	7e octet	8e octet
10101010	10101010	10101010	10101010	10101010	10101010	10101010	10101011

FIGURE 5.3 – Les 8 octets formant un préambule d'une trame Ethernet II.

5.4.2 Adresse de destination (Destination Address)

Une adresse physique de 6 octets (48 bits) qui identifie la station destinataire de la trame. Dans une trame diffusée, cette adresse peut être une adresse de diffusion (FF :FF :FF :FF :FF :FF) pour indiquer que la trame est destinée à toutes les stations du réseau local.

5.4.3 Adresse source (Source Address)

Une adresse physique de 6 octets qui identifie la station émettrice de la trame.

5.4.4 EtherType/Longueur (Type/Length)

Un champ de 2 octets qui spécifie soit le type de protocole encapsulé dans la trame (par exemple, IPv4, ARP, ...), soit la longueur de la trame si elle utilise le format IEEE 802.3. Le Tableau 5.1 indique des exemples de valeurs que peut prendre ce champ dans le cas d'une trame Ethernet II.

EtherType	Protocole
0x0800	IPv4
0x8100	802.1q (Encapsulation VLAN)
0x0806	ARP
0x8035	RARP
0x86DD	IPv6

TABLEAU 5.1 – Exemples de valeurs que peut prendre le champ EtherType

5.4.5 Données (Data)

Les données réelles de la trame, pouvant varier en taille en fonction du type de protocole transporté. Le champ en question doit avoir une taille entre 46 et 1500 octets (taille minimale de 46 octets et taille maximale de 1500 octets). il contient les données de la couche 3. Si la taille

des données est inférieure à 46 octets, alors elle devra être complétée avec des octets de bourrage (padding) (voir Figure 5.1).

5.4.6 Champ FCS (Frame Check Sequence)

La trame se termine par un champ FCS [5] qui est un champ de 4 octets contenant une somme de contrôle calculée sur l'ensemble des octets de la trame. Il est utilisé pour détecter d'éventuelles erreurs de transmission.

Calculé par l'émetteur, le FCS permet au récepteur de vérifier la validité des trames reçues. Si le FCS calculé au niveau du récepteur correspond au FCS reçu avec la trame, la trame est considérée comme exempte d'erreurs ; sinon, une erreur est détectée. La détection des erreurs se fait à l'aide du polynôme générateur :

— $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^5 + x^4 + x^2 + 1$.

Remarque :

— Lorsqu'une station reçoit une trame avec un FCS erroné, elle ignore la trame. L'emetteur la retransmettra plus tard.

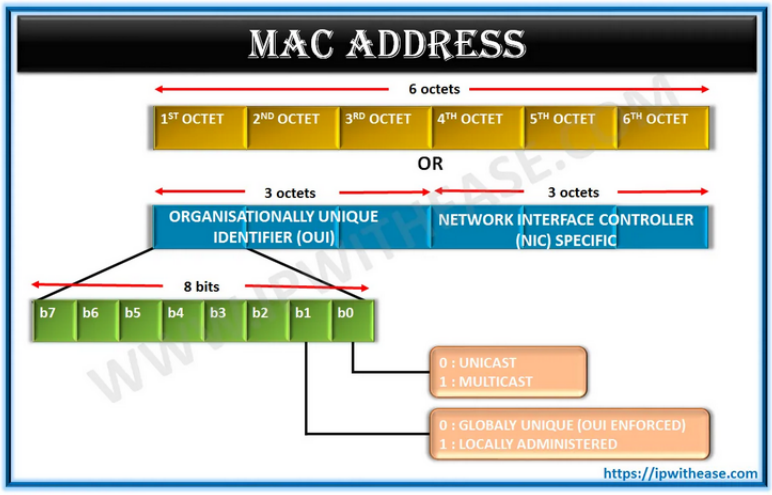


FIGURE 5.4 – Structure d'une adresse MAC [12].

5.5 Rappel sur les adresses MAC

— Les adresses MAC sont aussi appelées adresse physiques ou "BIA" (Burned-in Addresses)

— Chaque périphérique réseau possède un identifiant unique qu'on appelle l'adresse MAC

— 6 octets (en notation hexadécimale), par exemple : 00 :1B :44 :11 :3A :B7

  — Les 3 premiers octets désignent l'identifiant du constructeur. Identiques pour le matériel d'un constructeur. http://standards-oui.ieee.org/oui/oui.txt

5. Frame Check Sequence

— Les 3 derniers octets désignent l'identifiant de la carte. Ils constituent un identifiant unique attribué par le constructeur.

Exemple :

— Considérons une carte réseau avec l'adresse MAC 00 :A0 :C9 :01 :23 :45.

— L'OUI [6]est constitué des trois premiers octets : 00 :A0 :C9. Dans ce cas, il s'agit de **Intel corporation**.

— 01 :23 :45 est l'identifiant de la carte ou (NIC [7]).

— Voici les OUI pour d'autres fabricants bien connus :

  — Dell : 00 :14 :22

  — Nortel : 00 :04 :DC

  — Cisco : 00 :40 :96

  — Belkin : 00 :30 :BD

— Les deux nœuds suivants ont été fabriqués par le même fabricant car ils partagent les trois premiers octets :

  — 02 :E0 :C6 :8B :F2 :54

  — 02 :E0 :C6 :76 :D3 :FE

5.6 Conclusion

Dans ce Chapitre, nous avons exploré le format et les composants essentiels d'une trame Ethernet, pour mieux comprendre le fonctionnement de cette technologie cruciale pour la connectivité des réseaux locaux. En effet, Chaque périphérique sur le réseau est équipé d'une interface réseau avec une adresse MAC unique, permettant ainsi un acheminement efficace des données à travers le LAN. Il est important de noter que chaque type de réseau, qu'il s'agisse d'Ethernet, de HDLC ou de PPP, a son propre format de trame spécifique.

6. Organisationally Unique Identifier
7. Network Interface Controller

Chapitre 6

Réseaux locaux virtuels (VLANs)

6.1 Introduction

Les VLANs (Virtual Local Area Networks) constituent une technologie essentielle dans les réseaux informatiques modernes. Ils permettent de segmenter un réseau physique en plusieurs réseaux logiques indépendants, permettant ainsi une meilleure flexibilité, une meilleure gestion de la bande passante, et des améliorations en termes de sécurité et de performance.

6.2 Qu'est ce qu'un VLAN ?

- VLAN signifie Réseau Local Virtuel (Virtual Local Area Network)
- Un VLAN est créée au niveau du commutateur (switch)
- Un VLAN est une technologie qui permet de créer des réseaux séparés au sein de la même infrastructure de réseau physique, ce qui signifie :
  - **Segmentation Logique** : Les VLANs divisent un réseau physique en plusieurs réseaux logiques. Chaque VLAN est un domaine de diffusion distinct, ce qui signifie que les dispositifs sur le même VLAN peuvent communiquer directement, mais les dispositifs sur des VLANs différents nécessitent un routeur ou un commutateur de couche 3 pour communiquer.
  - **Indépendance de la Localisation Physique** : Les dispositifs et les utilisateurs peuvent être regroupés dans le même VLAN indépendamment de leur localisation physique. Par exemple, les employés travaillant dans différents bâtiments peuvent faire partie du même VLAN, rendant la gestion du réseau plus flexible et efficace.
  - **Sécurité et performance améliorées** : Les VLANs peuvent améliorer la sécurité du réseau en isolant les données et les systèmes sensibles du reste du réseau. Ils peuvent également améliorer les performances en réduisant le trafic de diffusion et en permettant une utilisation plus efficace des ressources réseau.
  - **Gestion simplifiée** : Les VLANs simplifient la gestion du réseau en permettant aux administrateurs de réaffecter facilement les appareils à différents VLANs par le biais

de changements de configuration plutôt que de recâblage physique.

6.3 Création de VLANs

— Un ou plusieurs ports d'un commutateur peuvent être assignés à un VLAN spécifique.

— Les ports assignés à **différents VLANs** ne peuvent communiquer **que via un routeur** (ou un commutateur de couche 3) comme l'illustre la Figure 6.1.

— Par défaut, tous les ports sont dans le VLAN 1, considéré comme le VLAN d'administration.

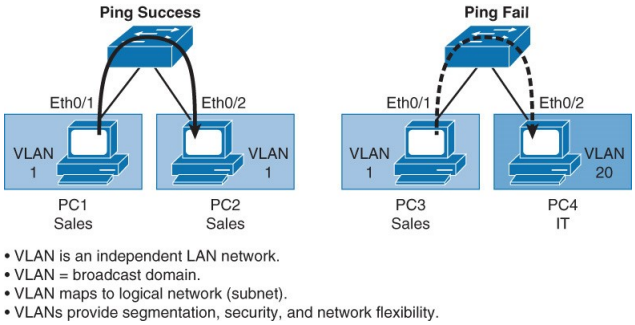


FIGURE 6.1 – Exemple de création de VLAN [13].

6.4 Types de VLANs

— **Data** VLAN (Par exemple, Faculty/Staff, Students, Guest) (voir Figure 6.2).

— **Default** VLAN (VLAN 1) : Créé par le fabricant, il est présent sur le commutateur et tous les ports du commutateur y sont placés.

— **Management** VLAN (IT VLAN) : Utilisé par les administrateurs. C'est le VLAN désigné pour le trafic de gestion, comme les protocoles de gestion à distance des commutateurs (par exemple, SSH, Telnet, SNMP).

— **Native** VLAN (Trames non marquées (Untagged frames) : Les trames circulent à travers le **trunk link** sans être marquées (without being tagged)).

6.5 Configuration d'un VLAN sous packet Tracer

Comment configurer le Data VLAN "Sales" de la Figure 6.2 ?
Switch S1 :

— S1>en

— S1#conf t

— S1(config)#vlan 4

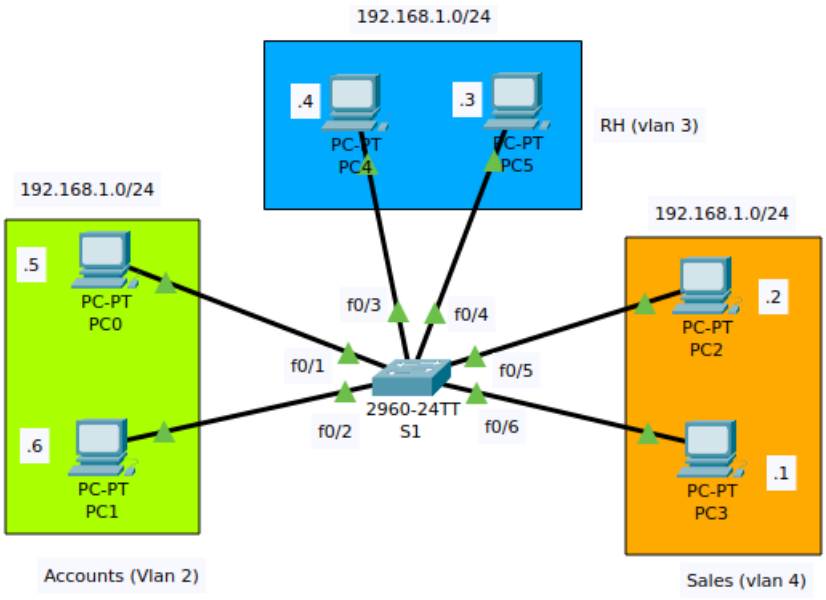


FIGURE 6.2 – **Data** VLANs : Accounts, Sales and RH.

— S4(config-vlan)#name Sales

— S4(config-vlan)#

— S4(config)#interface fa0/5

— S4(config-if)#switchport access vlan 4

— S4(config)#interface fa0/6

— S4(config-if)#switchport access vlan 4

— S4(config-if)#do wr

6.6 VLAN par défaut

La Figure 6.3 montre que tous les ports du commutateur appartiennent initialement au VLAN par défaut, le VLAN 1.

Remarque :

— Le VLAN par défaut ne peut pas être modifié.

— Le VLAN par défaut ne peut pas être supprimé.

6.7 VLAN management

Les Figures 6.4 et 6.5 illustrent respectivement un exemple de VLAN “Management” et sa configuration.

```
Switch#show vlan brief

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
1    default                           active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                 Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                 Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                 Fa0/13, Fa0/14, Fa0/15, Fa0/16
                                                 Fa0/17, Fa0/18, Fa0/19, Fa0/20
                                                 Fa0/21, Fa0/22, Fa0/23, Fa0/24
                                                 Gig0/1, Gig0/2
1002 fddi-default                      active
1003 token-ring-default                active
1004 fddinet-default                   active
1005 trnet-default                     active
Switch#
```

FIGURE 6.3 – **Default** VLAN : VLAN 1.

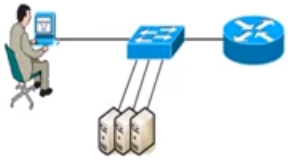


FIGURE 6.4 – **Management** VLAN [5].

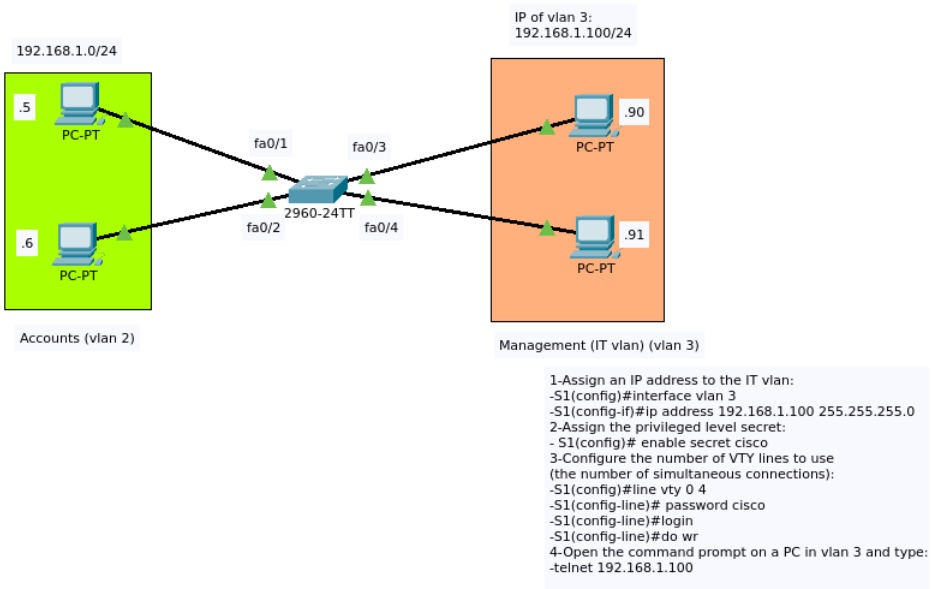


FIGURE 6.5 – Comment configurer un “Management” VLAN.

6.8 Native VLAN (VLAN Natif)

— Un VLAN Natif est un VLAN spécial dont le trafic traverse le tronc 802.1q (the 802.1q trunk) sans étiquette VLAN (trame non marquée, en Anglais untagged frame).

— Un VLAN Natif peut aussi être défini comme étant le VLAN auquel le trafic non marqué est assigné lorsqu'il arrive sur un port de tronc (trunk port).

— Par défaut, il s'agit généralement du VLAN 1, mais cela peut être modifié. Nous pouvons le changer en n'importe quel autre numéro, comme VLAN 2, VLAN 20, VLAN 99 ou tout autre numéro que nous souhaitons.

— Le VLAN Natif et le VLAN de gestion peuvent être identiques, mais **une bonne pratique de sécurité** consiste à les séparer. En effet :

  — Garder le VLAN Natif et le VLAN de gestion séparés améliore la sécurité. Si les deux étaient les mêmes, un attaquant pourrait potentiellement exploiter les vulnérabilités du trafic non étiqueté pour accéder au VLAN de gestion, prenant ainsi le contrôle des appareils réseau.

  — Le VLAN Natif peut être une cible pour les attaques de saut de VLAN, où un attaquant envoie des paquets à double étiquette pour se déplacer entre les VLANs. Séparer le VLAN de gestion atténue le risque que ces attaques affectent le trafic de gestion.

6.9 Scénario de VLAN Natif

1. Supposons que les téléphones IP et les ordinateurs soient connectés aux mêmes ports du commutateur, comme l'illustre la Figure 7.1.
2. Les téléphones IP peuvent étiqueter le trafic vocal avec un ID de VLAN spécifique, tandis que le trafic de données des ordinateurs n'est pas étiqueté :
3. Configuration VLAN :
   — VLAN vocal : VLAN 20
   — VLAN de données : VLAN 10
   — VLAN Natif : VLAN 10 (pour le trafic de données non étiqueté)

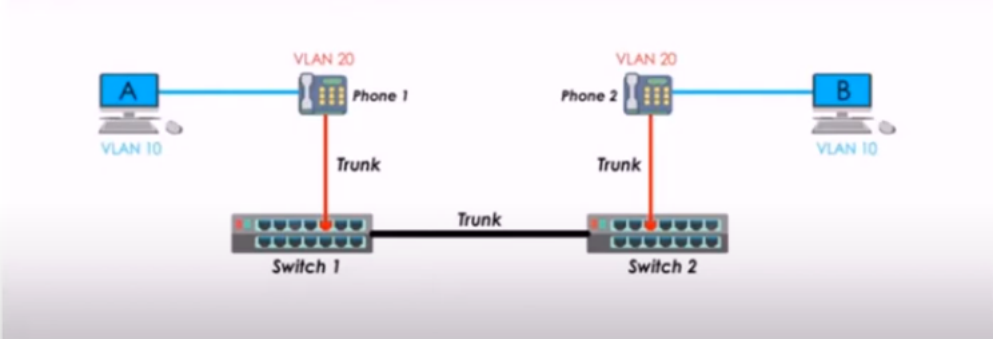


FIGURE 6.6 – VLAN Natif : un scénario [5].

Exemple de configuration :

1. Définir les VLANs :
   — VLAN 10 : Données
   — VLAN 20 : Voix

2. Configurer le port du commutateur :
   — Vous allez configurer le port du commutateur pour gérer à la fois le trafic étiqueté et non étiqueté, en définissant le VLAN natif sur le VLAN de données (VLAN 10) pour le trafic non étiqueté.
3. Configuration sous Packet Tracer :
   — Switch1(config)#interface gigabitEthernet 0/1
   — Switch1(config-if)#switchport mode trunk
   — Switch1(config-if)#switchport trunk native vlan 10
   — Switch1(config-if)#switchport voice vlan 20

Explication des commandes Packet Tracer utilisées :

1. **switchport mode trunk** : Cette commande définit le port en mode trunk, lui permettant de transporter le trafic pour plusieurs VLANs.
2. **switchport trunk native vlan 10** : Cela définit le VLAN 10 comme le VLAN natif. Tout trafic non étiqueté reçu sur ce port sera assigné au VLAN 10. Le VLAN Natif doit être le même aux deux extrémités du trunk. Sinon, le trunk ne fonctionnera pas correctement.
3. **switchport voice vlan 20** : Cela désigne le VLAN 20 comme le VLAN vocal, ce qui signifie que le trafic vocal étiqueté provenant du téléphone IP sera associé au VLAN 20.

Description du flux de trafic :

1. **Trafic de données non étiqueté** : Le trafic provenant d'un ordinateur connecté au téléphone IP ne sera pas étiqueté. Puisque le VLAN 10 est défini comme le VLAN natif, ce trafic de données non étiqueté sera placé dans le VLAN 10.
2. **Trafic vocal étiqueté** : Le trafic provenant du téléphone IP sera étiqueté avec le VLAN 20. La configuration du port du commutateur garantit que ce trafic étiqueté est traité correctement en tant que partie du VLAN 20.
3. La configuration décrite permet à la fois à l'ordinateur et au téléphone IP d'utiliser le même port physique tout en maintenant une séparation logique entre le trafic vocal et le trafic de données à l'aide des VLANs.

6.10 Modes de port du commutateur

1. **Port d'accès** : Connecte les dispositifs finaux et appartient à un seul VLAN. Il transporte un seul VLAN.
2. **Port trunk** : Connecte des commutateurs ou un commutateur à un routeur et transporte plusieurs VLANs (voir Figure 12.2).
3. **Port dynamique** : Négocie automatiquement pour devenir soit un port d'accès, soit un port trunk en fonction du dispositif connecté et de la configuration.

La Figure 6.8 montre comment configurer, sous Packet Tracer, des liens trunk dans une topologie réseau.

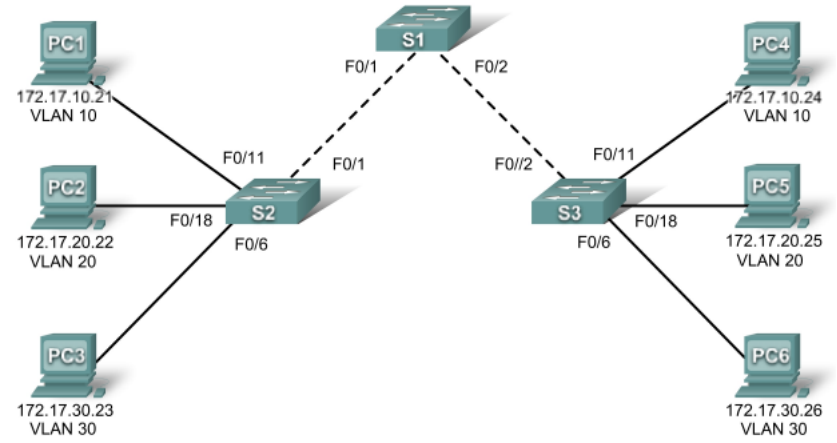


FIGURE 6.7 – Liens Trunk (S1,S2) et (S1,S3) [14].

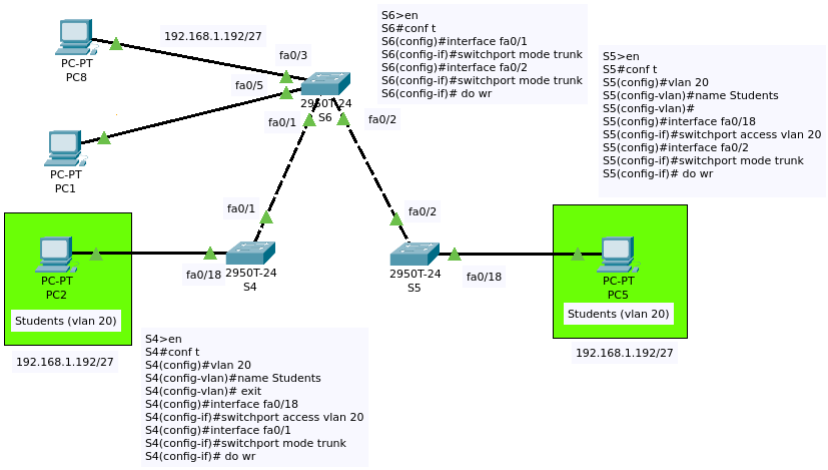


FIGURE 6.8 – Comment configurer les liens Trunk (S4,S6) et (S5,S6).

6.11 Trunking protocols

Le trunk consiste à insérer l'identifiant du VLAN dans une trame utilisateur. Toute trame traversant plusieurs commutateurs (switchs) conservera toujours l'information de son appartenance à son VLAN. Le commutateur (switch) de destination saura avec quels ports la trame peut être commutée (ports appartenant au même VLAN). Dans ce qui suit, nous allons expliquer deux trunking protocols, à savoir :

— IEEE 802.1Q (Standard)

— ISL [1] (Propriétaire, CISCO), **En voie de disparition**

1. Inter-Switch Link

6.11.1 IEEE 802.1Q

— C'est une norme de réseau définie par l'IEEE [2] pour le marquage VLAN sur les réseaux Ethernet.

— Les points clès de la norme IEEE 802.1Q sont :

  — **Norme ouverte** : IEEE 802.1Q est une norme ouverte (an open standard), ce qui signifie qu'elle est publiquement disponible et peut être mise en œuvre par tout fabricant ou organisation. Cela favorise l'interopérabilité entre les appareils de différents fournisseurs.

  — **Objectif** : Spécifie comment les balises VLAN sont insérées dans les trames Ethernet, permettant à plusieurs VLANs de coexister sur le même réseau physique et garantissant que les appareils peuvent identifier et gérer le trafic appartenant à différents VLANs.

  — **Mécanisme de marquage** : Elle ajoute une balise VLAN de 4 octets à l'en-tête de la trame Ethernet, qui comprend un ID de VLAN et d'autres informations de contrôle (voir Figure 6.9). Cela aide les commutateurs et autres appareils réseau à gérer et à router le trafic en fonction de l'appartenance au VLAN.

  — **Largement adopté** : En raison de sa nature ouverte, IEEE 802.1Q est largement adopté et pris en charge par la plupart des fabricants d'équipements réseau, garantissant la compatibilité entre différents appareils et réseaux.

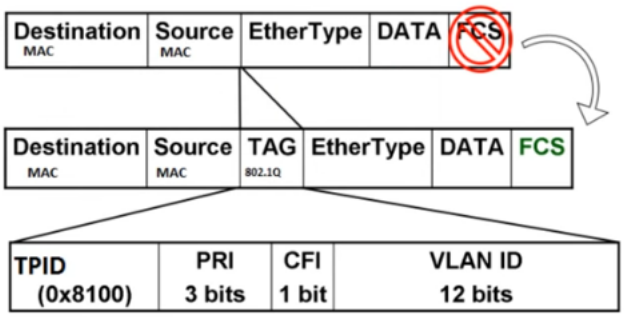


FIGURE 6.9 – Trame étiquetée (Balise VLAN = 4 octets)/Tagged frame (VLAN Tag = 4 bytes) [15].

Les champs de la balise VLAN (**4 octets**) dans une trame IEEE 802.1Q sont :

1. **PRI ou PCP (Priority Code Point)** : **3 bits** utilisés pour la priorité de la Qualité de Service (QoS).
2. **CFI (Canonical Format Indicator) ou DEI** : **1 bit**, maintenant communément appelé DEI (Drop Eligible Indicator).
3. **ID de VLAN** : **12 bits** pour l'identifiant du VLAN.

2. Institute of Electrical and Electronics Engineers

4. **TPID (Tag Protocol Identifier)** : **2 octets**, généralement réglé à 0x8100 pour indiquer la présence d'une balise 802.1Q.

6.11.2 Inter-Switch Link (ISL)

— ISL (Inter-Switch Link) est un autre protocole de marquage VLAN, mais il est propriétaire de Cisco Systems.

— Contrairement à IEEE 802.1Q, qui est une norme ouverte, ISL a été conçu pour les équipements Cisco.

— Les points clés de l'ISL :

  — **Protocole propriétaire** : ISL est un protocole propriétaire de Cisco, ce qui signifie qu'il est spécifiquement conçu pour être utilisé avec les commutateurs Cisco et autres équipements réseau.

  — **Encapsulation** : ISL encapsule une trame Ethernet d'origine en ajoutant un nouvel en-tête et une nouvelle queue de trame (voir Figure 6.10), ce qui inclut des informations sur le VLAN :

    — **Ajout de l'en-tête ISL** : Un en-tête de 26 octets est ajouté au début de la trame Ethernet d'origine. Cet en-tête contient des informations telles que l'identifiant du VLAN (VLAN ID), les indicateurs de priorité, et d'autres informations de contrôle.

    — **Ajout de la queue ISL** : Une queue de 4 octets est ajoutée à la fin de la trame. Cette queue inclut un champ de contrôle de redondance cyclique (CRC) pour assurer l'intégrité des données encapsulées.

Comme mentionné ci-dessus, le **trunk propriétaire Cisco ISL** a la particularité d'encapsuler toute la trame de l'utilisateur dans une nouvelle trame, nommée trame ISL. Nous tenons à rappeler que ce n'est pas le cas du **trunk normalisé 802.1Q** qui lui n'encapsule pas toute la trame de l'utilisateur comme ISL mais casse la trame et y insère une étiquette ou tag, nommée TAG 802.1Q

FIGURE 6.10 – Encapsulation d'une trame ethernet par le trunk propriétaire Cisco ISL

1. **En-tête ISL** : 26 octets, encapsule la trame Ethernet d'origine.
2. **Trame Ethernet d'origine** :
   — Adresse MAC de destination : 6 octets.
   — Adresse MAC source : 6 octets.
   — Ethertype/Longueur : 2 octets.
   — Données/Payload : de 46 à 1500 octets.

— CRC : 4 octets.

3. **Queue ISL** : 4 octets, fournit le CRC pour la vérification d'erreur.

6.11.3 802.1Q vs. ISL

— **Standard** : ISL est propre à Cisco, tandis que 802.1Q est une norme ouverte.

— **Taille de l'en-tête** : ISL ajoute un overhead plus important (30 octets au total) par rapport à la balise de 4 octets de 802.1Q.

— **Compatibilité** : 802.1Q est largement pris en charge par différents fournisseurs, tandis que ISL est limité aux dispositifs Cisco.

— **Le procédé** : Le trunk normalisé 802.1Q n'encapsule pas toute la trame de l'utilisateur comme le fait ISL, mais casse la trame et y insère une étiquette ou tag, nommée TAG 802.1Q.

— **En pratique**, en raison de sa nature propriétaire et de son overhead plus important, ISL a largement été supplanté par 802.1Q, surtout dans les environnements utilisant du matériel provenant de plusieurs fournisseurs. D'ailleurs, les commutateurs Cisco les plus récents peuvent ne prendre en charge que le trunking protocol 802.1Q.

6.12 VLAN Trunking Protocol (VTP)

VTP (VLAN Trunking Protocol) est un protocole propriétaire de Cisco utilisé pour gérer les configurations VLAN sur un réseau de commutateurs. Il simplifie la gestion des VLANs en distribuant et en synchronisant les informations VLANs à tous les commutateurs dans un domaine VTP. Les points clès de ce protocole sont :

— **Domaine VTP** : Un ensemble de commutateurs interconnectés qui partagent des informations VLANs. Tous les commutateurs dans un domaine VTP doivent avoir le même nom de domaine VTP.

— **Modes VTP** [33] :

  — **Mode Serveur** : Dans ce mode, les commutateurs peuvent créer, modifier et supprimer des VLANs pour l'ensemble du domaine VTP. Les modifications apportées à un commutateur en mode serveur sont propagées à tous les autres commutateurs du domaine VTP.

  — **Mode Client** : Les commutateurs en mode client reçoivent les informations VLAN des serveurs VTP mais ne peuvent pas créer, modifier ou supprimer des VLANs. Ils utilisent simplement les informations VLAN distribuées par les serveurs VTP. Lorsqu'un commutateur client VTP reçoit des informations VLAN mises à jour, il les stockent à son niveau, puis il les transmet aux autres commutateurs du réseau (commutateurs qui lui sont connectés). Cela garantit que tous les commutateurs du domaine VTP ont une base de données VLAN cohérente.

  - **Mode Transparent** : Les commutateurs en mode transparent ne participent pas activement à la gestion et à la synchronisation des informations VLAN via le VTP. Ils n'envoient pas d'annonces VTP mais vont transférer les annonces VTP reçues. Ils maintiennent leurs propres configurations VLAN. Notez que les commutateurs en mode transparent VTP ne synchronisent pas leur base de données VLAN avec les informations reçues des annonces VTP (ils ne stockent pas les informations à leur niveau comme le font les commutateurs en mode client avant de les transmettre). Ils se contentent de transmettre les annonces. Notez aussi que les commutateurs en mode Transparent peuvent créer, modifier et supprimer des VLANs locaux (sur eux mêmes seulement).
- **Versions VTP** : VTP a trois versions (1, 2 et 3). La version 3 de VTP offre des améliorations par rapport aux versions 1 et 2, notamment le support des VLANs étendus (4096 VLANs) et de meilleures fonctionnalités de sécurité.
- **Annonces VTP** : VTP utilise des annonces pour partager les informations VLAN. Celles-ci comprennent les annonces récapitulatives, les annonces de sous-ensemble ou partielles (Subset advertisements) et les demandes d'annonces. Il est à noter que :
  - **Annonces récapitulatives** : Font référence aux messages envoyés par les serveurs VTP pour informer les clients VTP de l'état actuel des configurations VLANs dans l'ensemble du domaine VTP.
  - **Annonces de sous-ensemble ou partielles** : Font référence aux messages VTP qui communiquent les changements ou mises à jour VLANs à un sous-ensemble ciblé de commutateurs au sein du réseau, plutôt qu'à tous les commutateurs du domaine VTP.
  - **Demandes d'annonces** : Font référence aux messages ou aux demandes envoyés par les clients VTP aux serveurs VTP. Ces demandes sont effectuées lorsqu'un client VTP a besoin d'informations mises à jour sur les VLANs ou de détails de configuration de la part du serveur VTP.
- **Numéro de révision de configuration** : Chaque domaine VTP a un numéro de révision de configuration qui aide les commutateurs à déterminer s'ils doivent mettre à jour leurs configurations VLAN. Un numéro de révision plus élevé indique une configuration VLAN plus récente.

6.12.1 Exemple simple de configuration VTP

- Switch# conf t
- Switch(config)# vtp domain mondomaine
- Switch(config)# vtp mode server
- Switch(config)# vtp version 2
- Switch(config)# vtp password monmotdepasse

Expliquons brièvement les commandes utilisées :

— **vtp domain mondomaine** : Définit le nom de domaine VTP sur « mondomaine ».

— **vtp mode server** : Configure le commutateur en tant que serveur VTP.

— **vtp version 2** : Configure le commutateur pour utiliser la version 2 de VTP.

— **vtp password monmotdepasse** : Définit un mot de passe VTP pour une sécurité accrue (optionnel).

6.12.2 Exemple d'une topologie comprenant les trois modes VTP

La Figure 6.11 illustre une topologie réseau avec configuration, incluant les 3 modes VTP.

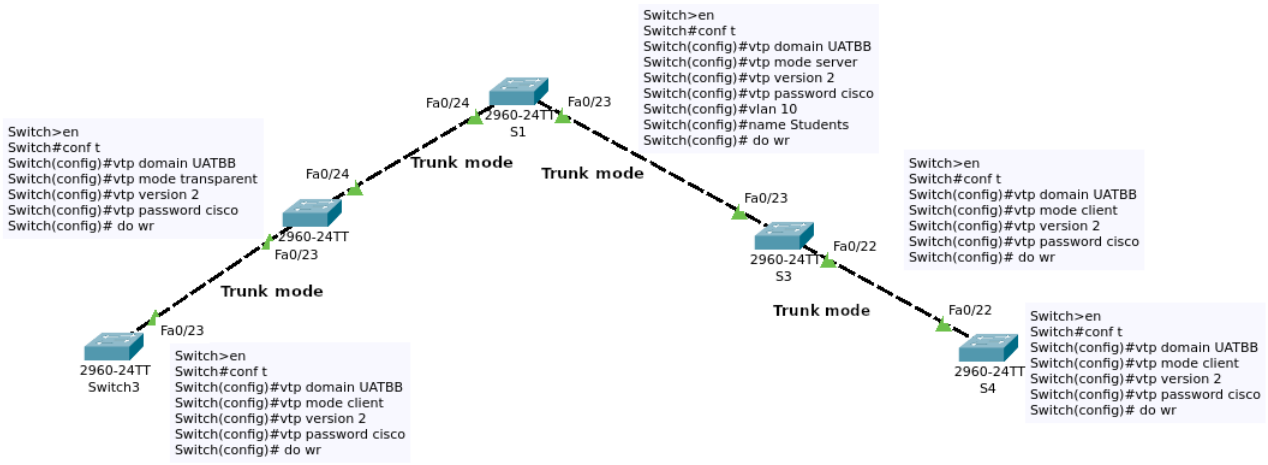


FIGURE 6.11 – Configuration VTP utilisant les trois modes VTP.

6.12.3 Avantages de VTP

— **Gestion centralisée des VLAN** : Simplifie la gestion des VLAN en permettant que les changements de configuration VLAN soient effectués sur un seul commutateur (serveur VTP) et propagés automatiquement à tous les commutateurs dans le domaine.

— **Cohérence** : Assure la cohérence de la configuration VLAN sur tous les commutateurs du réseau.

6.12.4 Désavantages de VTP

— **Risque de mauvaise configuration** : Une mauvaise configuration sur un serveur VTP peut propager des informations VLAN incorrectes à travers le réseau, pouvant potentiellement causer des perturbations réseau.

— **Risques de sécurité** : Un accès non autorisé au domaine VTP peut entraîner des modifications de la configuration VLAN, ce qui peut être atténué en utilisant des mots de passe VTP et les fonctionnalités de sécurité supplémentaires de la version 3.

6.13 Alternatives à VTP

Les alternatives suivantes aident à gérer les configurations de VLAN dans un environnement multi-fournisseurs sans recourir à des protocoles propriétaires comme le VTP de Cisco :

— **Configuration manuelle des VLAN sur chaque commutateur** : Bien que cette méthode assure la compatibilité entre différents fournisseurs, elle peut être fastidieuse (Prenant beaucoup de temps) et sujette aux erreurs, surtout dans les grands réseaux.

Configurer manuellement les VLAN sur chaque commutateur : Bien que cette méthode assure la compatibilité entre différents fournisseurs, elle peut être chronophage et sujette aux erreurs, surtout dans les grands réseaux.

— **Protocole d'arbre de recouvrement multiple (MSTP)** : MSTP [3] (IEEE 802.1s) permet des instances d'arbre de recouvrement basées sur les VLANs, permettant une meilleure distribution du trafic et de la redondance. Il peut fonctionner avec des équipements de différents fournisseurs à condition qu'ils prennent en charge MSTP.

— **Protocole de découverte de la couche de liaison (LLDP)** : LLDP [4] (IEEE 802.1AB) est un protocole de couche de liaison indépendant du fournisseur utilisé pour la découverte des périphériques réseau. Il peut aider à gérer la topologie du réseau et à garantir la cohérence des informations VLAN, bien qu'il ne gère pas directement les configurations VLAN.

— **Logiciel de gestion de réseau** : Utilisez un logiciel de gestion de réseau tiers qui prend en charge les environnements multi-fournisseurs. Exemples : SolarWinds, Nagios et PRTG Network Monitor. Ces outils peuvent automatiser et simplifier les tâches de gestion des VLANs sur du matériel diversifié.

— **Solutions SDN** : Comme OpenFlow, offrent un moyen de gérer de manière centralisée les configurations réseau, y compris les VLANs. Les contrôleurs SDN peuvent gérer les périphériques réseau quel que soit le fabricant, offrant une approche plus flexible et évolutive de la gestion du réseau.

6.14 Routage Inter-VLANs

Le routage inter-VLAN est essentiel pour la communication entre les périphériques de différents VLAN. Cela peut être réalisé à l'aide d'un routeur (**Router-on-a-Stick**) ou d'un **commutateur de couche 3**. Le choix entre ces méthodes dépend de facteurs tels que la taille du réseau, les exigences de performance et les capacités du matériel réseau :

— **Router-on-a-Stick** : Utilise une seule interface physique de routeur configurée avec plusieurs sous-interfaces, chaque sous-interface représentant un VLAN différent. Chaque sous-interface se voit attribuer une adresse IP dans le sous-réseau respectif du VLAN et est configurée pour gérer le trafic étiqueté pour ce VLAN à l'aide de l'encapsulation IEEE

3. Multiple Spanning Tree Protocol
4. Link Layer Discovery Protocol

802.1Q. La Figure 6.12 illustre un exemple de configuration du routage Inter-VLANs à l'aide d'un routeur (**Router-on-a-Stick**).

— **Commutateur de couche 3** : Les commutateurs de couche 3 ont des capacités de routage intégrées, leur permettant d'acheminer directement le trafic entre les VLANs. Cela est réalisé en créant des interfaces VLAN (SVI - Switch Virtual Interfaces) sur le commutateur et en activant le routage IP.

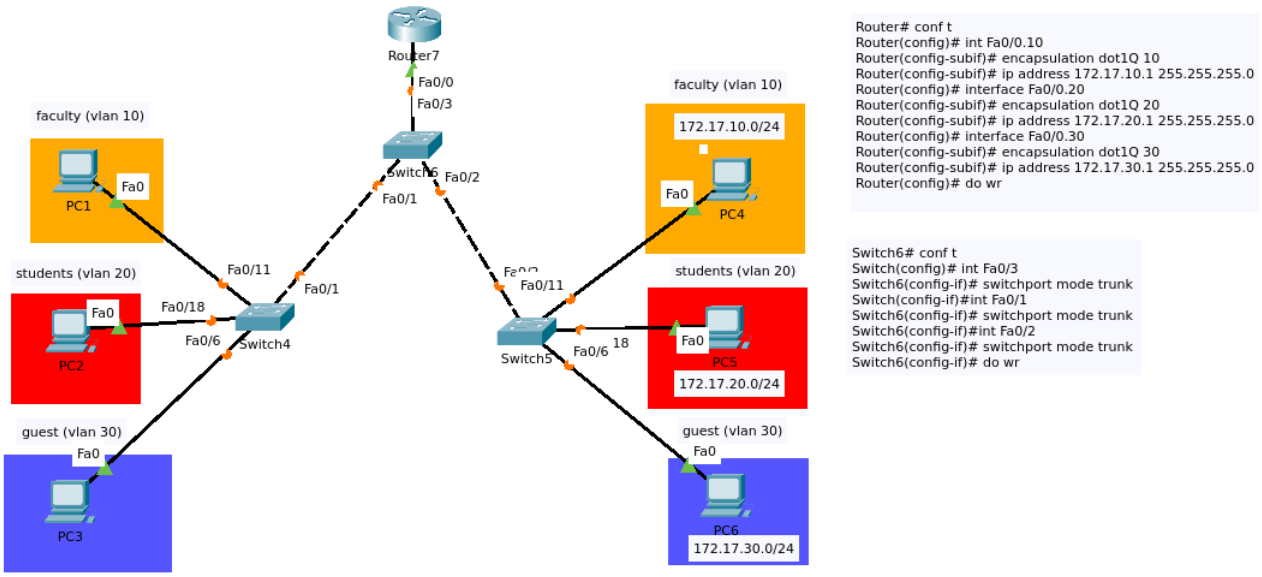


FIGURE 6.12 – Configuration Router-on-a-Stick.

Remarques sur la configuration du Router-on-a-Stick

— Assurez-vous que le port du commutateur connecté au routeur est en mode trunk pour transporter le trafic de plusieurs VLANs.

— Configurez les sous-interfaces sur l'interface physique du routeur, chacune correspondant à un VLAN différent.

6.15 Conclusion

Dans ce chapitre, nous avons exploré les VLANs (Virtual Local Area Networks), une technologie essentielle pour la segmentation logique des réseaux. Les VLANs permettent en effet de créer des sous-réseaux distincts au sein d'un même réseau physique, permettant ainsi une meilleure gestion du trafic et une sécurité renforcée. La maîtrise des VLANs est fondamentale pour les administrateurs réseaux, car elle permet de répondre aux besoins changeants des entreprises en termes de performance et de sécurité.

Chapitre 7

Adressage IPv_6

7.1 Introduction

IPv_6 a été introduit dans les années 1990 pour répondre aux limitations d'adressage de l'IPv_4, notamment en fournissant un espace d'adressage plus vaste pour répondre à la croissance exponentielle d'Internet et pour introduire de nouvelles fonctionnalités. Cependant, l'adressage IPv_4 reste largement utilisée et continue de coexister avec IPv_6 sur le réseau Internet actuel. Les deux protocoles peuvent fonctionner ensemble, permettant une migration progressive des adresses IPv_4 vers IPv_6

7.2 Limites de l'IPv_4

— Spécifcations datant de 1981 (RFC 791). Le protocole IPv_4 a été décrit formellement pour la première fois dans le RFC 791, publié en septembre 1981 par l'Internet Engineering Task Force (IETF).

— Taille de l'espace adressable de plus en plus insuffisante (2^{32}, soit 4.294.967.296) au regard du nombre d'appareils connectés à internet en croissance exponentielle (voir Figure 7.1). Il n'y a plus d'adresses en v_4 (depuis février 2011).

— IPv_4 ne possède pas de mécanismes de sécurité intégrés, ce qui rend les réseaux IPv_4 plus vulnérables aux attaques telles que le détournement d'adresses IP.

— Tailles volumineuses des tables de routage.

7.3 Quelques caractéristiques de l'IPv_6

— IPv_6 est en effet également appelé IPng (IP nouvelle génération ou IP génération suivante).

— En 1998, IPv_6, destiné à terme à remplacer IPv_4, a été adopté comme nouveau standard.

— IPv_6 a été finalisé dans la RFC 2460 en décembre 1998.

— Un espace d'adressage plus grand, passant de 2^{32} à 2^{128}

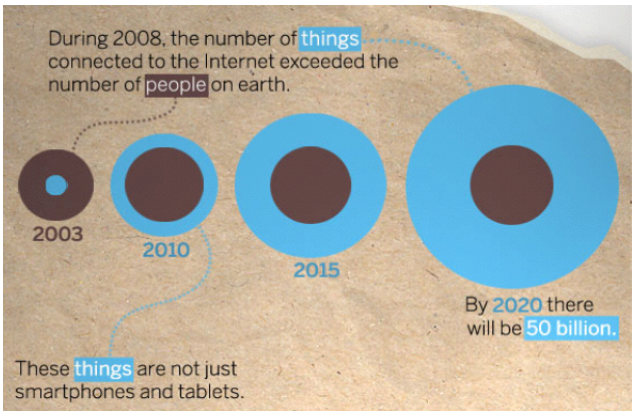


FIGURE 7.1 – Nombre croissant d'appareils connectés à Internet [16]

  - IPv_4 : 4 milliards d'adresses pouvant être assignées.
  - IPv_6 : 340 milliards de milliards d'adresses uniques.
- IPv_6 élimine la nécessité d'utiliser le NAT [1] pour l'adressage IPv_6 global.
- IPv_6 propose des mécanismes d'attribution automatique d'adresses IP, notamment via le protocole NDP (Neighbor Discovery Protocol).
- IPv_6 simplifie en effet le format de l'entête par rapport à IPv_4, bien que l'entête IPv_6 (voir Figure 7.2) soit légèrement plus longue que celle d'IPv_4 en raison de l'introduction de nouveaux champs et de l'adressage étendu (les adresses IPv_6 contiennent 128 bits).
- IPsec est intégré nativement dans IPv_6. Cela signifie que les fonctionnalités de sécurité fournies par IPsec sont intégrées directement dans le protocole IPv_6 lui-même, offrant ainsi des capacités de sécurité améliorées sans nécessiter de protocoles supplémentaires ou de logiciels tiers. Cela permet une sécurisation plus efficace des communications sur les réseaux IPv_6.
- IPv_6 simplifie considérablement la gestion de la table de routage, même avec une infrastructure réseau étendue et complexe. Ceci contribue à une meilleure efficacité du réseau dans son ensemble.
- Support de la mobilité : Dans l'ensemble, IPv_6 offre un meilleur support de la mobilité par rapport à IPv_4.
- Compatibilité avec IPv_4 : Une adresse IPv_6 peut contenir une adresse IPv_4. Les 32 bits de l'adresse IPv_4 sont placés dans les bits de poids faible et un préfixe de 96 bits est ajouté (80 bits mis à 0 suivis de 16 bits mis à 0 ou 1).

7.4 En-tête IPv_6

Dans ce qui suit, nous allons présenter la structure de l'en-tête IPv_6.

1. Network Address Translation

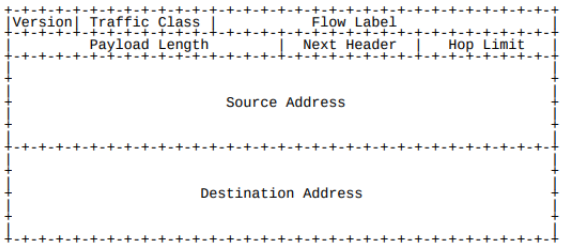


FIGURE 7.2 – En-tête IPv_6 [17]

TABLEAU 7.1 – Détail de l'en-tête IPv_6 [17].

Champ	Description
Version	4-bit (Numéro de version du protocole Internet = 6).
Traffic Class	Champ de classe de trafic sur 8 bits.
Flow Label	Étiquette de flux sur 20 bits.
Payload Length	Entier non signé sur 16 bits. Longueur (en octets)de la charge utile IPv_6, c'est-à-dire le reste du paquet, suivant cet en-tête IPv_6.
Next Header	Sélecteur sur 8 bits. Identifie le type d'en-tête immédiatement après l'en-tête IPv_6. Utilise les mêmes valeurs que le champ Protocole IPv_4 [RFC-1700 et suiv.].
Hop Limit	Entier non signé sur 8 bits. Décrémenté de 1 par chaque nœud qui transmet le paquet. Le paquet est détruit si la limite de saut est decrementée à zéro.
Source Address	Adresse sur 128 bits de l'expéditeur du paquet.
Destination Address	Adresse sur 128 bits du destinataire prévu du paquet (éventuellement pas le destinataire final, si un en-tête de Routage est présent).

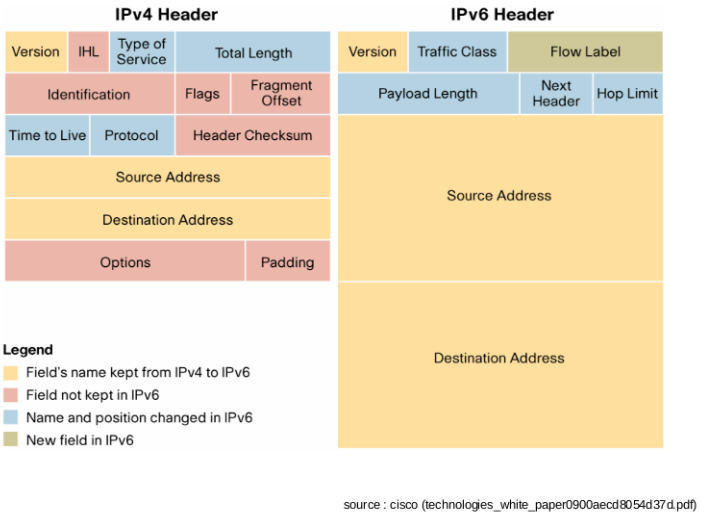


FIGURE 7.3 – En-tête IPv_4 vs. En-tête IPv_6 [17]

7.5 Processus de Mappage d'une Adresse IPv4 vers IPv6

— Une adresse IPv_4 mappée est la transformation d'une adresse IPv_4 en adresse IPv_6. Nous plaçons les 32 bits de l'adresse IPv_4 dans les bits de poids faibles de l'adresse IPv_6, puis nous ajoutons un préfixe de 96 bits. Le préfixe typique est 80 bits à 0 suivis de 16 bits à 1.

  — Ajout de 80 bits nuls (=0)

  — Ajout de 16 bits non nuls (=1)

  — Ajout de l'adresse IPv_4 (32 bits) après le préfixe de 96 bits. Notons, que l'adresse IPv_4 doit d'abord être convertie en binaire, puis on regroupe les chiffres binaires obtenus en paquets de 4 afin de les convertir en chiffres hexadécimaux. A la fin, on sépare les deux segments obtenus par une colonne (:).

Remarque

On dit souvent "adresse IPv_6 mappée sur IPv_4" pour désigner cette représentation d'une adresse IPv_4 dans un format IPv_6. Cependant, l'expression peut varier selon le contexte et les conventions utilisées. On peut également utiliser l'expression "adresse IPv_4 mappée" ou simplement "adresse mappée".

Exemple

Considérons l'adresse IPv_4 suivante : 192.0.2.1. Pour la représenter en tant qu'adresse IPv_6 mappée à IPv_4, nous plaçons les 32 bits de l'adresse IPv_4 (192.0.2.1) dans les bits de poids faible de l'adresse IPv_6, et nous ajoutons un préfixe de 96 bits :

— Préfixe IPv6 mappé à IPv4 : 0000 :0000 :0000 :0000 :0000 :ffff :

— Adresse IPv4 (en binaire) : 11000000.00000000.00000010.00000001

— Adresse IPv4 (en hexadécimal) : C000 :0201

— Adresse IPv6 mappée à IPv4 : 0000 :0000 :0000 :0000 :0000 :ffff :C000 :0201

7.6 Format des adresses IPv_6

— Les adresses IPv_6 sont donc de 128 bits (16 octets).

— Une nouvelle structure de paquet qui a des **adresses IP source et destination de 128 bits**, qui sont **quatre fois** plus grandes que les adresses IP actuelles IPv_4.

— Les adresses IPv_6 sont divisées en **8 groupes de 16 bits** représentés par 4 chiffres **hexadécimaux** et séparés par " :".

— Les lettres du système hexadécimal peuvent être en **majuscules** ou en **minuscules**.

— En IPv_6, les masques sont exprimés en notation CIDR qui correspond à un slash (/) à la fin de l'adresse suivi de la longueur du préfixe en bits. Par exemple, 2001 :0db8 :85a3 :0000 :0000 :8a2e :0370 :7334/ indique que les 64 premiers bits représentent la partie réseau, et les bits restants sont pour adresser les hôtes dans ce réseau.

Exemple

TABLEAU 7.2 – IPv_4 vs. IPv_6

IPv_4	IPv_6
32 bits = 4 octets	128 bits = 16 octets = 8 segments (groupes)
Décimal ↔ binaire	Hexadecimal ↔ Binaire
Notation décimale pointée	Notation en colonne
10.1.1.1 172.16.1.1 192.168.1.1	2035 :0001 :2BC5 :0000 :0000 :087C :0000 :000A : : : :1 FE80 : : FEC0 : : FD00 : : 2000 : : FF00 : :
Classful, Classless	Classless
Subnetmask, Prefix (CIDR)	Prefix (CIDR)
Classes A, B, C, D, E	No classes
Unicast, Multicast, Broadcast	Unicast, Multicast, Anycast

— 5800 :10C3 :E3C3 :F1AA :48E3 :0923 :D494 :AAFF

— 2001 :0660 :7401 :0200 :0000 :0000 :0edf :bdd7

Remarque Tel qu'illustré dans le Tableau 7.2, IPv_6 n'a pas d'adresses de diffusion traditionnelles ; à la place, il utilise des adresses de multidiffusion pour une communication efficace de un à plusieurs, améliorant les performances du réseau et éliminant le besoin de diffusions comme c'était le cas dans l'adressage IPv_4.

7.7 Comment compresser une adresse IPv_6 ?

Pour compresser une adresse IPv_6, vous pouvez suivre ces règles :

— Omettez les zéros initiaux dans chaque groupe de chiffres hexadécimaux.

— Si un segment dans l'adresse IPv_6 contient quatre zéros consécutifs, il peut être compressé en un seul zéro.

— Remplacez les groupes de zéros consécutifs par un double deux-points (: :) **une seule fois** dans l'adresse.

— Si plusieurs groupes de zéros peuvent être compressés, compressez la séquence la plus longue.

Exemples

1. Exemple 1 :
   — Adresse IPv_6 d'origine : 2001 :0db8 :0000 :0042 :0000 :8a2e :0370 :7334
   — Adresse IPv_6 compressée : 2001 :db8 :0 :42 : :8a2e :370 :7334
   L'adresse IPv_6 obtenue est sous une forme plus concise.
2. Exemple 2 :
   — Adresse IPv_6 d'origine : 2001 :0660 :7401 :0200 :0000 :0000 :0edf :bdd7
   — Adresse IPv_6 compressée : 2001 :660 :7401 :200 : :edf :bdd7

TABLEAU 7.3 – Compression des adresses IPv_6

Avant	Après
2001 :0000 :0000 :0002 :0001 :0000 :0000 :ABCD	2001 : :2 :1 :0 :0 :ABCD 2001 :0 :0 :2 :1 : :ABCD
FF01 :0 :0 :0 :0 :0 :0 :1	FF01 : :1
0 :0 :0 :0 :0 :0 :0 :1	: :1
0 :0 :0 :0 :0 :0 :0 :0	: :
2001 :0 :aabb :2 : :1	2001 :0 :aabb :2 : :1

— Adresse IPv_6 d'origine : 2001 :0db8 :3c4d :0015 :0000 :d234 :3eee :0000

— Adresse IPv_6 compressée : 2001 :db8 :3c4d :15 :0 :d234 :3eee :0

3. Exemple 3 :

— Adresse IPv_6 d'origine : 2030 :0000 :130F :0000 :0000 :09C0 :876A :130B

— Adresse IPv_6 compressée :

— 2030 :0 :130F :0 :0 :9C0 :876A :130B → ✓

— 2030 :0 :130F :0 :0 :9C :876A :130B → ✗

— 2030 : :130F : :9C0 :876A :130B → ✗

— 2030 :0 :130F : :9C0 :876A :130B → ✗

4. Exemple 4 :

— Adresse IPv_6 d'origine : 1080 :0 :0 :0 :8 :800 :200C :417A

— Adresse IPv_6 compressée : 1080 : :8 :800 :200C :417A

— Adresse IPv_6 d'origine : FF01 :0 :0 :0 :0 :0 :0 :101

— Adresse IPv_6 compressée : FF01 : :101

5. Exemple 5 :

— Adresse IPv_6 d'origine → 0 :0 :0 :0 :0 :0 :0 :1

— Adresse IPv_6 compressée → : :1

— Adresse IPv_6 d'origine → 0 :0 :0 :0 :0 :0 :0 :0

— Adresse IPv_6 compressée → : :

7.8 Adresses IPv_6 spéciales

— (: :/0) : Route par défaut

— (: :1/128) : Adresse de bouclage (loopback)

— http ://[2001 :db8 :1234 :5678 : :CAFE] :8080, où 8080 représente le numéro de port

7.9 Types d'adresses IPv_6

Dans cette Section, nous allons aborder les types des addresses IPv_6, à savoir :

— Unicast

— Multicast

— Anycast

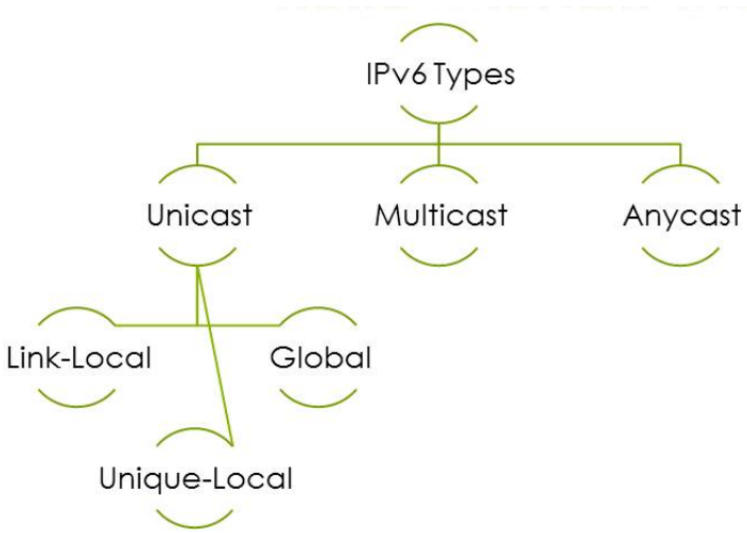


FIGURE 7.4 – Types d'adresses IPv_6 [18]

Le types Unicast comprend 3 types :

— **Link-Local** Unicast address, **eq. APIPA** [2] **in IP**v_4

— **Unique-Local** Unicast address (called before Site-Local), **eq. Private in IP**v_4

— **Global** Unicast address, **eq. Public in IP**v_4

Rappel 1 :

— **Unicast** : Un seul expéditeur et un seul destinataire (1 :1)

— **Multicast** : Un seul expéditeur et plusieurs destinataires (1 :n)

— **Broadcast** : Un seul expéditeur et tous les destinataires connectés à un réseau

— **Anycast** : Un seul expéditeur et le destinataire le plus proche parmi un groupe de destinataires

— **Concast** : Plusieurs expéditeurs et un seul destinataire (m :1)

— **Multipeer** : Plusieurs expéditeurs et plusieurs destinataires (m :n)

— **Geocast** : Un seul expéditeur et plusieurs destinataires dans une certaine région géographique

2. Automatic Private IP Addressing

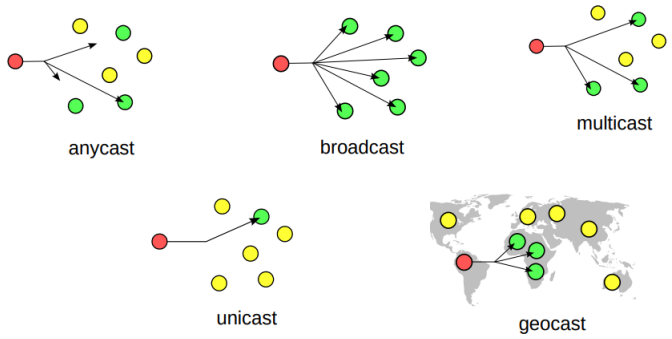


FIGURE 7.5 – Communication modes [17]

Rappel 2 :

— **Adresses unicast** : Identifie un nœud unique, et le trafic destiné à une adresse unicast est transféré à un seul nœud.

— **Adresses multicast** : Identifie un groupe de nœuds, et le trafic destiné à une adresse multicast est transféré à tous les nœud du groupe.

— **Adresses anycast** : Identifie un groupe de nœuds, et le trafic destiné à une adresse anycast est transféré au nœud "le plus proche" dans le groupe.

7.9.1 Structure des adresses Unicast Link-Local

— Similaire à **APIPA** en IPv_4, les adresses de liaison locale (link-local) en IPv_6 servent à l'attribution automatique d'adresses lorsqu'aucune autre configuration n'est disponible.

— Elles commencent par le motif binaire **1111 1110 10**, indiquant les 10 premiers bits de l'adresse.

— Le préfixe correct pour les adresses de liaison locale en IPv_6 est **FE80 : :/10**, désignant la plage d'adresses disponible pour la communication locale.

— Sont utilisées **dans le cadre du réseau local (LAN)**, facilitant la communication entre les appareils sur le même lien ou segment de réseau.

— Contrairement à l'APIPA en IPv_4, les adresses de liaison locale en IPv_6 ne sont **pas uniquement réservées aux cas de crise**.

Rappel sur les adresses APIPA :

— L'APIPA est un processus qui permet à un système d'exploitation de s'assigner automatiquement une adresse 169.254.$x.y$/16 lorsque le serveur DHCP [3] est hors service ou inaccessible.

— L'APIPA est sur le réseau 169.254.0.0/16, c'est-à-dire la plage : 169.254.0.1 → 169.254.255.254

3. Dynamic Host Configuration Protocol

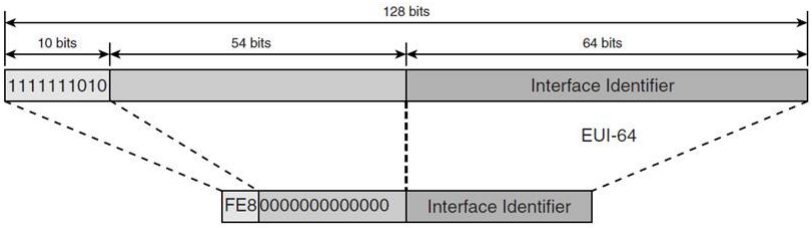


FIGURE 7.6 – Structure des addresses Unicast Link-Local [18]

— Il s'agit d'un réseau privé qui n'est pas routable sur Internet ou ailleurs. Lorsqu'une station possède une adresse APIPA, elle ne peut communiquer qu'avec d'autres machines configurées en APIPA (communication sur le même commutateur).

— Afin que deux machines ne s'attribuent pas la même adresse APIPA, une fois qu'une machine s'est attribuée une adresse APIPA (en choisissant deux chiffres aléatoires x et y), elle envoie une requête ARP [4] aux autres machines pour leur demander si elles n'ont pas la même adresse. Si c'est le cas, elle choisit deux autres nombres aléatoires x et y et s'assure qu'ils ne sont pas déjà pris (requête ARP), etc.

— L'APIPA est utilisé uniquement en cas de problème.

Remarques concernant les adresses APIPA

— La plupart du temps, les cartes réseau sont configurées pour envoyer des requêtes à leur assemblage afin d'établir une communication avec un serveur DHCP. Cependant, lorsque la carte ne parvient pas à obtenir de réponse à ses requêtes, elle choisit alors de se configurer avec une adresse APIPA.

— Le service APIPA continue de vérifier la présence d'un serveur DHCP dans la zone toutes les 5 minutes. Il réémet une demande de type DHCP Discover en espérant obtenir une réponse d'un serveur. S'il obtient une réponse, le service DHCP attribuera une adresse IP à la carte réseau et remplacera l'adresse IP APIPA en place.

— Pour désactiver/activer APIPA sur Debian :

1. Modifiez le fichier /etc/default/avahi-daemon en définissant AVAHI_DAEMON_DETECT_LOCAL à 0
2. Redémarrez le démon avahi via la commande /etc/init.d/avahi-daemon restart

7.9.2 Structure des adresses Unicast Site-Local

— Similaire aux **adresses IPv_4 privées**

— Destinées à être utilisées au sein d'un **réseau interne d'une organisation** et **non routables** sur l'internet global.

4. Address Resolution Protocol

— Commencent avec le motif binaire **1111 1110 11** pour identifier la plage d'adresses réservée à l'adressage site-local.

— Utilisent le préfixe **FEC0 : :/10**

— **Peuvent être répétées**

— Sont conçues pour être utilisées au **niveau de l'organisation**, facilitant la communication au sein de l'infrastructure réseau de l'organisation.

— Les adresses site-local sont **maintenant** considérées comme **obsolètes**.

7.9.3 Structure des adresses Unicast Unique-Local

— Similaire aux **adresses privées** en IPv_4, elles fournissent un moyen d'attribuer des adresses au sein d'une organisation sans routage global.

— Commencent par le motif binaire **1111 110** ou **1111 1101**, indiquant respectivement les 7 premiers et 8 premiers bits de l'adresse.

— Utilisent le préfixe **FC00 : :/7** et **FD00 : :/8**.

— **Ne devraient pas être répétées** entre différentes organisations pour éviter les conflits potentiels. Elles sont destinées à être utilisées au sein du réseau interne d'une seule organisation.

— **Ne sont pas routables sur l'internet global et fonctionnent au niveau de l'organisation**, facilitant la communication au sein de l'infrastructure réseau de l'organisation.

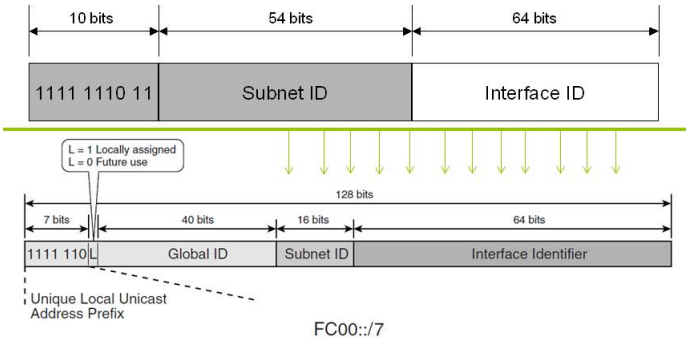


FIGURE 7.7 – Structures des adresses Unicast Site/Unique-Local [18]

7.9.4 Structure des adresses Unicast Global

— Similaires aux adresses IPv_4 publiques, les adresses unicast global dans IPv_6 sont attribuées par l'IANA [5] et sont utilisées pour la communication Internet globale.

— Commencent avec le motif binaire **001**, indiquant les trois premiers bits de l'adresse.

— Commencent avec le préfixe **2000 : :/3**, indiquant leur portée globale.

5. Internet Assigned Numbers Authority

— Le préfixe **2000 : :/3** englobe toutes les adresses IPv_6 qui commencent par le motif binaire **001**, les identifiant comme des adresses unicast global.

— Les adresses unicast global fonctionnent au niveau de l'internet, permettant la communication entre les appareils à travers différents réseaux sur l'internet mondial.

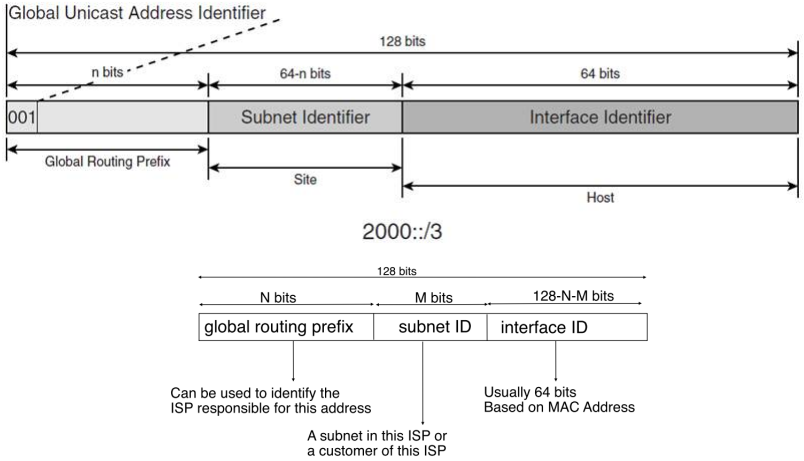


FIGURE 7.8 – Structure des adresses Unicast Global [18]

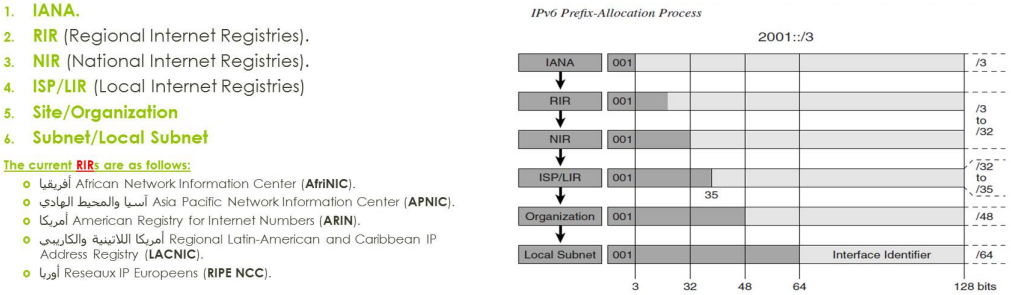


FIGURE 7.9 – Processus d'attribution de préfixe IPv6 [18]

7.10 EUI-64

— EUI-64 pour "Extended Unique Identifier" est une méthode pour former des adresses IPv_6 en utilisant l'adresse MAC [EUI-48] de la carte réseau qu'elle utilise. Concrètement, cela permet à un hôte de s'attribuer automatiquement une adresse IPv_6 unique.

— Les 64 derniers bits d'une adresse IPv_6 sont dérivés de l'adresse MAC de l'interface.

Pour passer d'une adresse MAC de 48 bits à un identifiant d'interface IPv_6 de 64 bits :

a) Les 24 premiers bits : numéro de constructeur avec le 7^{me} bit inversé.

b) Les 16 bits suivants ont la valeur FFFE.

c) Les 24 derniers bits : numéro de série de l'interface.

La valeur obtenue est un EUI-64.

FIGURE 7.10 – Comment transformer l'adresse MAC lors de l'utilisation de l'EUI-64 [18]

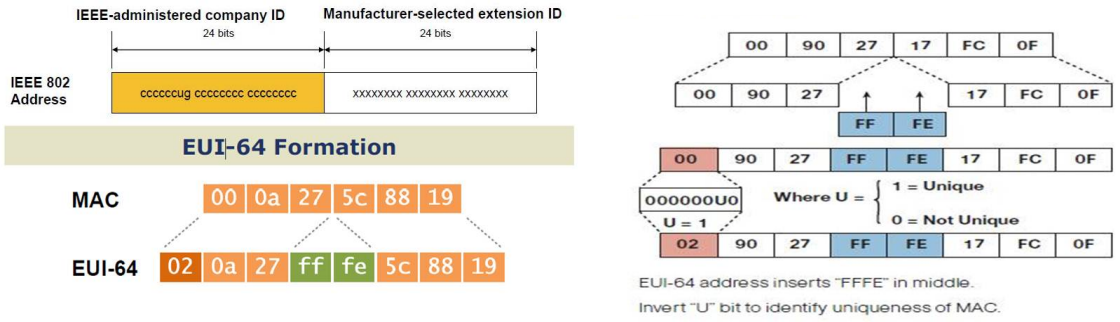
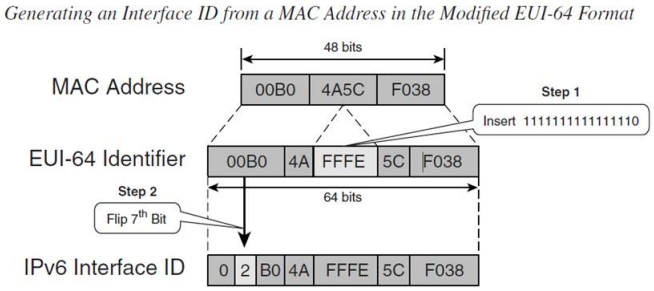


FIGURE 7.11 – Exemple [18]



7.11 SLAAC

Comme pour l'IPv_4, les adresses IPv_6 de type unicast global peuvent être configurées de deux manières :

— Manuellement

— Dynamiquement

Les méthodes d'attribution dynamiques des adresses en IPv_6 sont les suivantes :

— Stateless Address Autoconfiguration (SLAAC)

— Dynamic Host Configuration Protocol version 6 (DHCPv6)

Quant aux méthodes d'attribution des adresses en IPv_4, on a :

— DHCP (Dynamic Host Configuration Protocol)

— Autoconfiguration (APIPA - Automatic Private IP Addressing)

Remarque Importante :

Nous tenons à préciser qu'il existe deux types de DHCP (Dynamic Host Configuration Protocol) en IPv_6 :

— DHCPv_6 Stateless : Dans ce mode, le DHCPv_6 est utilisé pour fournir des paramètres de configuration supplémentaires aux hôtes IPv_6, tels que les adresses des serveurs DNS, sans leur attribuer d'adresses IPv_6. Les hôtes utilisent toujours l'autoconfiguration d'adresse sans état (SLAAC) pour obtenir leurs adresses IPv_6.

— DHCPv6 Stateful : Ce mode de DHCPv_6 attribue des adresses IPv_6 aux hôtes et fournit également des paramètres de configuration supplémentaires, similaires au DHCP traditionnel en IPv_4. Il est appelé "stateful" parce qu'il maintient des informations d'état sur les adresses IP attribuées.

— Les deux types de DHCPv_6, stateful et stateless, peuvent coexister dans le même réseau, et les hôtes peuvent utiliser l'une ou l'autre méthode, voire les deux simultanément, pour obtenir leur configuration IPv_6.

Dans ce qui suit, nous allons aborder uniquement la méthode SLAAC

— SLAAC [6] est une méthode par laquelle un appareil peut obtenir une adresse IPv_6 unicast global sans recourir aux services d'un serveur DHCPv_6.

— Le protocole ICMPv_6 [7] est au coeur du processus SLAAC.

— SLAAC utilise "ICMPv_6 Router Advertisement and Solicitation messages" pour fournir des informations d'adressage et d'autres configurations.

— La méthode **SLAAC** est donc utilisée pour l'autoconfiguration des adresses unicast global.

— **SLAAC** permet à un hôte de configurer automatiquement son adresse globale à partir de l'annonce d'un préfixe donné par un routeur :

  — Les premiers 64 bits sont donc donnés par le routeur.

  — Les derniers 64 bits sont pris à partir de l'adresse MAC (EUI-64) ou générés de manière aléatoire.

Remarque :

Les routeurs Cisco sont configurés en SLAAC **par défaut**.

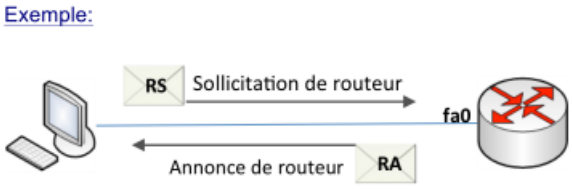


FIGURE 7.12 – Autoconfiguration IPv_6 [19]

6. Stateless Address Autoconfiguration
7. Internet Control Message Protocol for IPv_6

7.12 Adresses IPv_6 Multicast

— Servent à envoyer des paquets de données à **plusieurs destinataires simultanément**.

— Commencent par le motif binaire "**11111111**" dans le premier octet.

— Sont assignées à un **groupe spécifique de nœuds** qui se sont enregistrés pour recevoir les paquets envoyés à cette adresse.

— Ont une **portée** associée, indiquant l'étendue du réseau sur lequel les paquets multicast seront distribués.

— Peuvent être utilisées à diverses fins, telles que les **protocoles de routage**, la **découverte de services**, et le **streaming multimédia**.

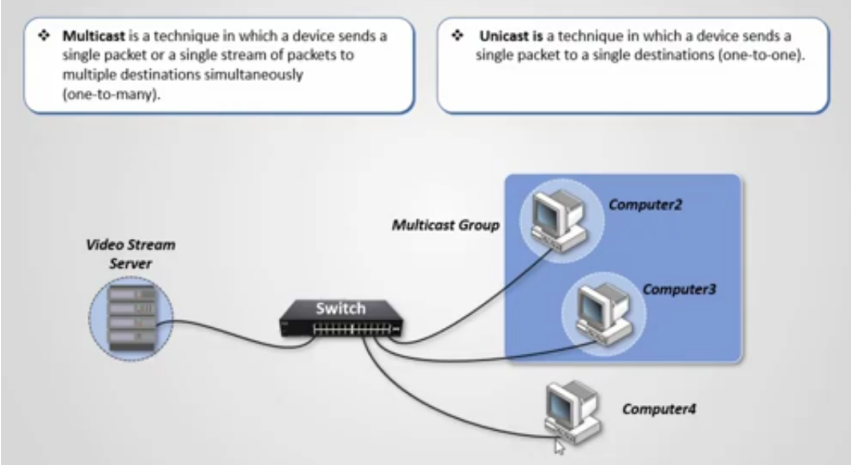


FIGURE 7.13 – Groupe Multicast [6]

Remarque importante :

Un paquet envoyé à un **groupe multicast** a toujours une **adresse source unicast**. Une adresse multicast ne peut être qu'une adresse de destination et ne peut jamais être une adresse source.

Les préfixes couramment utilisés pour les adresses IP multicast en IPv_6 sont :

— **FF00 : :/8** : C'est le préfixe des adresses multicast IPv_6. Toutes les adresses multicast commencent par le motif binaire "11111111" dans le premier octet, ce qui correspond au préfixe FF00 : :/8. Elles sont l'équivalent en IPv_4 de **224.0.0.0/4**.

— **FF02 : :/16** : Il s'agit du préfixe d'adresses multicast de portée lien-local. Les adresses multicast dans ce préfixe sont utilisées pour la communication au sein du même lien ou segment de réseau.

— **FF05 : :/16** :Il s'agit du préfixe d'adresses multicast de portée locale au site. Les adresses de multidiffusion au sein de ce préfixe sont utilisées pour la communication au sein du même site ou de la même organisation.

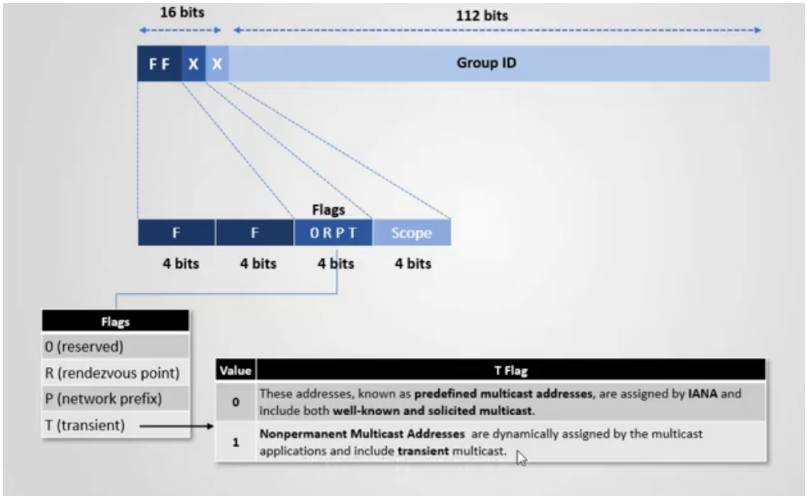


FIGURE 7.14 – Format des adresses IPv_6 multicast [6]

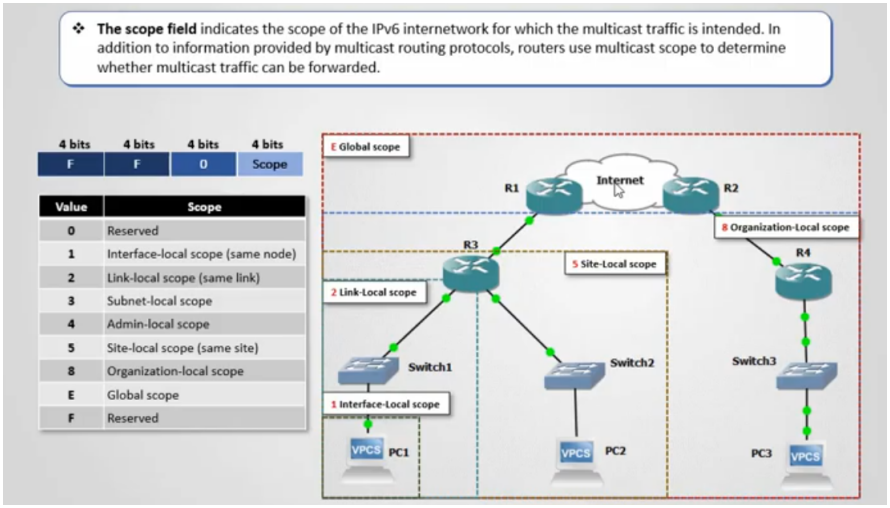


FIGURE 7.15 – Portée des adresses IPv_6 multicast [6]

Ces préfixes aident à organiser et catégoriser les adresses multicast en fonction de leur portée et de leur utilisation prévue au sein du réseau IPv_6.

Remarque : Le champ **flags** fait partie du format des adresses multicast IPv_6, mais il n'est souvent pas explicitement affiché. Il comprend des bits réservés pour une utilisation future et qui sont généralement définis à zéro.

7.13 Conclusion

Dans ce Chapitre, nous avons passé en revue les concepts de base d'IPv_6. Ce dernier a été conçu pour surmonter les limitations d'IPv_4 et permettre ainsi la croissance continue d'Internet. Avec son espace d'adressage étendu, son efficacité améliorée et ses fonctionnalités de sécurité

renforcées, IPv_6 offre une base solide pour l'avenir des réseaux. En adoptant IPv_6, les organisations peuvent garantir que leurs réseaux restent évolutifs, efficaces et sécurisés dans le paysage numérique en constante évolution. Enfin, il est à noter que la transition d'IPv_4 à IPv_6 se poursuit malgé sa lenteur.

Chapitre 8

Protocoles de transport TCP/UDP

8.1 Introduction

Les protocoles de transport jouent un rôle crucial dans la communication sur Internet en fournissant les mécanismes nécessaires pour permettre l'échange des données entre applications sur différents ordinateurs. Parmi les protocoles de transport les plus couramment utilisés, on citera les protocoles TCP [1] et UDP [2]. Chacun de ces protocoles, de par ses caractéristiques spécifiques, est adapté à certain types d'applications. Dans ce qui suit, nous allons expolorer en profondeur chacun de ces deux protocoles.

8.2 Transmission Control Protocol (TCP)

Le protocole TCP est un protocole de transport orienté connexion, ce qui signifie qu'il établit une connexion fiable entre deux hôtes avant de transmettre des données. TCP est conçu pour fournir une communication fiable, ordonnée et sans erreurs.

Le TPDU (Transport Protocol Data Unit) est utilisé par les protocoles de transport comme TCP et UDP pour encapsuler et transmettre les données entre les hôtes dans un réseau. Chaque protocole a son propre format de TPDU : TCP utilise des segments pour assurer une transmission fiable et ordonnée, tandis que UDP utilise des datagrammes pour une transmission rapide mais sans garantie de livraison.

Un TPDU contient généralement :

— **Un en-tête :** Informations de contrôle (comme les numéros de séquence, les accusés de réception, les indicateurs de contrôle, etc.).

— **Les données utiles (Payload) :** Ce sont les données utilisateur encapsulées dans le segment TCP après l'en-tête.

Nous tenons à rappeler qu'outre le TPDU de la couche transport, il existe d'autres unités de données de base utilisées dans les autres couches du modèle OSI :

1. Transmission Control Protocol
2. User Datagram Protocol

— **Couche application :** PDU (Protocol Data Unit), l'unité de données propre au protocole applicatif, comme un email, un fichier HTTP, une requête SQL, etc.

— **Couche présentation :** PPDU (Presentation Protocol Data Unit), l'unité de données de la couche de présentation, qui gère la syntaxe et la sémantique des données.

— **Couche session :** SPDU (Session Protocol Data Unit), l'unité de données de la couche de session, qui gère l'établissement, la synchronisation et la terminaison des sessions de communication.

— **Couche réseau :** NPDU (Network Protocol Data Unit) ou paquet, l'unité de données de la couche réseau, comme un paquet IP, qui gère l'acheminement des données entre réseaux.

— **Couche liaison de données :**

  — LPDU (Logical Link Control Protocol Data Unit), l'unité de données de la couche de liaison logique, qui gère la fiabilité de la transmission des trames sur le lien physique.

  — MPDU (Media Access Control Protocol Data Unit) ou trame - L'unité de données de la couche d'accès au média, qui gère l'accès physique au support de transmission.

— **Couche physique :** Bit, l'unité de base de la couche physique, qui représente un '0' ou un '1' binaire.

Chaque couche du modèle OSI a donc son unité de données spécifique, adaptée aux fonctions qu'elle remplit dans l'architecture réseau. Ces unités s'emboîtent les unes dans les autres lors de la transmission des données entre systèmes.

8.2.1 Principales Caractéristiques du protocole TCP

Le protocole TCP présente plusieurs caractéristiques essentielles qui le rendent fiable et adapté aux communications réseau nécessitant une transmission de données fiable et ordonnée, comme par exemple la navigation web, le transfert de fichiers et les courriels. Les principales caractéristiques de ce protocole sont (classées par concept) :

1. **Connexion orientée :** Avant d'échanger des données, TCP établit une connexion entre l'emetteur et le récepteur via un processus appelé "three-way handshake". Cette connexion est maintenue pendant toute la durée de la communication et est arrêtée à la fin.
2. **Fiabilité :** TCP assure la livraison fiable des segments de données en utilisant des accusés de réception (ACK) et des mécanismes de retransmission pour les segments perdus ou corrompus.
3. **Ordonnancement des données :** TCP garantit que les segments de données sont reçus dans le bon ordre, même s'ils sont transmis dans un ordre différent. Par exemple, les segments envoyés peuvent emprûnter des chemins différents et par conséquent arriver dans le désordre (voir Figure 8.1). Les segments sont numérotés et réassemblés dans l'ordre correct par le récepteur en utilisant les numeros de séquence. TCP maintient ainsi l'intégrité et l'ordre des données, ce qui assure une communication fiable entre les applications réseau.
4. **Intégrité des données :** TCP utilise une somme de contrôle (checksum) pour vérifier l'intégrité des données transmises. Si un segment est corrompu, il est rejeté et retransmis.

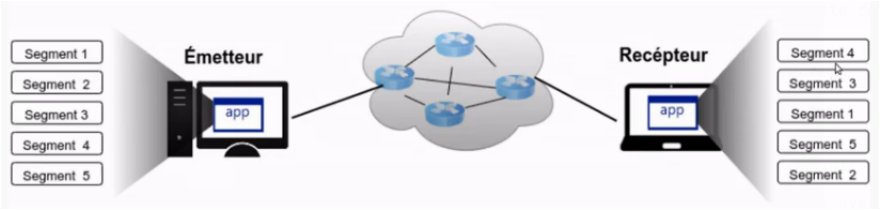


FIGURE 8.1 – Arrivée désordonnée des segments TCP [20]

5. **Contrôle de flux :** TCP utilise des fenêtres d'émission et des mécanismes comme le glissement de fenêtre (sliding window) pour réguler la quantité de données envoyées, évitant ainsi de saturer le récepteur.
6. **Contrôle de congestion :** TCP intègre des algorithmes pour détecter et éviter la congestion du réseau, tels que TCP Tahoe et TCP Reno. Ces algorithmes ajustent dynamiquement la vitesse de transmission en fonction des conditions du réseau.
7. **Multiplexage :** TCP permet à plusieurs applications d'utiliser simultanément le réseau en utilisant des ports pour différencier les flux de données. Chaque connexion TCP est identifiée par une paire de ports source et destination.
8. **Communication bidirectionnelle :** TCP permet des communications bidirectionnelles où les deux extrémités peuvent envoyer et recevoir des données simultanément. C'est ce qu'on appelle le mode de communication full duplex.
9. **Communication point à point :** Une connexion TCP est établie entre deux points spécifiques (deux hôtes), ce qui signifie qu'il y a une relation directe et exclusive entre le client et le serveur. Cette connexion dédiée assure une communication fiable et sécurisée entre les deux parties, sans interférence extérieure. Les communications multicast et broadcast, qui impliquent l'envoi de données à plusieurs destinataires, ne sont pas prises en charge par TCP et nécessitent l'utilisation d'autres protocoles comme UDP pour le multicast et les mécanismes spécifiques du réseau pour le broadcast.

   Remarque :

   Notons ques ces caractéristiques ont été listées selon un ordre qui présente les concepts de connexion et de fiabilité, avant de passer à l'intégrité des données, celui du contrôle de flux et de congestion, et enfin aux aspects de multiplexage et de transmission bidirectionnelle/point à point.

8.2.2 Segment TCP

Le segment TCP contient plusieurs champs dans son en-tête comme indiqué dans la Figure 8.2. Chaque champ ayant une fonction spécifique. Dans ce qui suit, nous allons détailler la fonction de chacun de ces champs :

1. **Source Port (16 bits) :** Ce champ spécifie le numéro de port de la source, identifiant l'application expéditrice.

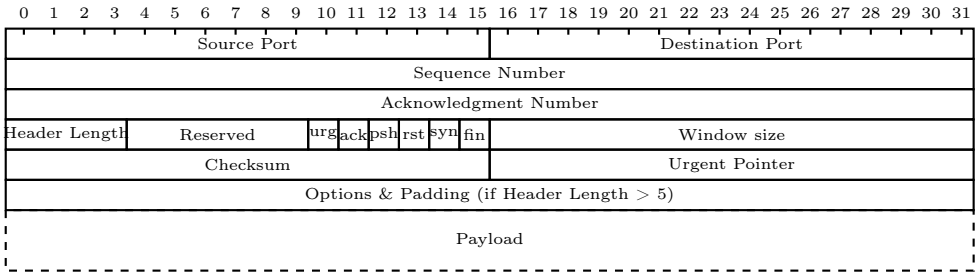


FIGURE 8.2 – Entête + Charge Utile d'un segment TCP

2. **Destination Port (16 bits) :** Ce champ spécifie le numéro de port de la destination, identifiant l'application destinataire.

   Notons qu'un port associé à une adresse IP d'une station forment une adresse TSAP [3] unique de 48 bits, appelée Socket. Le doublet (Numéro de socket source, Numéro de socket destination) identifie une connexion, par exemple, 192.168.1.1 :8080.

   Exemple de ports :

   — 21 : FTP

   — 23 : Telnet

   — 25 : SMTP

   — 110 : POP3

   — 80 : HTTP

   — etc

3. **Sequence Number (32 bits) :** Ce champ est utilisé pour numéroter les octets des données transmises. Il indique le numéro de séquence du premier octet de ce segment.
4. **Acknowledgment Number (32 bits) :** Ce champ est utilisé pour accuser réception des octets reçus. Il contient le numéro de séquence du prochain octet attendu du côté récepteur.

   Notons que TCP n'acquitte pas un à un chaque segment qu'il reçoit, mais acquitte l'ensemble du flot de données jusqu'à l'octet k-1 en envoyant un acquittement de valeur k.
5. **Header Length (4 bits) :** Aussi appelé "Data offset", ce champ indique la taille de l'entête TCP en mots de 32 bits. Il permet de savoir où commencent les données dans le segment. Un en-tête TCP peut avoir une taille variant de 20 octets (5 mots de 32 bits - bits 0101, aucune option) à 60 octets (15 mots de 32 bits -bits 1111, maximum d'options).
6. **Reserved (6 bits) :** Ce champ est réservé pour une utilisation future et doit être mis à zéro.
7. **Flags (6 bits) :** Plusieurs indicateurs (flags) de contrôle, chacun ayant un rôle spécifique :

   — **URG (1 bit) :** Indique que le champ Urgent Pointer est significatif.

   — **ACK (1 bit) :** Indique que le champ Acknowledgment Number est significatif.

3. Transport Service Access Point

- **PSH (1 bit) :** Demande une transmission immédiate des données au niveau de l'application réceptrice.
- **RST (1 bit) :** Réinitialise la connexion.
- S**YN (1 bit) :** Utilisé pour synchroniser les numéros de séquence lors de l'établissement de la connexion.
- **FIN (1 bit) :** Indique que l'expéditeur a terminé l'envoi de données.

8. **Window Size (16 bits) :** Ce champ spécifie la taille de la fenêtre de réception, indiquant combien d'octets de données le récepteur est prêt à recevoir.
9. **Champ total de contrôle (Checksum) (16 bits) :** Utilisé pour vérifier la validité de l'entête et des données transmises. Il est obligatoirement calculé par l'émetteur et vérifié par le récepteur.
10. **Checksum (16 bits) :** Ce champ est utilisé pour vérifier l'intégrité de l'en-tête TCP et des données.
11. **Urgent Pointer (16 bits) :** Utilisé uniquement si le flag URG est activé. Il pointe vers les données urgentes dans le segment.
12. **Options (variable) :** Ce champ optionnel peut contenir diverses options TCP, telles que la négociation de la taille maximale des segments (MSS), les horodatages, etc.
13. **Padding :** Des bits de bourrage sont ajoutés pour s'assurer que l'en-tête TCP est un multiple de 32 bits.

8.2.3 Fonctionnement d'une connexion TCP

Les étapes clès du fonctionnement d'une connexion TCP sont l'établissement de la connexion, le transfert des données, et la terminaison (ou libération) de la connexion.

8.2.3.1 Etablissement d'une connexion TCP (Three-Way Handshake)

L'établissement d'une connexion TCP nécessite 3 étapes connues sous le nom de "Three-way Handshake". Ces 3 étapes correspondent à l'échange de 3 segments : **SYN**, **SYN-ACK** et **ACK** comme l'illustre la Figure 8.3.

- **SYN (Synchronize) :** Le client envoie un segment TCP avec le drapeau SYN mis à 1 pour demander une connexion. Ce segment contient un numéro de séquence initial.
- **SYN-ACK (Synchronize-Acknowledgment) :** Le serveur répond avec un segment contenant à la fois les drapeaux SYN et ACK (acknowledgment) mis à 1. Le numéro de séquence du serveur est inclus, ainsi qu'un accusé de réception pour le numéro de séquence initial du client.
- **ACK (Acknowledgment) :** Le client répond avec un segment contenant le drapeau ACK mis à 1, accusant réception du numéro de séquence du serveur. À ce stade, la connexion est établie en mode full-duplex, c'est-à-dire dans les deux sens.

La Figure 8.4, détaille davantage cet échange :

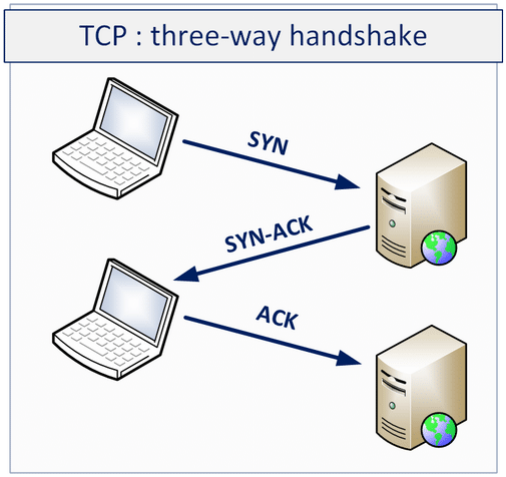


FIGURE 8.3 – Le Three-way Handshake [21].

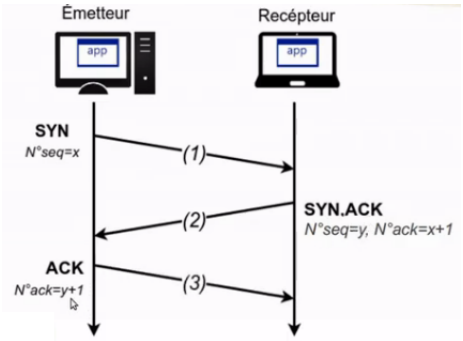


FIGURE 8.4 – Détail de l'etablissement d'une connexion TCP [20]

1. **Initiation par l'émetteur :**
   - L'émetteur (client) envoie un segment SYN avec un numéro de séquence initial (ISN [4]) x.
   - Drapeau SYN = 1 pour indiquer une demande de connexion.
2. **Réponse du récepteur :**
   - Le récepteur (serveur) répond avec un segment SYN-ACK.
   - Il inclut un accusé de réception (ACK) pour x+1, indiquant qu'il a reçu et reconnu le segment SYN du client.
   - Le serveur inclut son propre ISN y dans ce segment.
   - Drapeau SYN = 1 pour indiquer qu'il accepte la connexion et envoie sa propre demande de synchronisation.
3. **Confirmation de l'émetteur :**

4. Initial Sequence Number

— L'émetteur envoie un segment ACK en réponse, accusant réception du SYN-ACK du serveur.

— Il inclut un accusé de réception (ACK) pour y+1, indiquant qu'il a reçu et reconnu le segment SYN du serveur.

— Drapeau SYN = 0 car la phase de synchronisation est terminée.

Ce processus assure que les deux parties ont synchronisé leurs numéros de séquence et sont prêtes à échanger des données.

Remarques concernant l'ISN :

— Rappelons que les ISN permettent de suivre et de gérer l'ordre des segments de données échangés entre le client et le serveur, assurant que les données arrivent dans le bon ordre et sans perte (Gestion des flux de données).

— L'ISN est généré aléatoirement par chaque hôte. Sous Wireshark, on peut récupérer la valeur de l'ISN en regardant le champ "Sequence Number (raw)".

— Le choix aléatoire des ISN aide à protéger contre certaines attaques, comme l'usurpation de connexion (TCP sequence prediction attack).

— Notons que pour faciliter la lecture et la compréhension lorsqu'un humain analyse la capture de segments, Wireshark utilise un champ appelé le **numéro de séquence relatif** qui est positionné à 0 initialement. Il permet à un individu d'analyser les segments.

— Le fait que le champ "**numéro de séquence relatif**" commence à 0, pourrait laisser croire que l'ISN débute à 0 sur chacun des deux hôtes, ce qui totalement **faux** puisque la valeur initiale de l'ISN est générée aléatoire par chaque machine.

— Rappelons nous que sous Wireshark, la **vraie** valeur de l'ISN est stockée dans le champ "Sequence Number (raw)".

8.2.3.2 Transfert de données

Une fois la connexion établie, les données peuvent être échangées entre le client et le serveur (dans les deux sens puisque les connexions TCP sont bidirectionnelles) de manière fiable et ordonnée :

— **Numéros de séquence :** Chaque octet de données dans un segment TCP est numéroté séquentiellement. Ces numéros de séquence permettent de suivre les données et de s'assurer qu'elles arrivent dans le bon ordre.

— **Accusés de réception :** Le récepteur envoie des accusés de réception (ACK) pour les segments reçus. Cela permet à l'émetteur de savoir quelles données ont été correctement reçues.

— **Fenêtres d'émission :** TCP utilise un mécanisme de fenêtre glissante (sliding window) pour contrôler le flux de données. La fenêtre d'émission limite la quantité de données qu'un émetteur peut envoyer avant de recevoir un accusé de réception. Cela aide à éviter la congestion du réseau.

— **Contrôle de congestion :** TCP implémente des algorithmes comme le contrôle de congestion de TCP Tahoe et Reno pour éviter la saturation du réseau. Ces algorithmes ajustent dynamiquement la taille de la fenêtre d'émission en fonction des conditions du réseau.

Exemple d'un transfert de données via TCP :

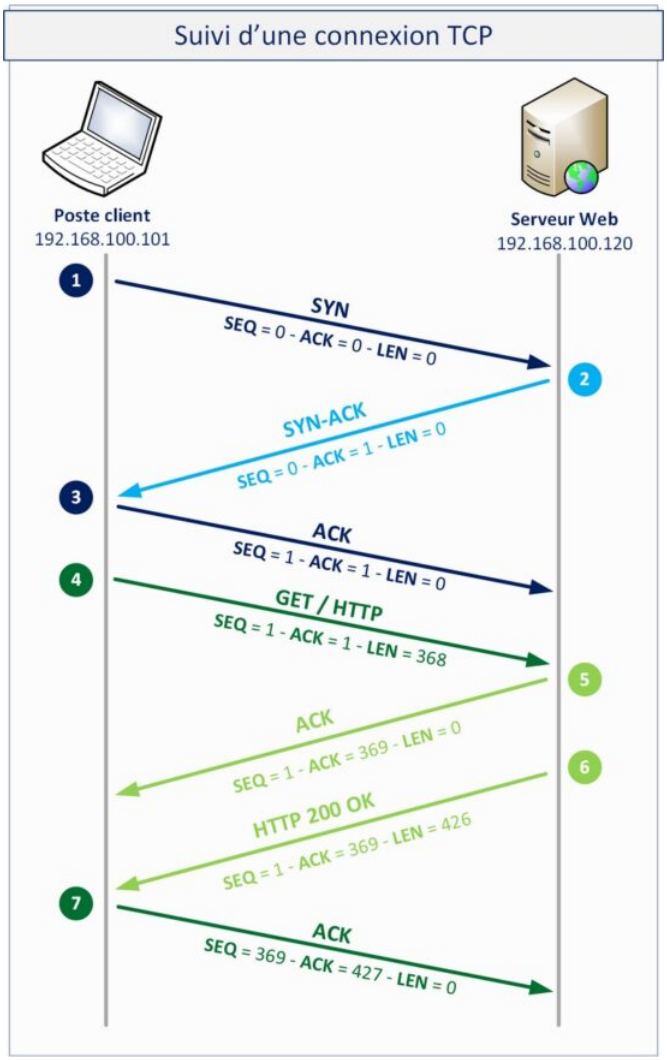


FIGURE 8.5 – Analyse d'une connexion TCP (Wireshark) [21].

La Figure 8.5 décrit les segments échangés lors d'un transfert de données via TCP.

— Le segment (4) correspond à la requête HTTP émise par le client au serveur Web. Il lui demande le contenu de la page Web.

— Le segment (5) est une réponse du serveur Web au client pour lui confirmer qu'il a bien reçu sa demande.

— Le segment (6) est toujours du serveur vers le client. Il correspond à l'envoi des données (contenu de la page Web) au client.

— Le segment (7) correspond à la réponse du client au serveur pour lui indiquer qu'il a reçu des données de sa part.

8.2.3.3 Clôture d'une connexion TCP (Four-Way Handshake)

Il faut (4) segments TCP pour libérer une connexion : un segment FIN et un segment ACK pour chaque direction. Cependant, il est possible que le premier segment ACK et le second segment FIN soient combinés dans un même segment, ce qui réduit le nombre à (3) segments.

— **FIN (Finish) :** L'une des parties (client ou serveur) envoie un segment avec le drapeau FIN mis à 1 pour indiquer qu'elle souhaite terminer la connexion.

— **ACK (Acknowledgment) :** L'autre partie accuse réception du segment FIN en envoyant un segment ACK (segment avec drapeau ACK mis à 1).

— **FIN (Finish) :** La partie qui a accusé réception du FIN envoie ensuite son propre segment FIN (segment avec drapeau FIN mis à 1) pour terminer la connexion de son côté.

— **ACK (Acknowledgment) :** La partie initiale envoie un dernier segment ACK (segment avec drapeau ACK mis à 1) pour accuser réception du FIN final. La connexion est alors complètement terminée.

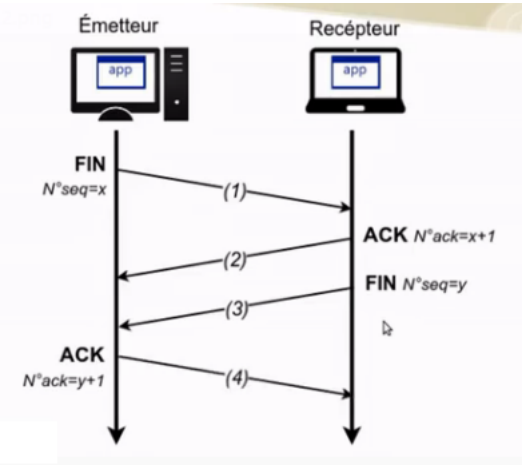


FIGURE 8.6 – Clôture d'une connexion TCP [20].

Remarques :

— Notons que lorsqu'un sens seulement de la connexion TCP est fermé, le flux de données peut continuer indéfiniment dans l'autre sens. Une fois que les deux directions sont fermées, la connexion est libérée (voir Figure 8.10). En effet, il est possible de fermer un sens de la connexion tout en continuant à envoyer des données dans l'autre sens. Ce mécanisme est utile pour certaines applications où une partie a terminé l'envoi de données mais doit encore recevoir des réponses. Ceci est appelé fermeture partielle de la connexion TCP.

— Comme nous l'avons signalé ci-dessus, Il faut normalement 4 segments TCP pour libérer une connexion TCP : un segment FIN et un segment ACK pour chaque direction. Cependant,

il est possible que le premier ACK et le second FIN soient combinés dans le même segment, ce qui réduit le nombre total à 3 :

- L'initiateur envoie un segment FIN (FIN1).
- L'autre partie envoie un segment combiné contenant à la fois l'ACK pour le FIN1 et son propre FIN (FIN2).
- L'initiateur envoie un segment ACK pour accuser réception du FIN2.

8.2.4 Fonctionnalités supplémentaires de TCP

- **Retransmissions :** Si la réception d'un segment n'a pas fait l'objet d'un accusé de réception dans un certain délai, le segment est retransmis.
- **Contrôle d'erreurs :** TCP utilise des sommes de contrôle (checksums) pour détecter les erreurs dans les segments de données. Les segments corrompus sont rejetés et retransmis.
- **Options TCP :** TCP permet également des options pour ajuster des paramètres spécifiques, comme la taille maximale de segment (MSS), les horodatages, etc.

8.2.5 Utilisations typiques de TCP

- Transfert de fichiers (FTP)
- Navigation web (HTTP/HTTPS)
- Courrier électronique (SMTP, IMAP, POP3)
- Protocoles de gestion de réseau (SSH, Telnet)

8.2.6 Concepts de la fenêtre glissante et ajustement de fenêtre

1. **Fenêtre glissante (sliding window) :**
   - Le concept de la fenêtre glissante est le mécanisme fondamental qui permet l'envoi continu de segments de données sans attendre l'envoi d'ACK après chaque segment.
   - La fenêtre glissante est une technique clé dans le protocole TCP, permettant une gestion efficace et fiable du flux de données entre l'émetteur et le récepteur, en fonction des capacités et des conditions actuelles de chaque côté de la connexion.
   - Son objectif est d'optimiser l'utilisation de la bande passante et maintenir un flux de données continu.
   - Le concept de fenêtre glissante est implémenté à travers le déplacement de la fenêtre au fur et à mesure que les ACKs sont reçus, permettant l'envoi de nouveaux segments.
2. **Ajustement de la fenêtre :**
   - L'ajustement de la fenêtre en cours de transmission fait référence aux modifications dynamiques de la taille de la fenêtre de réception en fonction des conditions actuelles du réseau et de la charge du système. Cet ajustement est crucial pour optimiser le flux de données et prévenir la congestion du réseau. C'est un autre concept clé dans le protocole TCP.

— Son objectif est de prévenir la surcharge du récepteur et gérer la congestion du réseau en adaptant la taille de la fenêtre de réception.

— Le concept de l'ajustement de la fenêtre est implémenté à travers l'augmentation ou la diminution de la taille de la fenêtre dynamiquement par le recepteur et la communication de la nouvelle valeur de la taille de la fenêtre via les ACKs.

3. **Premier Exemple illustratif :**

Supposons que nous avons une connexion TCP entre un client (émetteur) et un serveur (récepteur) :

— **Initialisation :**

— Fenêtre de réception initiale : 4 000 octets.

— Segments envoyés : 1 000 octets chacun.

— **Transmission avec fenêtre glissante :**

(a) **État initial :**

— Le client envoie quatre segments (Seq. numbers respectifs = 0, 1 000, 2 000, 3 000).

(b) **Réception des ACK :**

— Le serveur reçoit les segments et envoie un ACK = 4 000.

— **Glissement de la fenêtre :**

— La fenêtre glisse et permet l'envoi de nouveaux segments (Seq. numbers respectifs = 4 000, 5 000, 6 000, 7 000).

— **Ajustement de la fenêtre en cours de transmission :**

(a) **Changement de conditions :**

— Le serveur détecte une charge élevée et réduit sa capacité de réception. Il envoie un ACK avec une nouvelle taille de fenêtre de 2 000 octets.

(b) **Réaction de l'émetteur :**

— Le client ajuste sa fenêtre d'envoi pour ne plus envoyer que deux segments à la fois (Seq. numbers respectifs = 8 000, 9 000).

(c) **Amélioration des conditions :**

— Le serveur peut à nouveau gérer plus de données et annonce une fenêtre de réception de 6 000 octets dans un nouvel ACK.

(d) **Réaction de l'émetteur :**

— Le client ajuste sa fenêtre d'envoi et peut envoyer jusqu'à six segments à la fois (Seq. numbers respectifs = 10 000, 11 000, 12 000, 13 000, 14 000, 15 000).

4. **Deuxième Exemple illustratif :**
   Supposons que :
   - Émetteur (Client) : Envoie des données vers le récepteur (Serveur).
   - Récepteur (Serveur) : A annonce une fenêtre de 4 000 octets.
   - Taille des segments envoyés : 1 000 octets chacun.
   - Numéro de séquence initial de l'émetteur : 0
   - **Envoi initial de segments :**
     L'émetteur envoie les 4 premiers segments :
     - Segment 1 : Seq = 0, Longueur = 1 000 octets.
     - Segment 2 : Seq = 1 000, Longueur = 1 000 octets.
     - Segment 3 : Seq = 2 000, Longueur = 1 000 octets.
     - Segment 4 : Seq = 3 000, Longueur = 1 000 octets.
   - **Accusé de réception (ACK)**
     Le serveur reçoit les 4 segments et envoie un ACK pour le dernier segment reçu :
     - ACK = 4 000 (indique que les octets 0 à 3 999 ont été reçus et que le serveur attend l'octet 4 000).
   - **Glissement de la fenêtre**
     L'émetteur reçoit l'ACK = 4 000. La fenêtre glisse de 4 000 octets, permettant à l'émetteur d'envoyer quatre nouveaux segments :
     - Segment 5 : Seq = 4 000, Longueur = 1 000 octets.
     - Segment 6 : Seq = 5 000, Longueur = 1 000 octets.
     - Segment 7 : Seq = 6 000, Longueur = 1 000 octets.
     - Segment 8 : Seq = 7 000, Longueur = 1 000 octets.
   - **Perte et retransmission**
     - Supposons que le segment 6 est perdu en cours de route. Le serveur reçoit les segments 5, 7 et 8 et envoie un ACK = 5 000, car il attend toujours le segment 6.
   - **Réaction de l'émetteur**
     - L'émetteur reçoit l'ACK = 5 000 et sait que le segment 6 a été perdu. Il retransmet le segment 6. Nous tenons à rappeler que l'émetteur détecte la perte du segment 6 de deux manières principales :
       (a) **Réception de plusieurs ACK dupliqués**
       - L'émetteur reçoit plusieurs ACK = 5000. Normalement, un émetteur TCP considère la réception de trois ACK dupliqués comme un indicateur de perte de segment.

- Après avoir envoyé le segment 7, l'émetteur reçoit un ACK = 5000 (indiquant que le serveur attend encore le segment 6).
- Après avoir envoyé le segment 8, l'émetteur reçoit encore un ACK = 5000.
- La réception de trois ACK dupliqués (ACK = 5000) signale à l'émetteur que le segment 6 est probablement perdu.
- Nous constatons ici que même si les autres données (les segments 7 et 8) sont correctement reçues, TCP ne les acquitte pas si leur numéros de séquence est supérieur à celui attendu (5000). L'emetteur renvoie un acquittement contenant le numéro de séquence attendu.

(b) Expiration du temporisateur de retransmission

- En plus de surveiller les ACK dupliqués, l'émetteur utilise un temporisateur de retransmission pour chaque segment envoyé. Ce temporisateur est appelé le temporisateur de retransmission ou retransmission timeout (**RTO**) en Anglais.
- Si le temporisateur pour le segment 6 expire avant de recevoir un ACK pour ce segment, l'émetteur retransmet le segment 6.

- Segment 6 (retransmis) : Seq = 5 000, Longueur = 1 000 octets.

- **Réception et glissement final**
  - Le serveur reçoit enfin le segment 6 et envoie un ACK = 8 000 pour indiquer qu'il a reçu tous les segments jusqu'à l'octet 7 999. La fenêtre glisse encore une fois, permettant l'envoi de nouveaux segments :
    - Segment 9 : Seq = 8 000, Longueur = 1 000 octets.
    - Segment 10 : Seq = 9 000, Longueur = 1 000 octets.
    - Segment 11 : Seq = 10 000, Longueur = 1 000 octets.
    - Segment 12 : Seq = 11 000, Longueur = 1 000 octets.

8.2.7 Negociation des tailles des fenêtres et du MSS

En ce qui concerne la négociation des tailles des fenêtres :

1. **Remarque 1 :**
   - Au moment de l'établissement de la connexion, l'émetteur et le récepteur négocient la taille de la fenêtre via l'échange des segments SYN et SYN-ACK.
   - La taille de la fenêtre permet à l'émetteur de savoir combien d'octets il peut envoyer avant de devoir attendre un accusé de réception (ACK) du récepteur.
   - Elle permet aussi au récepteur d'indiquer à l'émetteur combien d'octets de données il est prêt à recevoir avant d'envoyer un ACK.
   - La taille de la fenêtre peut être ajustée dynamiquement pendant la communication pour s'adapter aux conditions du réseau et à la charge du récepteur.

2. **Remarque 2 :**
   - **Indépendance des tailles de fenêtre :** Chaque côté de la connexion gère et annonce sa propre taille de fenêtre de réception. La taille de la fenêtre du client (W_{client}) et celle du serveur ($W_{serveur}$) sont indépendantes et peuvent être différentes.
   - **Adaptabilité :** Les tailles de fenêtre peuvent être ajustées dynamiquement pendant la communication en fonction des conditions du réseau, de la charge du système et d'autres facteurs.
   - **Contrôle de flux :** Les tailles de fenêtre de réception permettent de contrôler le flux de données, en évitant que l'émetteur n'envoie trop de données d'un coup, ce qui pourrait saturer le récepteur.

En ce qui concerne le MSS [5]

- Le MSS spécifie la quantité maximale de données utiles (payload) que chaque segment TCP peut contenir.
- Le MSS est conçu pour éviter la fragmentation IP, qui se produit lorsque les paquets IP sont trop grands pour être transmis dans un seul fragment sur un réseau sous-jacent. La fragmentation peut entraîner une surcharge réseau et une diminution des performances.
- Le MSS est déterminé lors de l'établissement de la connexion TCP via l'option MSS dans le paquet SYN. Les deux hôtes négocient alors le MSS le plus adapté à leur connexion.
- Le calcul du MSS se fait comme suit :
  - MSS = MTU [6] - Taille de l'en-tête IP - Taille de l'en-tête TCP.
  - Par exemple, sur un réseau Ethernet standard avec une MTU de 1500 octets :
    - MSS = 1500 octets - 20 octets (en-tête IP) - 20 octets (en-tête) TCP) = 1460 octets

8.2.8 Exemples de synthèse

1. **Exemple 1 : Etablissement d'une connexion TCP** En supposant que le numéro de séquence initial de l'emetteur est 904 et le numéro de squénce du récepteur est 2500, l'établissement de la connexion TCP se fait comme illustré dans la Figure 8.7
2. **Exemple 2 : Echange avec acquittement groupé (Sliding Window)**
   La Figure 8.8 illustre le concept de la fenêtre glissante.
3. **Exemple 3 : Echange avec accusés de réception**
   La Figure 8.9 illustre un échange avec accusés de réception.
4. **Exemple 4 : Echange TCP complet**
   Soit le cas de figure suivant :
   - A initie la communication

5. Maximum Segment Size
6. Maximum Transmission Unit

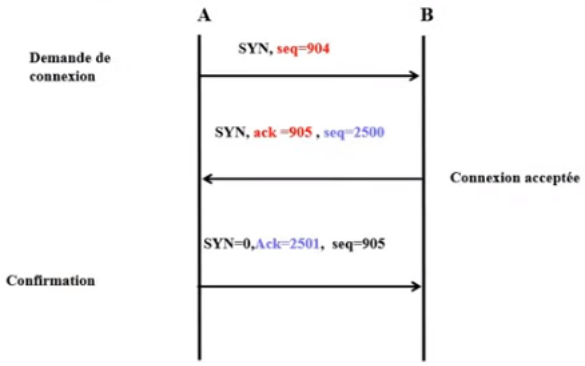


FIGURE 8.7 – Etablissement d'une connexion TCP [22]

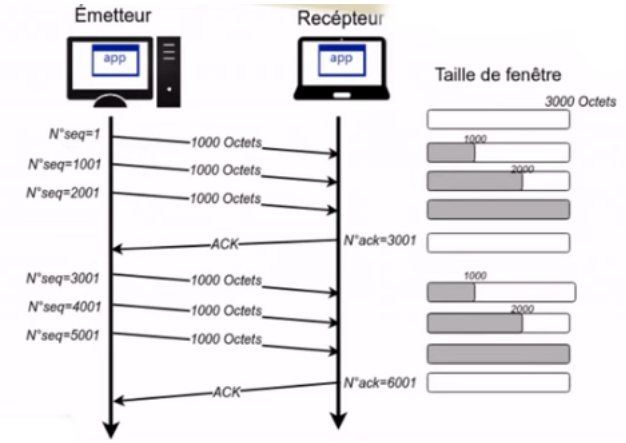


FIGURE 8.8 – Exemple d'échanges TCP avec acquittement groupé (fenêtre glissante) [20]

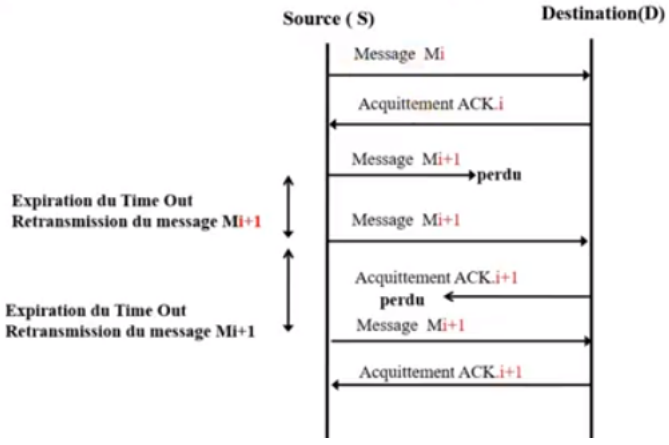


FIGURE 8.9 – Exemple d'échange avec accusés de réception [22]

— B ferme la connexion

— Numéro de séquence initial de A est 20 et ce lui de B est 40

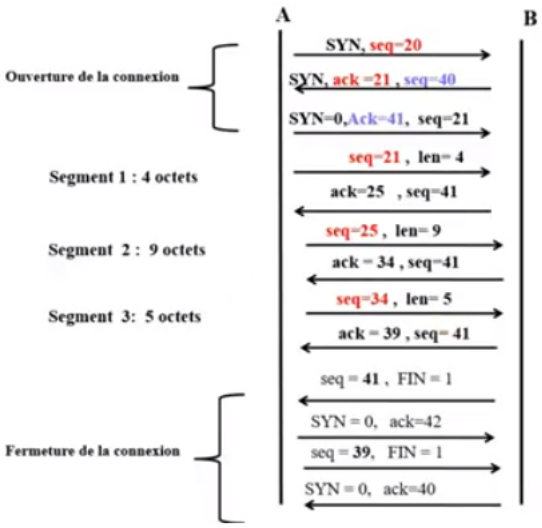


FIGURE 8.10 – Echanges TCP [22].

— La taille des données transférées de A vers B est de 18 octets :
  — Les données sont transmises en 3 segments de taille 4, 9 et 5 octets respectivement.
  — Sans perte de données ni d'acquittements.

La Figure 8.10 illustre toutes les étapes d'un échange TCP. Notons que lors de cet échange, B n'envoie pas de données à A. Par conséquent son suméro de séquence ne change pas.

8.3 User Datagram Protocol (UDP)

Le User Datagram Protocol (UDP) est un protocole de transport **sans connexion**, ce qui signifie qu'il envoie des datagrammes indépendants sans établir une connexion préalable. UDP est conçu pour être simple et rapide, sans les garanties de fiabilité fournies par TCP.

8.3.1 Principales caractéristiques du protocole UDP

— **Sans connexion :** UDP n'établit pas de connexion avant d'envoyer des données, ce qui réduit la latence.

— **Non fiable :** UDP n'assure pas la livraison fiable des paquets. Les paquets peuvent être perdus, dupliqués ou arriver dans le désordre sans correction automatique.

— **Faible surcharge :** UDP a une surcharge minimale en termes de contrôle de flux et de congestion, ce qui permet des transmissions plus rapides et plus légères.

— **Transmission directe :** Les paquets UDP sont envoyés directement à la destination sans vérification préalable de l'état du récepteur.

Remarques :

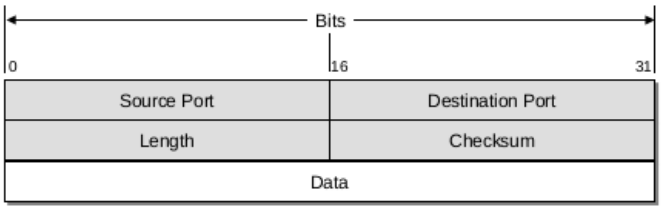


FIGURE 8.11 – Datagramme UDP [23]

La communication UDP est principalement un protocole de communication point-à-point, mais il peut aussi prendre en charge des communications de type point-à-multipoint. Les principales caractéristiques de la communication UDP sont :

— **Point-à-point :** Dans sa forme la plus basique, UDP établit une communication directe entre un expéditeur (client) et un récepteur (serveur) identifiés par leurs adresses IP et numéros de port. C'est le mode de communication le plus courant pour les applications utilisant UDP.

— **Point-à-multipoint (diffusion/multidiffusion) :** UDP permet aussi d'envoyer des datagrammes à plusieurs destinataires simultanément. Nous rapellons que la diffusion (broadcast) consiste à envoyer les données à tous les hôtes sur un même réseau local, quant à la multidiffusion (multicast), elle consiste à envoyer les données à un groupe d'hôtes abonnés à un groupe multicast spécifique.

8.3.2 Datagramme UDP

Le datagramme UDP a une structure très simple, composée de quatre champs principaux dans son en-tête, suivis des données utiles comme le montre la Figure 8.11.

— **Port Source (16 bits) :** Identifie le port du processus émetteur sur l'hôte source. Ce champ est facultatif ; si aucune valeur n'est fournie, il est mis à zéro.

— **Port Destination (16 bits) :** Identifie le port du processus récepteur sur l'hôte destination. Ce champ est obligatoire pour s'assurer que le datagramme est reçu par l'application correcte.

— **Longueur (16 bits) :** Spécifie la longueur totale (en octets) du datagramme UDP, incluant l'en-tête et les données utiles. La longueur minimale est de 8 octets (uniquement l'en-tête).

— **Checksum (16 bits) :** Utilisé pour détecter les erreurs dans l'en-tête et les données du datagramme. Ce champ est facultatif dans IPv_4 (peut être mis à zéro), mais obligatoire dans IPv6.

— **Données (variable) :** Contient les données de l'application. La taille des données est déterminée par la valeur du champ Longueur, moins la taille de l'en-tête (8 octets).

8.3.3 Utilisations typiques de UDP

— Diffusion en continu de vidéos et de jeux en ligne

— Voix sur IP (VoIP)
— Protocoles de découverte de réseau (DHCP)
— Applications en temps réel où la vitesse est plus critique que la fiabilité

Plus spécifiquement :

— **TFTP (Trivial File Transfer Protocol) :** Utilise exclusivement UDP pour le transfert de fichiers.
— **DNS (Domain Name System) :** Utilise principalement UDP pour les requêtes de résolution de nom, bien qu'il puisse basculer sur TCP pour des transferts de zone ou des réponses de grande taille.
— **NFS (Network File System) :** Historiquement basé sur UDP, les versions récentes utilisent souvent TCP pour améliorer la fiabilité et les performances.
— **SNMP (Simple Network Management Protocol) :** Utilise UDP pour la gestion et la surveillance des réseaux.
— **RIP (Routing Information Protocol) :** Utilise UDP pour l'échange d'informations de routage entre les routeurs.

8.4 Autres protocoles de transport couramment utilisés

Voici quelques autres protocoles de transport couramment utilisés, en plus de TCP et UDP :

— **SCTP (Stream Control Transmission Protocol) :** Un protocole de transport orienté connexion, fiable et sans perte, qui offre des fonctionnalités améliorées par rapport à TCP, comme la multidiffusion, la répartition de charge et la récupération rapide en cas de pannes.
— **DCCP (Datagram Congestion Control Protocol) :** Un protocole de transport orienté connexion, qui fournit un contrôle de congestion pour les applications sensibles au délai comme les jeux en ligne et la vidéo en streaming.
— **RTP (Real-time Transport Protocol) :** Un protocole de transport principalement utilisé pour les applications de voix sur IP et de vidéoconférence en temps réel, qui fournit des fonctionnalités de gestion du temps et de séquençage des paquets.
— **QUIC (Quick UDP Internet Connections) :** Un protocole de transport récent, conçu par Google, qui utilise UDP comme couche de transport tout en offrant des fonctionnalités similaires à TCP, comme la gestion de la congestion et la fiabilité.
— **HTTP/3 (HTTP over QUIC) :** Une version récente du protocole HTTP qui s'appuie sur QUIC pour améliorer les performances et la sécurité des connexions web.

8.5 Les ports

Les ports réseau sont des points de communication utilisés par les protocoles de transport, tels que TCP et UDP, pour identifier des services spécifiques sur un hôte. Les ports sont identifiés par des numéros de 16 bits comme indiqué respectivement dans les Figures 8.2 et 8.11, ce qui donne une plage de 0 à 65535.

8.5.1 Catégories des ports

L'IANA [7] classe les numéros de port en trois catégories principales, comme l'illustre le Tableau 8.1 :

Tableau 8.1 – Catégorie des ports

Catégorie	Plage	Description	Exemples
Ports bien connus	0-1023	Services et protocole standardisés	80 (HTTP), 443 (HTTPS)
Ports enregistrés	1024-49151	Applications Spécifiques et Services	3306 (MySQL), 8080 (HTTP alternatif)
Ports Dynamiques	49152-65535	Connexions Temporaires ou Ephémères	Utilisés par des clients. Ces ports sont alloués dynamiquement par l'O.S lorsqu'une application démarre une connexion réseau

8.5.2 Exemples de ports bien connus (Well known ports)

La RFC [8] 1060 indique les ports prédéfinis pour les services (voir Tableau 8.2).

8.6 Round-Trip Time (RTT)

— Le RTT [9], ou temps de trajet aller-retour, est une mesure de la latence d'un réseau. Il représente le temps qu'il faut pour qu'un paquet de données soit envoyé d'un point A à un point B, et qu'un accusé de réception (ou une réponse) soit renvoyé de B à A.

— Le RTT est une métrique importante pour évaluer les performances d'un réseau. Un RTT faible signifie une communication rapide et réactive, tandis qu'un RTT élevé indique une latence plus importante, ce qui peut affecter les performances des applications, notamment celles nécessitant des échanges fréquents de données, comme le web, les jeux en ligne et la vidéoconférence.

— Notons que malgré que le protocole UDP soit utilisé pour les applications en temps réel comme les jeux en ligne et la vidéoconférence en raison de ses faibles délais et de l'absence de contrôle de flux, le RTT reste une mesure cruciale. Il affecte directement la latence, la qualité de service et la réactivité de ces applications.

— Le RTT peut être mesuré à l'aide de divers outils et protocoles, dont le plus courant est la commande ping.

7. Internet Assigned Numbers Authority
8. Request For Comments
9. Round-Trip Time

TABLEAU 8.2 – Exemples de ports bien connus.

Service	Port	Service	Port
FTP	20 et 21	FTPS	989 et 990
SSH	22		
Telnet	23		
SMTP	25		
DNS	53		
TFTP	69		
Web	80	HTTPS	443
POP3	110	POP3S	995
IMAP	143	IMAPS	993
SNMP	161		
BGP	179		
RIP	520		

— Le RTT a un impact direct sur les performances de TCP, notamment sur :
  — Temps de réponse : Les applications interactives, comme la navigation web, seront plus réactives avec un RTT plus faible.
  — Contrôle de flux : TCP utilise le RTT pour ajuster les temporisateurs de retransmission et gérer la fenêtre de congestion.
  — Fenêtre glissante : Une fenêtre glissante bien gérée peut aider à compenser les effets d'un RTT élevé en permettant l'envoi de plusieurs segments avant de recevoir des ACK.

Exemple illustratif de calcul du RTT :
Supposons que nous avons deux ordinateurs, A et B :

— A envoie un paquet à B.
— Le paquet met 50 ms pour atteindre B.
— B renvoie un accusé de réception (ACK) à A.
— L'ACK met 50 ms pour revenir à A.

Dans ce cas, le RTT est de 50 ms (aller) + 50 ms (retour) = 100 ms.

8.7 Conclusion

TCP et UDP sont deux protocoles de la couche de transport qui offrent deux approches différentes pour le transport des données sur un réseau.

— TCP, avec sa fiabilité et ses mécanismes de contrôle de flux et de congestion, est approprié pour les applications nécessitant une transmission précise et ordonnée des données.

— Quant au protocole UDP, avec sa simplicité et sa rapidité, il convient mieux aux applications où la vitesse et la faible latence sont essentielles, et où une certaine perte de paquets est acceptable.

Par conséquent, le choix entre ces deux protocoles de transport, en l'occurrence TCP et UDP, dépend des exigences spécifiques de l'application en termes de fiabilité, de performance et de complexité.

Chapitre 9

NAT-PAT

9.1 Introduction

Le NAT [1] et le PAT [2] constituent des techniques fondamentales dans les réseaux IPv_4. Elles permettent notamment une utilisation éfficace des adresses IPv_4 et offrent des avantages en termes de sécurité et de flexibité, comme nous allons le détailler à travers ce Chapitre.

9.2 Bref historique

La croissance rapide d'Internet au début des années 1990 a entraîné une demande accrue d'adresses IPv_4. Par conséquent, à la fin de cette décennie, il est devenu évident que le stock d'adresses IPv_4 finirait par s'épuiser. Voici les étapes importantes :

— **Début des années 1990** : L'utilisation d'Internet a explosé, entraînant une demande accrue d'adresses IPv_4.

— **Fin des années 1990 à début des années 2000** : L'épuisement des adresses IPv_4 est devenu une préoccupation majeure alors que le pool d'adresses commençait à se réduire rapidement.

— **2008** : L'Internet Assigned Numbers Authority (IANA) a attribué les derniers blocs d'adresses IPv_4 aux Registres Internet Régionaux (RIR), marquant une reconnaissance formelle du problème d'épuisement.

Pour faire face à l'épuisement rapide des adresses IPv_4, plusieurs solutions ont été conçues et mises en oeuvre :

— NAT

— Adressage privé

— Classless Inter-Domain Routing (CIDR) (Subnetting, VLSM [3])

— Transition vers IPv_6

1. Network Address Translation
2. Port Address Translation
3. Variable Length Subnet Mask

9.3 NAT (Network Address Translation)

Le NAT est une technique qui permet de modifier les adresses IPv_4 dans les paquets lors de leur passage d'un réseau local à un réseau externe, généralement l'Internet. Cette technique permet notamment de masquer les adresses IPv_4 privées des réseaux locaux (voir Tableau 9.1) derrière une ou plusieurs adresses IPv_4 publiques (voir Tableau 9.2) lors du transit des paquets d'un réseau local vers Internet.

Autrement, Le NAT est une technique utilisée pour permettre aux appareils d'un réseau privé d'accéder à des réseaux externes, notamment Internet, en traduisant leurs adresses IPv_4 privées en adresses IPv_4 publiques.

Tableau 9.1 – Plages d'adresses IPv_4 privées.

Classe	Plage d'adresses privées
A	de 10.0.0.0 à 10.255.255.255 (1 réseau de la classe A)
B	de 172.16.0.0 à 172.31.255.255 (16 réseaux de la classe B)
C	de 192.168.1.0 à 192.168.255.255 (255 réseaux de la classe C)

Remarque :

— Les adresses privées sont réservées à un usage **interne**. Elles peuvent être utilisées **librement** par tout administrateur ou utilisateur au sein de leur réseau local.

— Elles ne sont pas routées sur ce qu'on appelle le backbone de l'Internet (les réseaux de backbone). C'est pourquoi les routeurs de bord des FAI [4] sont configurés pour empêcher le routage de ces adresses (les routeurs Internet sont tous configurés pour éliminer toutes les adresses privées).

Tableau 9.2 – Plages d'adresses IPv_4 publiques.

Classe	Plage d'adresses publique
A	1.0.0.0 - 9.255.255.255 11.0.0.0 - 126-255.255.255
B	128.0.0.0 - 169.253.255.255 169.255.0.0 à 172.15.255.255 172.32.0.0 - 191-255.255.255
C	192.0.0.0- 192.167.255.255 à 192.169.0.0 - 223-255.255.255

Remarque :

4. Fournisseur d'Accès à Internet : Opérateur qui propose l'accès à Internet

— L'adressage public est soumis aux restrictions de déclaration et d'enregistrement de l'adresse IP auprès d'une organisation spécialisée, l'IANA, ce que les FAI font généralement en acquérant une plage d'adresses IP pour leurs abonnés.

— Pour obtenir une adresse IP publique ou un bloc d'adresses IP publiques, vous devez contacter votre FAI ou le Registre Internet Local (LIR). Ces entités sont responsables de l'obtention des pools d'adresses IP auprès des RIR dans leur région.

— Notez que la stabilité de l'Internet dépend de l'unicité de chaque adresse publique.

9.4 Variantes du NAT

Le processus générique de translation se décline en trois variantes, chacune ayant ses propres caractéristiques et utilisations :

1. **NAT Statique (Static NAT)** : Implique un mappage un-à-un entre une adresse IPv_4 privée et une adresse IPv_4 publique. Autrement dit, chaque adresse privée est associée de manière statique à une adresse publique.
   — **Utilisation** : Est utile lorsqu'on a besoin qu'un appareil interne soit constamment accessible depuis l'extérieur, comme un **serveur web** ou un **serveur de messagerie**.
   — **Exemple** : Un serveur web dans un réseau privé peut être associé à une adresse IPv_4 publique spécifique, garantissant ainsi qu'il soit toujours joignable à cette adresse.
2. **NAT Dynamique (Dynamic NAT)** : Le NAT dynamique utilise un pool d'adresses IPv_4 publiques et attribue temporairement une adresse IPv_4 publique à un appareil interne lorsqu'il en a besoin pour accéder à un réseau externe.
   — **Utilisation** : Cette méthode est utile lorsque plusieurs appareils internes ont besoin d'accéder à Internet, mais pas nécessairement en même temps.
   — **Exemple** : Lorsque des appareils internes effectuent des requêtes sortantes, ils se voient attribuer des adresses IPv_4 publiques disponibles du pool de manière dynamique.
3. **PAT (Port Address Translation) ou (NAT Overload, NAT Surchargé)** : Le PAT, également connu sous le nom de NAT Overload, permet à plusieurs appareils d'un réseau local de partager une seule adresse IPv_4 publique. Chaque session de communication est distinguée par un numéro de port unique.
   — **Utilisation** : Cette variante est largement utilisée dans les réseaux domestiques et les petits bureaux pour permettre à plusieurs appareils de partager une seule adresse IPv_4 publique.
   — **Exemple** : Plusieurs appareils dans une maison (ordinateurs, smartphones, tablettes) peuvent accéder simultanément à Internet en utilisant une seule adresse IPv_4 publique, le PAT gérant les différentes sessions grâce aux numéros de port. Un exemple concret est celui de l'accès à internet des appareils se trouvant dans nos maisons via la InternetBox ADSL d'Algérie Télécom.

9.5 Avantages de la translation d'adresses

Elle a des avantages en termes de :

— **Economie des adresses IP**v_4 : Le NAT et le PAT permettent de réduire le nombre d'adresses IPv_4 publiques nécessaires, un avantage crucial compte tenu de la pénurie d'adresses IPv_4 qui a vu le jour au début des années 90. En effet, comme nous l'avons déjà mentionné précédemment, la très forte croissance d'Internet au début des années 90 a mené très rapidement à la saturation des adresses IPv_4.

— **Sécurité** : En cachant les adresses IPv_4 internes, le NAT et le PAT ajoutent une couche de sécurité, en empêchant les utilisateurs externes d'accéder directement aux appareils internes.

— **Flexibilité** : Ils permettent une réorganisation des adresses IPv_4 internes sans affecter sans affecter l'accès à Internet, facilitant ainsi la gestion du réseau.

9.6 Synthèse

— La translation d'adresses est un processus générique qui permet de substituer une adresse par une autre, rendant ainsi possible la dissimulation des adresses privées des réseaux locaux derrière une ou plusieurs adresses IPv_4 publiques.

— Comme mentionné précédemment, ce processus joue un rôle crucial dans les réseaux modernes en améliorant à la fois la sécurité et la gestion des adresses.

— L'appareil réseau qui gère le NAT et le PAT est généralement le **routeur**.

— Le routeur traduit les adresses IPv_4 privées des appareils au sein du réseau local en une adresse IPv_4 publique pour la communication avec les réseaux externes, tels que l'Internet, en utilisant à la fois les techniques NAT et PAT.

— Ce faisant, le NAT et le PAT fournissent une **couche supplémentaire de sécurité** et **aident à conserver le nombre limité d'adresses IP**v_4 **publiques disponibles**.

9.7 Fonctionnement de la translation d'adresses

Pour permettre à un terminal disposant d'une adresse IPv_4 privée de communiquer avec le réseau public, le processus de NAT fait intervenir une entité tierce entre le terminal, ayant une adresse IPv_4 privée, et tout autre terminal ayant une adresse IPv_4 publique. Ce mécanisme consiste à insérer un boîtier entre le réseau Internet et le réseau local afin d'effectuer la translation de l'adresse IPv_4 privée en une adresse IPv_4 publique.

Aujourd'hui, la plupart des boîtiers ou InternetBox des FAI [5] proposent à leurs abonnés cette fonctionnalité. En effet, tous les appareils qui se connectent reçoivent par le biais du service DHCP [6] une adresse IPv_4 privée, que le boîtier se charge de translater en une adresse IPv_4 publique.

5. Fournisseur d'Accès à Internet
6. Dynamic Host Configuration Protocol

9.7.1 Principe du NAT

Nous tenons tout d'abord à rappeler rapidement ce qui suit :

— **NAT statique** traduit une adresse IPv_4 privée en toujours la même adresse IPv_4 publique. En effet, s'il y a 4 utilisateurs nécessitant une traduction d'adresse, il sera donc nécessaire d'utiliser 4 adresses IPv_4 publiques.

— **NAT dynamique** traduit une adresse IPv_4 privée avec une adresse IPv_4 publique appartenant à un pool d'adresses. Dans ce type de NAT, l'adresse IPv_4 publique utilisée pour la traduction n'est donc pas toujours la même. Par conséquent, les machines ne peuvent pas être atteintes depuis Internet car leur adresse n'est pas fixe. Le bénéfice évident est la sécurité. Enfin, il est à noter que s'il n'y a pas assez d'adresses IPv_4 publiques disponibles, les utilisateurs devront attendre qu'une adresse se libère pour pouvoir être traduits.

— La traduction d'adresses est principalement effectuée sur les **routeurs de bord** d'une entreprise connectée à Internet.

— Le réseau utilisant des adresses IPv_4 privées est ainsi appelé le réseau interne (**inside network**), tandis que la partie du réseau utilisant des adresses IPv_4 publiques (Internet) est appelée le réseau externe (**outside network**).

En ce qui concerne le fonctionnement proprement dit du NAT, l'opération de traduction se déroule comme suit :

— Lorsqu'un utilisateur sur le réseau interne (inside) souhaite communiquer avec un hôte sur le réseau externe (outside), le routeur (passerelle NAT) reçoit le paquet avec l'adresse IPv_4 privée.

— Selon le type de NAT utilisé, le routeur réécrit le paquet :

  — **NAT statique** : Le routeur remplace l'adresse IPv_4 privée par une adresse IPv_4 publique fixe.

  — **NAT dynamique** : Le routeur remplace l'adresse IPv_4 privée par une adresse IPv_4 publique provenant d'un pool d'adresses disponibles.

  — Ensuite, le routeur consulte sa table de routage pour acheminer le paquet vers la destination correcte.

9.7.2 Principe du PAT

Bref rappel :

— Le PAT, également connu sous le nom de **NAT overload**, permet d'attribuer une seule adresse IPv_4 publique pour la traduction de plusieurs adresses IPv_4 privées.

— Chaque utilisateur est différencié par un **numéro de port unique** qui lui est attribué lorsqu'il souhaite communiquer.

En ce qui concerne la traduction d'adresse lorsqu'on utilise le PAT, elle se fait comme suit :

— Le routeur remplace l'adresse IPv_4 privée par sa propre adresse IPv_4 publique et attribue un numéro de port unique pour distinguer les différents dispositifs internes.

Remarque :

— Étant donné qu'il y a 65 536 ports différents, un routeur pourrait théoriquement traduire jusqu'à 65 536 adresses IPv_4 privées différentes en utilisant une seule adresse IPv_4 publique. Cependant, en pratique, un dispositif gère généralement la traduction d'environ 2 000 à 10 000 ports par adresse IPv_4 publique, en fonction de facteurs tels que les performances du dispositif et l'efficacité de l'implémentation du NAT.

9.8 Multiplexage

Le multiplexage dans le contexte du NAT fait référence à la technique de gestion de multiples connexions en utilisant une seule adresse IPv_4 publique en les différenciant grâce à des informations supplémentaires, telles que les numéros de port.

Le PAT est une forme courante de multiplexage NAT où plusieurs adresses IPv_4 privées sont mappées à une seule adresse IPv_4 publique :

— Comme mentionné précédemment, le PAT est réalisé en utilisant différents numéros de port pour distinguer plusieurs connexions. Par exemple, une seule adresse IP publique peut gérer plusieurs connexions de différents dispositifs internes en attribuant un numéro de port unique à chaque connexion.

— Le dispositif NAT (ou la passerelle NAT) maintient une table de traduction qui mappe chaque combinaison d'adresse IP interne et de port à une combinaison d'adresse IP publique et de port. Cette table aide le dispositif à savoir quels paquets entrants doivent être transmis à quel dispositif interne.

9.9 Exemple 1

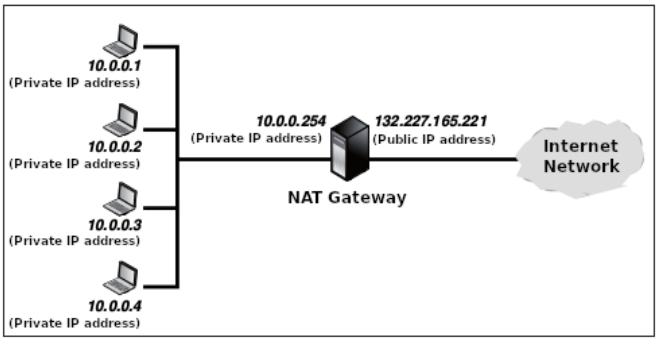


FIGURE 9.1 – Exemple 1 : Traduction d'adresse [24]

La figure 9.1 illustre un exemple où une passerelle NAT effectue la traduction d'adresses pour quatre terminaux. Cette passerelle possède deux interfaces réseau. La première est caractérisée

par une adresse IPv_4 publique (132.227.165.221). Connectée au réseau Internet, elle est reconnue et adressable normalement dans le réseau Internet.

La deuxième interface est caractérisée par une adresse IPv_4 non publique (10.0.0.254). Connectée au réseau local, elle ne peut communiquer qu'avec les terminaux ayant une adresse IPv_4 non publique du même réseau.

Lorsqu'un terminal avec une adresse IPv_4 privée essaie de se connecter au réseau Internet, il envoie ses paquets à la passerelle NAT. Celle-ci remplace l'adresse IPv_4 privée originale par sa propre adresse IPv_4 publique (132.227.165.221). Cette opération est appelée traduction d'adresse. De cette manière, les terminaux avec une adresse IPv_4 privée sont reconnus et adressables dans le réseau Internet par une adresse IPv_4 publique.

Par exemple, si un expéditeur avec une adresse IPv_4 de 10.0.0.3 envoie un paquet à la passerelle NAT depuis son port 12345, la passerelle NAT modifie le paquet en remplaçant l'adresse IPv_4 source par sa propre adresse IPv_4 publique et le port source par un port disponible, tel que 23456. La passerelle NAT note cette correspondance dans sa table NAT. Ainsi, lorsqu'elle reçoit un paquet destiné au port 23456, elle consulte cette affectation de port dans sa table et transfère le paquet à l'adresse IPv_4 interne et au port d'origine.

Remarque :

La traduction d'adresses est bien sûr effectuée dans les deux sens de la communication, afin de permettre la transmission des requêtes ainsi que la réception des réponses correspondantes. Pour cela, la passerelle NAT maintient une table de correspondance des paquets afin de savoir à qui distribuer les paquets reçus.

9.10 Exemple 2

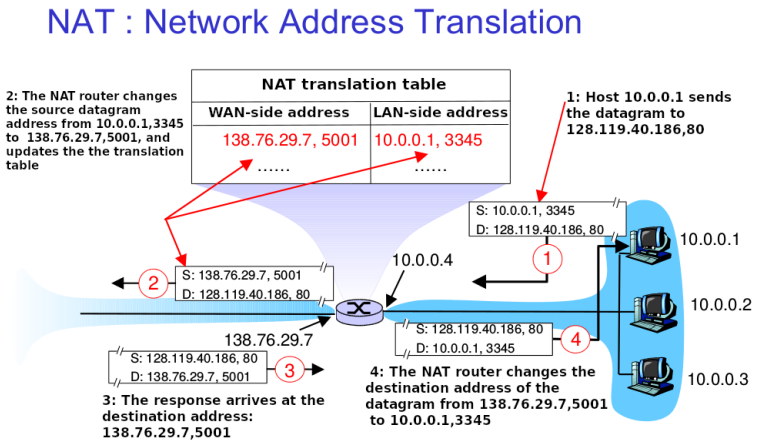


FIGURE 9.2 – Exemple 2 : Modification des paquets pendant le NAT [25]

La Figure 9.2 illustre la modification des paquets lors de leur passage du réseau interne vers le réseau externe.

9.11 Types d'adresses NAT selon Cisco

Cisco définit 4 types d'adresses pour le NAT :

— **Adresse locale interne** : Adresse IP assignée à un hôte dans le LAN.

— **Adresse globale interne** : Adresse(s) IP attribuée(s) par le FAI et reconnue(s) par Internet pour représenter le LAN.

— **Adresse locale externe** : Adresse IP d'un hôte sur le réseau externe telle qu'elle est connue des utilisateurs du réseau interne. La plupart du temps, elle est identique à l' "adresse globale externe".

— **Adresse globale externe** : Adresse IP assignée à un hôte dans le réseau externe.

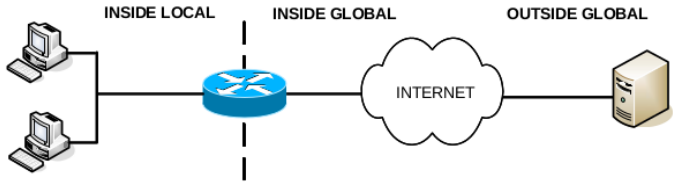


FIGURE 9.3 – Définitions de Cisco [26]

9.12 Configuration du NAT dans Packet Tracer

9.12.1 NAT statique

La Figure 9.4 illustre l'implémentation du NAT statique dans Packet Tracer.

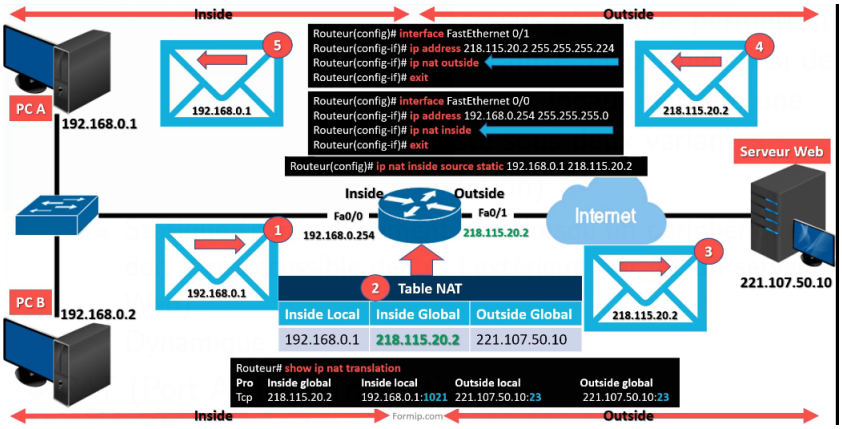


FIGURE 9.4 – NAT statique [27]

9.12.2 NAT dynamique

La Figure 9.5 illustre l'implémentation du NAT dynamique dans Packet Tracer.

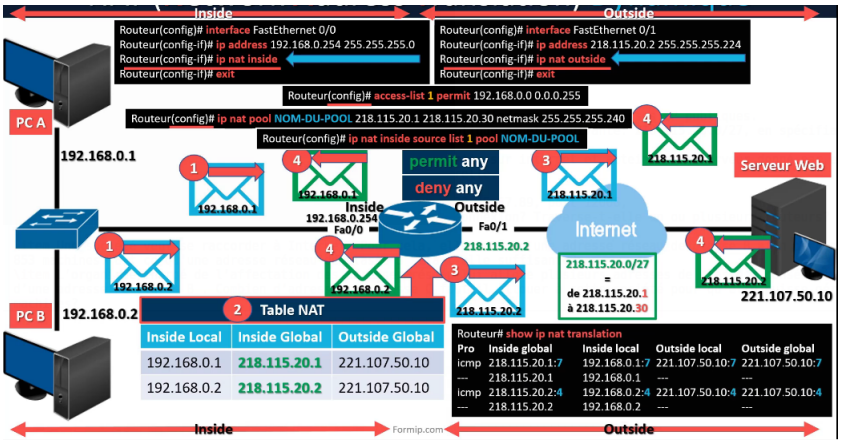


FIGURE 9.5 – NAT dynamique [27]

9.12.3 NAT overload

La Figure 9.6 illustre l'implémentation du PAT statique dans Packet Tracer.

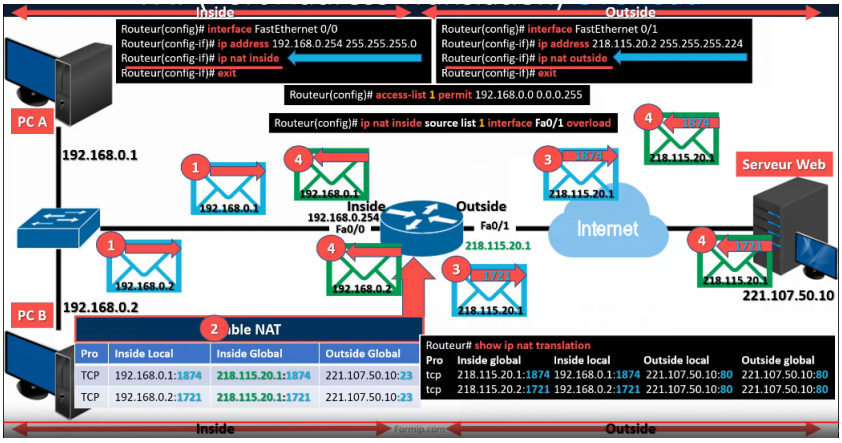


FIGURE 9.6 – Port Address Translation (NAT overload) [27]

9.13 Conclusion

Comme nous avons pu le découvrir tout au long de ce Chapitre, le NAT (Network Address Translation) et le PAT (Port Address Translation) constituent des mécanismes indispensables pour la gestion des réseaux modernes. Ils permettent en effet une utilisation efficace des adresses IPv_4 et offrent des avantages significatifs en termes de sécurité et de flexibilité. Ils constituent une des solutions conçues pour faire face à la pénurie des adresses IPv_4 apparue dès le début des années 90.

Chapitre 10

Optimisation du routage réseau : Bellman-Ford, Dijkstra et Floyd-Warshall

10.1 Introduction

Dans ce chapitre, nous abordons les techniques d'optimisation du routage dans les réseaux à l'aide de trois algorithmes fondamentaux : Bellman-Ford, Dijkstra et Floyd-Warshall. Ces algorithmes, notamment Bellman-Ford et Dijkstra, largement utilisés dans les protocoles de routage, permettent de déterminer les chemins les plus courts à travers un réseau en tenant compte des coûts et des distances entre les nœuds.

10.2 Bref Rappel

— **Dijkstra :** Chemin le plus court d'**un** nœud vers **tous** les nœuds, les poids d'arêtes **négatifs** ne sont pas autorisés.

— **Bellman-Ford :** Chemin le plus court d'**un** nœud vers **tous** les nœuds, les poids d'arêtes **négatifs** sont autorisés.

— **Floyd-Warshall :** Chemin le plus court entre **toutes** les paires de sommets, les poids d'arêtes **négatifs** sont autorisés.

10.3 Algorithme de Bellman-Ford

L'algorithme de Bellman-Ford partage certaines caractéristiques avec la programmation dynamique, cependant, il est plus précisément décrit comme un algorithme basé sur la relaxation.

L'algorithme de Bellman-Ford est utilisé pour trouver les plus courts chemins depuis un sommet source unique vers tous les autres sommets dans un graphe pondéré. Contrairement à

l'algorithme de Dijkstra, il peut gérer des graphes avec des arêtes de poids négatif, mais ne peut pas gérer les cycles de poids négatif.

10.3.1 Entrée de l'algorithme :

— Un graphe pondéré $G = (V, E, w)$ où V est un ensemble de sommets, E est un ensemble d'arêtes, et w représente le poids des arêtes.

— Un sommet source src à partir duquel les plus courts chemins seront calculés.

10.3.2 Sortie de l'algorithme :

— Les distances des plus courts chemins depuis le sommet source vers tous les autres sommets, stockées dans un tableau de distances.

— Un tableau des prédécesseurs pour reconstruire les chemins.

10.3.3 Pseudocode

Algorithme 10.1 Bellman-Ford : Pseudocode d'initialisation

Require: Graph $G(V, E, w)$ with vertices V, edges E and $w : E \to \mathbb{R}$ is a function that assigns a weight to each edge. Note that if there is no edge between vertex u and vertex v, weight(u, v) = ∞
Require: Source vertex $src \in V$
Ensure: Shortest path distances from src to all other vertices in G
Ensure: Predecessor information to reconstruct shortest paths

```
procedure BellmanFord(G, src)
    Initialize distance[] and predecessor[] arrays :
    for each vertex v in G except src do
        distance[v] = ∞
        predecessor[v] = null
    end for
    distance[src] = 0
end procedure
```

Algorithme 10.2 Bellman-Ford : Pseudocode de relaxation

```
procedure BellmanFord(G, src)
    for each vertex i from 1 to |V| − 1 do
        for each edge (u, v) in G do
            if distance[u] + weight(u, v) < distance[v] then
                distance[v] = distance[u] + weight(u, v)
                predecessor[v] = u
            end if
        end for
    end for
end procedure
```

Algorithme 10.3 Bellman-Ford : Pseudocode des cycles de poids négatif

```
1: procedure BellmanFord(G, src)
2:     for each edge (u, v) in G do
3:         if distance[u] + weight(u, v) < distance[v] then
4:             Error : Negative-weight cycle detected
5:             Terminate the algorithm
6:         end if
7:     end for
8:     Return distance[], predecessor[]
9: end procedure
```

Explication :

1. **Étape 1 : Initialisation**
   - À cette étape, l'algorithme initialise :
     (a) Les distances à tous les sommets à l'infini (∞), ce qui implique que la distance du sommet source à tous les autres sommets est initialement considérée comme infinie.
     (b) La distance au sommet source à 0, indiquant que le plus court chemin du sommet source à lui-même est de zéro.
   - De plus, il initialise un tableau pour suivre les prédécesseurs de chaque sommet.
2. **Étape 2 : Relaxation**
   - Pour chaque sommet, relâcher toutes les arêtes. Cela signifie vérifier s'il existe un chemin plus court vers chaque sommet en passant par un autre sommet, et mettre à jour en conséquence.
   - Mathématiquement, pour une arête (u,v) avec un poids weight(u,v), nous vérifions si : distance[v] > distance[u] + weight(u,v), où :
     - distance[u] : Correspond à la distance actuellement connue la plus courte depuis la source jusqu'au sommet u
     - distance[v] : Correspond à la distance actuellement connue la plus courte depuis la source jusqu'au sommet v.
     - weight(u,v] : Correspond au poids de l'arête reliant le sommet u au sommet v.
   - Si la condition est vraie : Cela signifie que passer par le sommet u depuis la source pour atteindre le sommet v offre un chemin plus court. Par conséquent, vous mettez à jour la distance vers le sommet v et définissez le prédécesseur de v comme étant u (indiquant que le chemin le plus court vers v passe actuellement par u) :
     - distance[v] = distance[u] + weight(u,v)
3. **Étape 3 : Vérification des cycles de poids négatif**
   - Enfin, l'algorithme vérifie la présence de cycles de poids négatif.
   - S'il trouve un chemin plus court même après |V|-1 itérations, cela indique la présence d'un cycle de poids négatif.

10.3.4 Exemple

On considère le graphe représenté dans la Figure 10.1 avec 5 sommets (A, B, C, D, E) et des arêtes pondérées. En utilisant l'algorithme de Bellman-Ford, on va essayer de trouver les chemins les plus courts du sommet A vers tous les autres sommets.

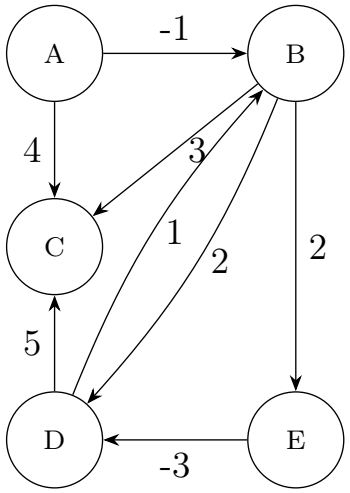


FIGURE 10.1 – Exemple

Initialisation :

— Définir la distance vers le sommet source A à 0.

— Définir la distance vers tous les autres sommets à l'infini (∞)

TABLEAU 10.1 – Initialisation.

Sommet	Distance	Prédécesseur
A	0	-
B	∞	-
C	∞	-
D	∞	-
E	∞	-

Relaxation :

— **Première itération (Relaxation de toutes les arêtes) :**

1. Arête (A, B, -1)
   — distance[B]=min(∞,0-1)=-1
   — Le prédécesseur de B est A.

2. Arête (A, C, 4) :
   — distance[C]=min(∞,0+4)=4
   — Le prédécesseur de C est A.
3. Arête (B, C, 3) :
   — distance[C]=min(4,-1+3)=2
   — Le prédécesseur de C est B.
4. Arête (B, D, 2) :
   — distance[D]=min(∞,-1+2)=1
   — Le prédécesseur de D est B.
5. Arête (B, E, 2) :
   — distance[E]=min(∞,-1+2)=1
   — Le prédécesseur de E est B.
6. Arête (D, B, 1) :
   — distance[B]=min(-1,1+1)= -1 (pas de mise à jour)
7. Arête (D, C, 5) :
   — distance[C]=min(2,1+5)=2 (pas de mise à jour)
8. Arête (E, D, -3) :
   — distance[D]=min(1,1-3)=-2
   — Le prédécesseur de D est E.

TABLEAU 10.2 – Relaxation : 1ère itération.

Sommet	Distance	Prédécesseur
A	0	-
B	-1	A
C	2	B
D	-2	E
E	1	B

— **Deuxième itération (Relaxation de toutes les arêtes) :**

1. Arête (A, B, -1)
   — distance[B]=min(-1,0-1)=-1 (aucune mise à jour)
2. Arête (A, C, 4) :
   — distance[C]=min(2,0+4)=2 (aucune mise à jour)
3. Arête (B, C, 3) :

— distance[C]=min(2,-1+3)=2 (aucune mise à jour)

4. Arête (B, D, 2) :
   — distance[D]=min(-2,-1+2)=-2 (aucune mise à jour)
5. Arête (B, E, 2) :
   — distance[E]=min(1,-1+2)=1 (aucune mise à jour)
6. Arête (D, B, 1) :
   — distance[B]=min(-1,-2+1)=-1 (aucune mise à jour)
7. Arête (D, C, 5) :
   — distance[C]=min(2,-2+5)=2 (aucune mise à jour)
8. Arête (E, D, -3) :
   — distance[D]=min(-2,1-3)=-2 (aucune mise à jour)

TABLEAU 10.3 – Relaxation : 2^e itération.

Sommet	Distance	Prédécesseur
A	0	-
B	-1	A
C	2	B
D	-2	E
E	1	B

— **En relaxant toutes les arêtes trois fois, le tableau ressemble à ceci (voir Tableau 10.4) :**

TABLEAU 10.4 – Relaxation : 3^e itération.

Sommet	Distance	Prédécesseur
A	0	-
B	-1	A
C	2	B
D	-2	E
E	1	B

— **Relaxation de toutes les arêtes une dernière fois (voir Tableau 10.5) :**

— **Vérification des Cycles de Poids Négatifs :**

Tableau 10.5 – Relaxation : 4^e itération.

Sommet	Distance	Prédécesseur
A	0	-
B	-1	A
C	2	B
D	-2	E
E	1	B

Dans un graphe sans cycles de poids négatifs, après $|V|$-1 itérations, les distances se stabiliseront. Si nous effectuons une autre itération et découvrons qu'une distance peut encore être réduite, cela indique la présence d'un cycle de poids négatif.

— **Relaxer toutes les arêtes une fois de plus (voir Tableau 10.6) :**

Tableau 10.6 – Relaxation : 5^e itération.

Sommet	Distance	Prédécesseur
A	0	-
B	-1	A
C	2	B
D	-2	E
E	1	B

10.3.5 Complexité Temporelle

La complexité temporelle de l'algorithme de Bellman-Ford est $\mathcal{O}(V * E)$, où V est le nombre de sommets et E est le nombre d'arêtes dans le graphe :

— **Initialisation :** Le réglage des distances initiales prend un temps de $\mathcal{O}(V)$.

— **Relaxation :** L'algorithme relaxe toutes les arêtes |V| - 1 fois, ce qui donne une complexité temporelle de $\mathcal{O}(V * E)$ pour cette étape.

— **Vérification des Cycles Négatifs :** Dans la dernière étape, l'algorithme vérifie la présence de cycles de poids négatif en itérant à travers toutes les arêtes, ce qui prend un temps de $\mathcal{O}(E)$.

Par conséquent, la complexité temporelle globale est dominée par l'étape de relaxation, menant à $\mathcal{O}(V * E)$.

10.4 Algorithme de Dijkstra

L'algorithme de Dijkstra est un algorithme glouton utilisé pour trouver le chemin le plus court à partir d'un seul sommet source vers tous les autres sommets d'un graphe avec des poids d'arête non négatifs.

10.4.1 Entrée de l'Algorithme

— Un graphe pondéré $G = (V, E, w)$ représenté par une liste ou une matrice d'adjacence, où V est un ensemble de sommets, E est un ensemble d'arêtes, et w représente les poids non négatifs des arêtes.

— Un sommet source s à partir duquel les chemins les plus courts vers tous les autres sommets doivent être calculés.

10.4.2 Sortie de l'Algorithme

— Un tableau de distances où $d[i]$ contient la distance la plus courte du sommet source s au sommet i.

— Un tableau de prédécesseurs qui aide à reconstruire le chemin le plus court du source à tout sommet.

10.4.3 Pseudocode

Algorithme 10.4 Dijkstra's Algorithm : Initialisation

```
1: procedure DIJKSTRA(graph, source)
2:     Input : graph, source
3:     Output : distance[], predecessor[]
4:     for each vertex v in graph do
5:         distance[v] = ∞                          ▷ Initialize distances to infinity
6:         predecessor[v] = null                    ▷ No predecessors yet
7:     end for
8:     distance[source] = 0                         ▷ Distance from source to itself is 0
9:     Create a priority queue Q with all vertices, prioritized by distance
10: end procedure
```

Algorithme 10.5 Dijkstra's Algorithm : Boucle principale

```
1: procedure DIJKSTRA(graph, source)
2:     while Q is not empty do
3:         u = Extract_Min(Q)                    ▷ Vertex with the smallest distance
4:         for each neighbor v of u do
5:             alt = distance[u] + weight(u, v)
6:             if alt < distance[v] then
7:                 distance[v] = alt             ▷ Update shortest distance to v
8:                 predecessor[v] = u            ▷ Update predecessor of v
9:                 Decrease_Key(Q, v, alt)
10:            end if
11:        end for
12:    end while
13: end procedure
```

Algorithme 10.6 Dijkstra's Algorithm : Sortie finale

```
1: procedure DIJKSTRA(graph, source)
2:     Return distance[], predecessor[]
3: end procedure
```

10.4.4 Exemple

On considère le graphe reprenté dans la Figure 10.2 avec 5 sommets (A, B, C, D, E) et des arêtes pondérées. En utilisant l'algorithme de Dijkstra, on va essayer de trouver les chemins les plus courts du sommet A vers tous les autres sommets.

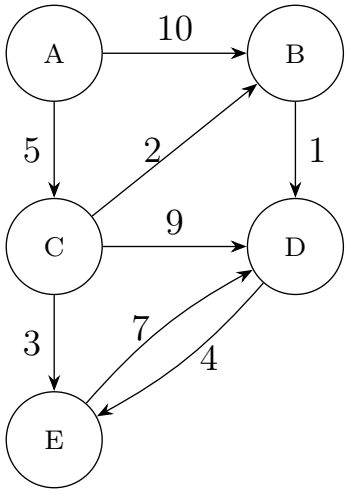


FIGURE 10.2 – Exemple.

— **La matrice d'adjacence :**

$$\mathbf{d} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 0 & 10 & 5 & \infty & \infty \\ \infty & 0 & \infty & 1 & \infty \\ \infty & 2 & 0 & 9 & 3 \\ \infty & \infty & \infty & 0 & 4 \\ \infty & \infty & \infty & 7 & 0 \end{pmatrix} \end{array}$$

1. **Initialisation :**
   - Définir les distances initiales de A à tous les sommets :
     - distance(A) = 0 (sommet de départ)
     - distance(B) = ∞
     - distance(C) = ∞
     - distance(D) = ∞
     - distance(E) = ∞
   - Définir l'ensemble traité S=$\emptyset$
   - Le prédécesseur de chaque sommet est inconnu au départ.
2. **Étape 1 (Traitement de A) :**
   - Sélectionner le sommet non traité avec la plus petite distance, qui est A avec une distance de 0.
   - Mettre à jour les distances des voisins de A (B et C) :
     - distance(B) = min(∞,0+10)=10
     - distance(C) = min(∞,0+5)=5
   - Prédécesseurs : B $\rightarrow$ A, C $\rightarrow$ A
   - Marquer A comme traité : S={A}.
3. Étape 2 (Traitement de C) :
   - Sélectionner le prochain sommet non traité avec la plus petite distance, qui est C avec une distance de 5.
   - Mettre à jour les distances des voisins de C (B, D, E) :
     - distance(B) = min(10,5+2)=7
     - distance(D) = min(∞,5+9)=14
     - distance(E) = min(∞,5+3)=8
   - Prédécesseurs : B $\rightarrow$ C, D $\rightarrow$ C, E $\rightarrow$ C
   - Marquer C comme traité : S={A,C}.
4. **Étape 3 (Traitement de B) :**
   - Sélectionner le prochain sommet non traité avec la plus petite distance, qui est B avec une distance de 7.

- Mettre à jour la distance du voisin de B (D) :
  - distance(D) = min(14,7+1)=8
- Prédécesseur : D → B
- Marquer B comme traité : S={A,C,B}.

5. **Étape 4 (Traitement de E) :**
   - Sélectionner le prochain sommet non traité avec la plus petite distance, qui est E avec une distance de 8.
   - Mettre à jour la distance du voisin de E (D) :
     - distance(D) = min(8,8+7)=8 (pas de mise à jour nécessaire)
   - Marquer E comme traité : S={A,C,B,E}.
6. **Étape 5 (Traitement de D) :**
   - Le dernier sommet non traité est D avec une distance de 8. Aucun voisin à mettre à jour.
   - Marquer D comme traité : S={A,C,B,E,D}.
7. **Distances finales les plus courtes depuis A :**
   - A vers A : 0
   - A vers B : 7
   - A vers C : 5
   - A vers D : 8
   - A vers E : 8

Remarque Importante :

- Ordre de traitement de D et E : On peut choisir l'un ou l'autre sommet dans n'importe quel ordre puisque leurs distances sont égales (8).
- Pour tout sommet dans l'algorithme de Dijkstra, si tous ses voisins sont déjà dans l'ensemble S (ce qui signifie qu'ils ont déjà été traités), il n'est pas nécessaire de mettre à jour leurs distances. En effet, le chemin le plus court vers chacun de ces voisins a déjà été déterminé lorsqu'ils ont été traités. Ainsi, revisiter ou mettre à jour ces voisins n'aboutirait à aucun chemin plus court.

10.4.5 Complexité temporelle

- $\mathcal{O}(V^2)$ avec une matrice d'adjacence
- $\mathcal{O}((E+V)\log V)$ en utilisant une liste d'adjacence avec une file de priorité en min-heap

10.5 Comment l'algorithme de Bellman-Ford est utilisé dans le RIP

Commençons par un petit rappel sur le protocole RIP (Routing Information Protocol) :

— Routage à vecteur de distance : RIP est basé sur l'algorithme de Bellman-Ford.

— Il s'agit d'un protocole de routage à vecteur de distance.

— Chaque routeur maintient une table de routage avec les meilleures distances connues pour chaque destination et le prochain saut pour atteindre cette destination.

— Mises à jour périodiques : Les routeurs partagent périodiquement leurs tables de routage avec leurs voisins (toutes les 30 secondes par défaut) ou en cas de changement important dans la topologie du réseau.

— Chaque routeur partage périodiquement l'intégralité de sa table de routage avec ses voisins.

— Nombre de sauts comme métrique : RIP utilise le nombre de sauts comme métrique pour mesurer la distance, avec un maximum de 15 sauts (pour éviter les boucles de routage).

Exemple :

Considérons un petit réseau de routeurs avec la topologie suivante (voir le graphe de la Figure 10.3) :

— A-B a un coût de 2

— A-C a un coût de 1

— B-D a un coût de 3

— C-D a un coût de 4

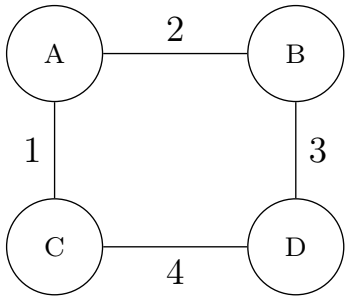


FIGURE 10.3 – Topologie du réseau.

1. **État Initial :**
   — Nous allons nous concentrer sur le routeur B et essayer de remplir sa table de routage.
   — Initialement, les routeurs A, B, C, et D ne connaissent que les connexions directes avec leurs voisins respectifs.

Tableau 10.7 – Table de A

Destination	**Coût**	**Prochain saut**
A	0	-
B	2	B (Direct)
C	1	C (Direct)
D	∞	-

Tableau 10.8 – Table de B

Destination	**Coût**	**Prochain saut**
A	2	A (Direct)
B	0	-
C	∞	-
D	3	D

Tableau 10.9 – Table de C

Destination	**Coût**	**Prochain saut**
A	1	A (Direct)
B	∞	-
C	0	-
D	4	D

Tableau 10.10 – Table de D

Destination	**Coût**	**Prochain saut**
A	∞	-
B	3	D
C	4	C
D	0	-

2. **Étape 1 :** Premier échange de tables. Réception des informations des voisins. Le routeur B recevra des mises à jour de ses voisins A et D sur leurs coûts respectifs pour atteindre d'autres routeurs.
   - Table de A :
     - Destination A : Coût 0, Prochain saut -
     - Destination B : Coût 2, Prochain saut B
     - Destination C : Coût 1, Prochain saut C
     - Destination D : Coût ∞, Prochain saut -
   - Table de D :

- Destination A : Coût ∞, Prochain saut -
- Destination B : Coût 3, Prochain saut B
- Destination C : Coût 4, Prochain saut C
- Destination D : Coût 0, Prochain saut -

3. **Étape 2 :** Le routeur B reçoit la table du routeur A et met à jour sa propre table. Le routeur B utilise l'algorithme de Bellman-Ford pour mettre à jour sa table en fonction des informations reçues. Il traite les informations du routeur A comme suit :

- Destination A :
  - Coût actuel dans la table de B : 2 (connexion directe à A)
  - Nouveau coût via A : 2 (coût de B à A) + 0 (coût de A à A) = 2
  - Pas de mise à jour nécessaire (coût déjà de 2 via le lien direct).
- Destination B :
  - Coût actuel dans la table de B : 0 (lui-même)
  - Nouveau coût via A : 2 (coût de B à A) + 2 (coût de A à B) = 4
  - Pas de mise à jour nécessaire (le coût de B à lui-même reste 0).
- Destination C :
  - Coût actuel dans la table de B : ∞ (initialement inconnu)
  - Nouveau coût via A : 2 (coût de B à A) + 1 (coût de A à C) = 3
  - Mise à jour : B apprend qu'il peut atteindre C via A avec un coût de 3.
- Destination D :
  - Coût actuel dans la table de B : 3 (connexion directe à D)
  - Nouveau coût via A : 2 (coût de B à A) + ∞ (coût de A à D) = ∞
  - Pas de mise à jour nécessaire (la connexion directe de B à D avec un coût de 3 reste meilleure).

Tableau 10.11 – Table de routage mise à jour pour B (Après avoir reçu la table de A)

Destination	**Coût**	**Prochain saut**
A	2	A
B	0	-
C	3	A
D	3	D

- **Étape 2 :** Le routeur B reçoit la table de D et met à jour sa propre table. Le routeur B utilise l'algorithme de Bellman-Ford pour mettre à jour sa table en fonction des informations reçues. Il traite les informations de D comme suit :
- Destination A :

  - Coût actuel dans la table de B : 2 (connecté directement à A)
  - Nouveau coût via D : 3 (coût de B à D) + ∞ (coût de D à A) = ∞
  - Aucune mise à jour nécessaire (la connexion directe de B à A avec un coût de 2 reste meilleure).
- Destination B :
  - Coût actuel dans la table de B : 0 (lui-même)
  - Nouveau coût via D : 3 (coût de B à D) + 3 (coût de D à B) = 6
  - Aucune mise à jour nécessaire (le coût de B à lui-même reste 0).
- Destination C :
  - Coût actuel dans la table de B : 2 (via A)
  - Nouveau coût via D : 3 (coût de B à D) + 4 (coût de D à C) = 7
  - B apprend qu'il peut atteindre C via D avec un coût de 7. (Pas de mise à jour, car $3 < 7$)
- Destination D :
  - Coût actuel dans la table de B : 3 (directement connecté à D)
  - Nouveau coût via D : 3 (coût de B à D) + 0 (coût de D à D) = 3
  - Aucune mise à jour nécessaire (le coût direct de 3 via D reste le même).

TABLEAU 10.12 – Table de routage mise à jour pour B (après avoir reçu la table de D)

Destination	**Coût**	**Prochain saut**
A	2	A
B	0	-
C	3	A
D	3	D

10.6 Convergence du RIP

- **Itération et Convergence :** Ce processus se répète à chaque routeur, et chaque routeur met continuellement à jour sa table en fonction des nouvelles informations reçues de ses voisins. Finalement, tous les routeurs du réseau convergeront vers un ensemble stable de chemins (c'est-à-dire que les tables de routage ne changeront plus).
- **Détection de la Convergence :** Le point de convergence est atteint lorsque les tables de tous les routeurs se stabilisent et ne sont plus mises à jour en fonction des vecteurs de distance échangés par les voisins.

10.7 Conclusion

Les algorithmes de Bellman-Ford, Dijkstra et Floyd-Warshall jouent un rôle fondamental dans l'optimisation du routage réseau. Chacun possède des caractéristiques uniques qui le rend adapté à des situations spécifiques, que ce soit en termes de complexité, de gestion des cycles négatifs ou de traitement des réseaux denses. L'exploration de leur implémentation dans des protocoles comme RIP, OSPF et BGP, ainsi que l'analyse de leurs impacts dans des environnements réseaux dynamiques, est fortement recommandée.

Chapitre 11

Protocoles ARP, RARP et ICMP

11.1 Introduction

Les protocoles ARP (Address Resolution Protocol), RARP (Reverse Address Resolution Protocol) et ICMP (Internet Control Message Protocol) sont des protocoles fondamentaux pour le fonctionnement quotidien des réseaux, assurant la résolution des adresses et la gestion des erreurs pour une communication fluide et fiable. Ce chapitre explorera en détail leur fonctionnement, leur utilisation, et leur importance dans les réseaux modernes.

11.2 Bref apercçu sur les protocoles ARP et RARP

— Les protocoles ARP et RARP fonctionnent à la **frontière** entre la couche de liaison de données (couche 2) et la couche réseau (couche 3) du modèle OSI, facilitant la traduction entre ces deux couches.

— Les deux protocoles sont cruciaux pour l'interaction entre la couche de liaison de données et la couche réseau, permettant une communication réseau fluide.

— L'ARP (Address Resolution Protocol, RFC 826) traduit les adresses IPv_4 en adresses MAC, permettant la communication au sein d'un réseau local.

— Le RARP (Reverse Address Resolution Protocol, RFC 903) était utilisé pour traduire les adresses MAC en adresses IPv_4, principalement pour les dispositifs sans stockage local (dispositifs sans disque).

— Le RARP est moins couramment utilisé aujourd'hui, ayant été remplacé par des protocoles plus avancés tels que BOOTP (Bootstrap Protocol) et DHCP (Dynamic Host Configuration Protocol).

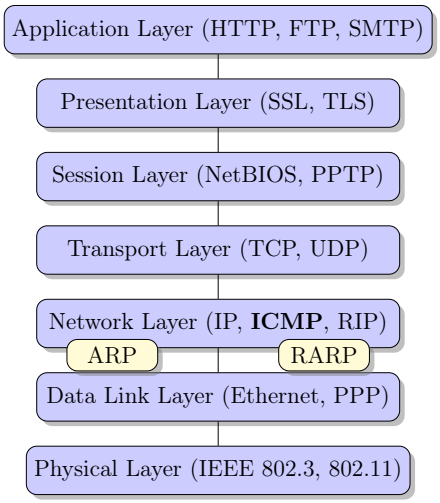


FIGURE 11.1 – Modèle OSI avec Exemples de Protocoles.

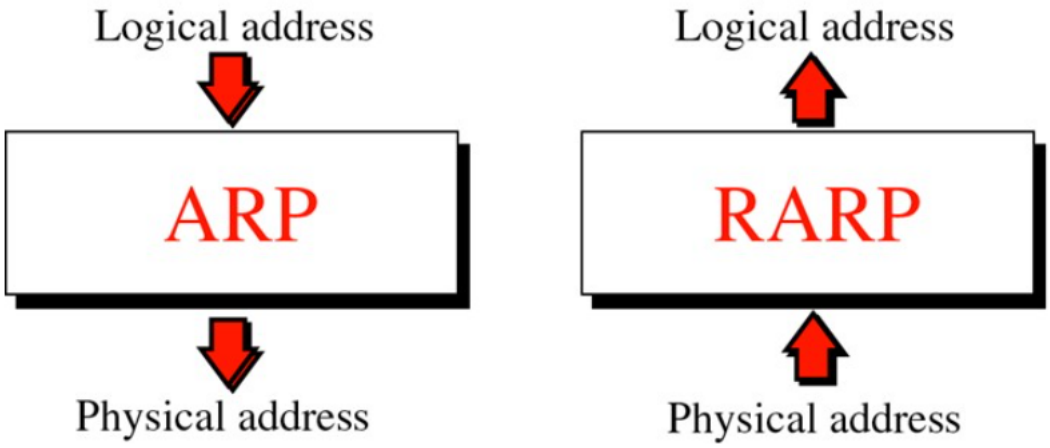


FIGURE 11.2 – Objectif de l'ARP et du RARP [28]

11.3 Protocole ARP

11.3.1 Objectif

L'ARP est utilisé pour mapper une adresse IPv_4 connue à une adresse MAC (Media Access Control) physique au sein d'un réseau local. Cela est nécessaire pour que les dispositifs puissent communiquer sur un réseau, notamment sur les réseaux Ethernet.

11.3.2 Comment fonctionne le protocole ARP ?

Lorsqu'un dispositif souhaite communiquer avec un autre dispositif sur le même réseau, il a besoin de l'adresse MAC correspondant à l'adresse IPv_4 du destinataire. ARP envoie une requête de diffusion à tous les dispositifs sur le réseau, demandant "Qui possède cette adresse

IP ?". Le dispositif avec l'adresse IPv_4 correspondante répond avec son adresse MAC, permettant au dispositif demandeur d'établir la communication.

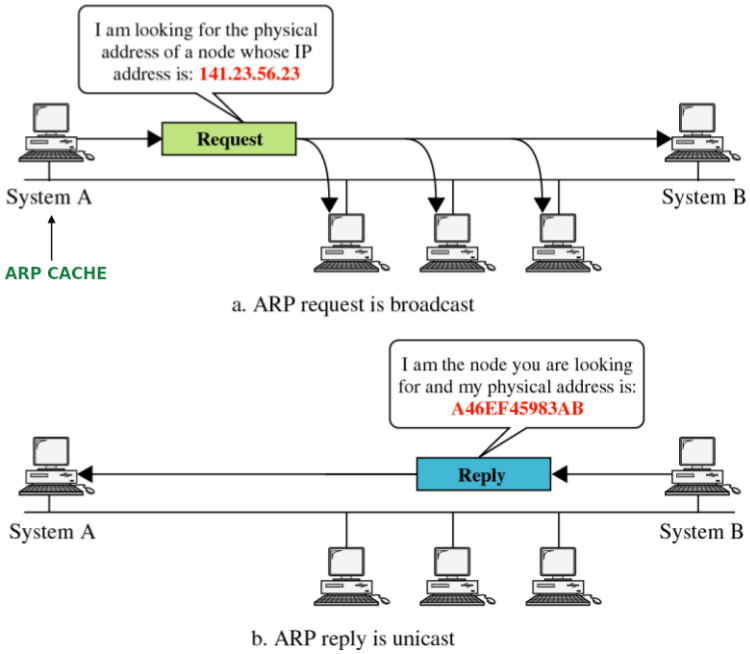


FIGURE 11.3 – Comment fonctionne le protocole ARP ? [28]

11.3.3 Exemple

Un hôte avec l'adresse IP 130.23.3.20 et l'adresse physique B2 :34 :55 :10 :22 :10 a un paquet à envoyer à un autre hôte avec l'adresse IP 130.23.43.25 et l'adresse physique A4 :6E :F4 :59 :83. Les deux hôtes se trouvent sur le même réseau Ethernet.

— Montrer les paquets de requête et de réponse ARP encapsulés dans des trames Ethernet.

11.3.4 Cache ARP

Afin d'éviter l'envoi répétitif de paquets de requêtes ARP, un hôte peut stocker les adresses IP et les adresses MAC correspondantes dans sa **table ARP** (ou cache ARP). Les entrées de cette table sont généralement sujettes à un processus de "vieillissement", ce qui permet d'effacer les informations si aucune activité n'est détectée pendant un certain temps. Lorsqu'un hôte reçoit une réponse ARP, il met à jour son cache ARP en conséquence.

Notons que le protocole ARP est un protocole sans état ; la plupart des systèmes d'exploitation mettront à jour leur cache si une réponse est reçue, indépendamment du fait qu'ils ont réellement envoyé une demande.

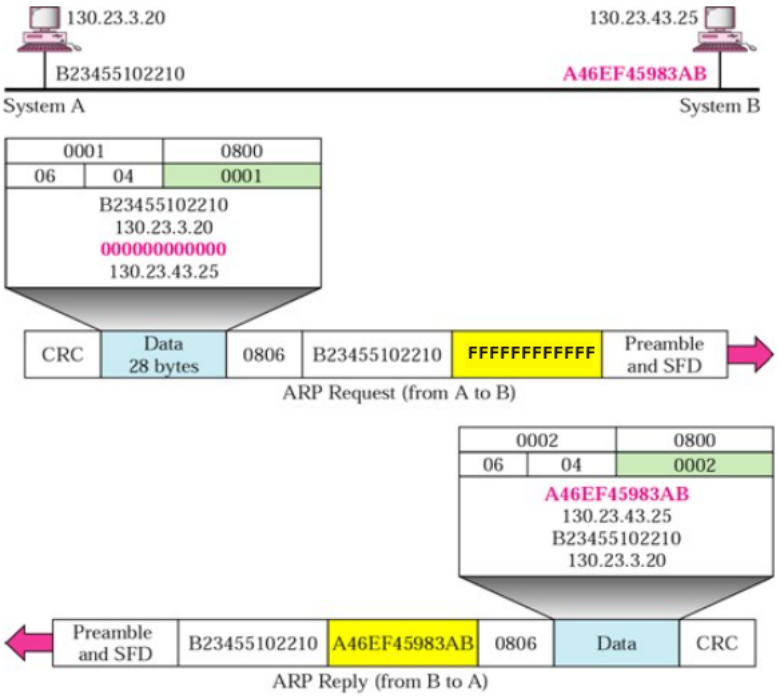


FIGURE 11.4 – Exemple [28]

11.3.5 Remarque

Il est à noter que le protocole IPv_6 utilise un protocole différent appelé Neighbor Discovery Protocol (NDP) pour des objectifs similaires à ARP.

11.3.6 Structure d'un paquet ARP

1. **Type de matériel (2 octets)**
   - Spécifie le type de matériel utilisé (par exemple, Ethernet est 1).
2. **Type de protocole (2 octets)**
   - Spécifie le type de protocole utilisé (par exemple, IPv4 est 0x0800).
3. **Longueur de l'adresse matérielle (1 octet)**
   - Indique la longueur de l'adresse matérielle (adresse MAC). Pour Ethernet, cette valeur est 6.
4. **Longueur de l'adresse du protocole (1 octet)**
   - Indique la longueur de l'adresse du protocole (adresse IP). Pour IPv4, cette valeur est 4.
5. **Opération (2 octets)**
   - Spécifie le type d'opération :
     - 1 pour une demande ARP

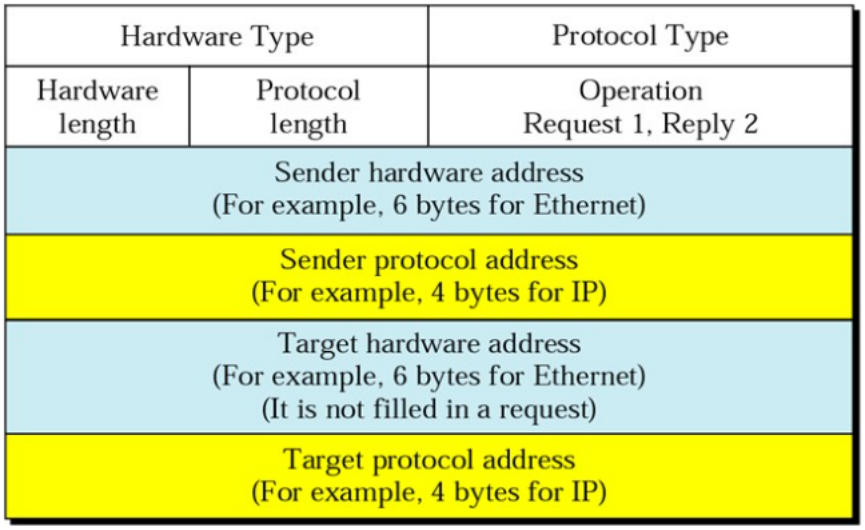


FIGURE 11.5 – Structure d'un paquet ARP [28]

— 2 pour une réponse ARP.

6. **Adresse matérielle de l'expéditeur (SHA) (6 octets)**
   — L'adresse MAC de l'expéditeur.
7. **Adresse du protocole de l'expéditeur (SPA) (4 octets)**
   — L'adresse IP de l'expéditeur.
8. **Adresse matérielle de la cible (THA) (6 octets)**
   — L'adresse MAC de la cible. Ce champ est réglé à 0 lors de l'envoi d'une demande ARP car l'expéditeur essaie de découvrir l'adresse MAC de la cible.
9. **Adresse du protocole de la cible (TPA) (4 octets)**
   — L'adresse IP de la cible.

11.3.7 Encapsulation d'un paquet ARP

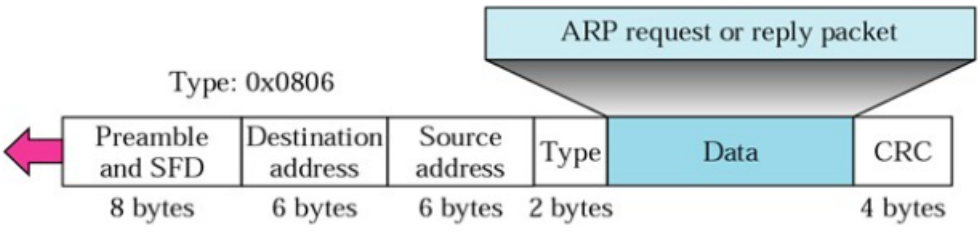


FIGURE 11.6 – Encapsulation paquet ARP [28]

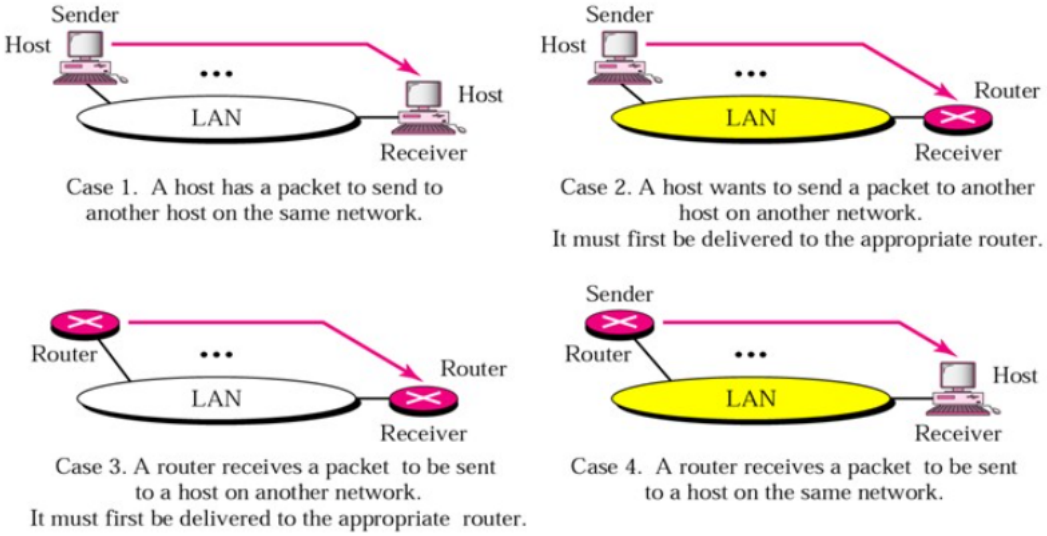


FIGURE 11.7 – Quatre cas d'utilisation d'ARP [28]

11.3.8 Cas d'utilisation d'ARP

11.3.9 Proxy ARP

Un Proxy ARP, ou parfois appelé ARP proxy, est un dispositif (tel qu'un routeur ou un serveur) qui répond aux requêtes ARP sur un réseau IP pour une ou plusieurs adresses IP qui ne sont pas réellement présentes sur ce réseau local. Cette technique permet à des dispositifs sur des sous-réseaux différents de communiquer comme s'ils étaient sur le même sous-réseau [34]. Cette technique est décrite dans la RFC 1027.

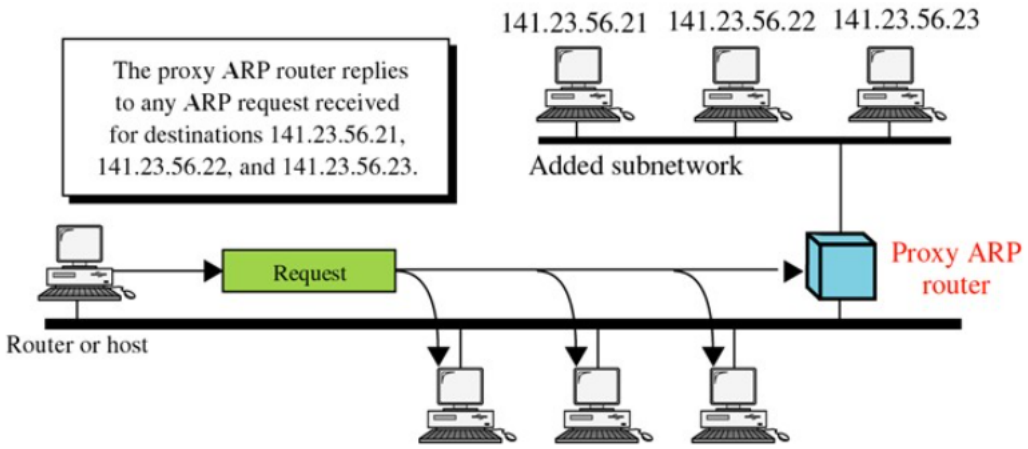


FIGURE 11.8 – Proxy ARP [28]

11.3.10 ARP spoofing

— Construire des réponses ARP usurpées

— Un ordinateur cible peut être convaincu d'envoyer des trames destinées à l'ordinateur A à la place vers l'ordinateur B

— L'ordinateur A ne se rendra pas compte que cette redirection a eu lieu.

— Ce processus de mise à jour du cache ARP de l'ordinateur cible est appelé "empoisonnement ARP".

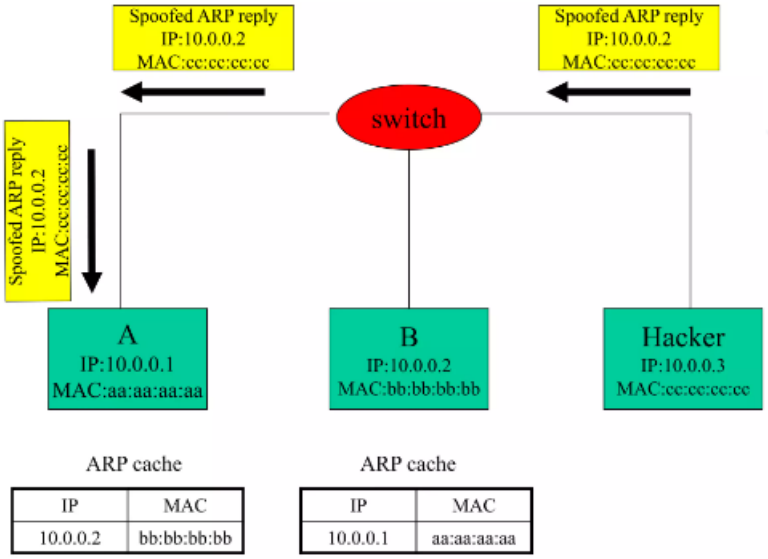


FIGURE 11.9 – ARP Spoofing (1) [29]

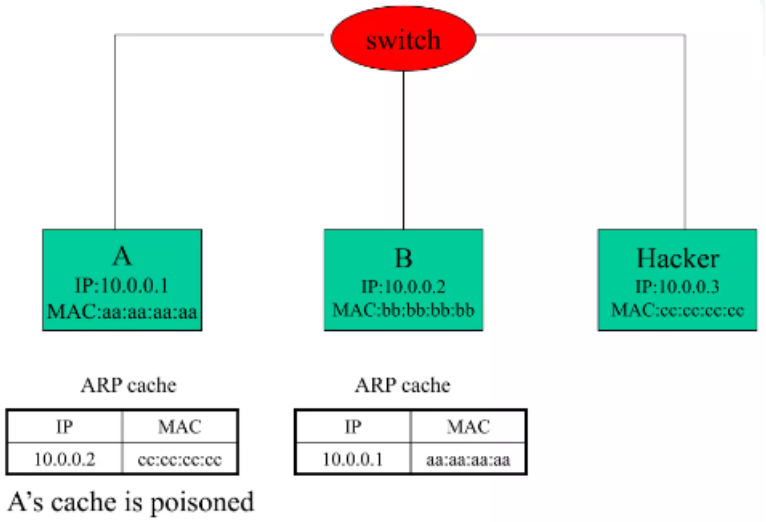


FIGURE 11.10 – ARP Spoofing (2) [29]

— Désormais, tous les paquets qu'A a l'intention d'envoyer à B iront à la machine du hacker.

— L'entrée du cache expirera, elle doit donc être mise à jour en renvoyant à nouveau la réponse ARP.

— À quelle fréquence ?

  — Cela dépend du système en question.

  — En général, 40 secondes devraient suffire.

11.4 Protocole RARP

11.4.1 Objectif

RARP effectue la fonction inverse d'ARP, en mappant une adresse MAC connue à une adresse IPv_4. Il est principalement utilisé par les stations de travail sans disque ou les dispositifs qui ne possèdent pas d'adresse IP stockée.

11.4.2 Comment fonctionne le protocole RARP ?

Un dispositif avec une adresse MAC connue envoie une requête RARP au réseau, demandant son adresse IPv_4 correspondante. Un serveur RARP sur le réseau, qui maintient une table des correspondances MAC vers IPv_4, répond avec l'adresse IPv_4 appropriée.

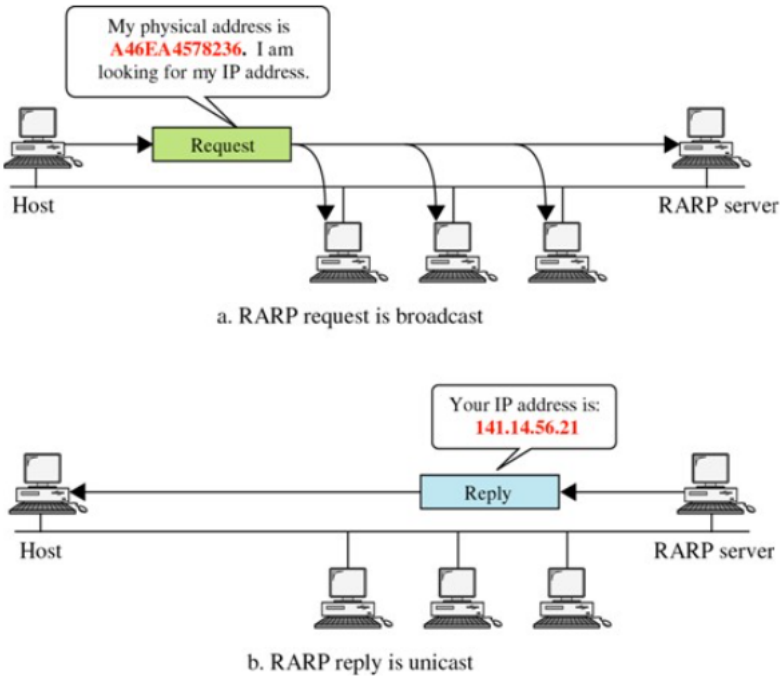


FIGURE 11.11 – Comment fonctionne le protocole RARP [28]

11.4.3 Structure d'un paquet RARP

La structure d'un paquet RARP est presque identique à celle d'un paquet ARP :

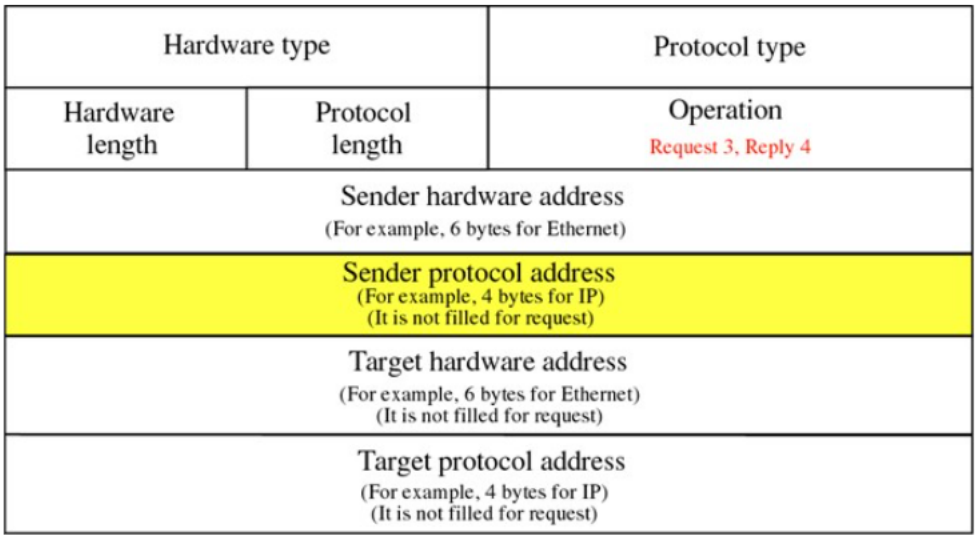


FIGURE 11.12 – Structure d'un paquet RARP [28]

1. **Type de matériel (2 octets)**
   - Identifie le type de matériel (par exemple, Ethernet est 1).
2. **Type de protocole (2 octets)**
   - Identifie le type de protocole (par exemple, IPv4 est 0x0800).
3. **Longueur de l'adresse matérielle (1 octet)**
   - Longueur de l'adresse matérielle. Pour Ethernet, cette valeur est 6.
4. **Longueur de l'adresse de protocole (1 octet)**
   - Longueur de l'adresse de protocole. Pour IPv4, cette valeur est 4.
5. **Opération (2 octets)**
   - Spécifie le type d'opération :
     - 3 pour une requête RARP
     - 4 pour une réponse RARP
6. **Adresse matérielle de l'expéditeur (SHA) (6 octets)**
   - L'adresse MAC de l'appareil demandeur.
7. **Adresse de protocole de l'expéditeur (SPA) (4 octets)**
   - Généralement définie à 0 lors de la requête, car l'appareil demande son adresse IP.
8. **Adresse matérielle du destinataire (THA) (6 octets)**

— L'adresse MAC de l'appareil demandeur (identique à SHA). La requête RARP est envoyée par l'appareil pour découvrir son adresse IP.

— **Adresse de protocole du destinataire (TPA) (4 octets)**

— Définie à 0 dans la requête RARP, car l'appareil demande au serveur RARP de remplir ce champ avec l'adresse IP appropriée.

11.4.4 Encapsulation d'un paquet RARP

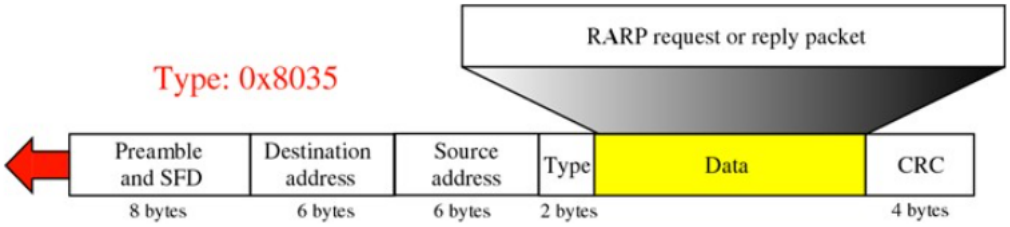


FIGURE 11.13 – Encapsulation d'un paquet RARP [28]

11.5 Protocole ICMP

11.5.1 Qu'est-ce que l'ICMP ?

Le protocole IP offre un service *non fiable*, ce qui signifie que si un paquet est perdu ou si une anomalie se produit au niveau IP, il ne fournit aucune information concernant la survenue, la nature ou l'origine de cette erreur.

Pour pallier cette faiblesse du protocole IP, les concepteurs ont introduit un protocole appelé ICMP (Internet Control Message Protocol) dans la suite de protocoles TCP/IP. La fonction principale de l'ICMP est de signaler à la station d'origine toute erreur pouvant survenir au niveau IP.

Ainsi, lorsque le protocole IP échoue dans certains cas à accomplir correctement la tâche qui lui est assignée, il informe le protocole ICMP, qui envoie un paquet à la station source pour lui notifier la nature et l'origine de l'erreur. Ce paquet est reçu par le protocole ICMP de la station source, qui informe ensuite le protocole IP de la survenue de cette erreur. Le protocole IP, désormais informé, peut agir en conséquence.

En résumé, on dira :

— ICMP, RFC 792

— Le rôle de l'ICMP est d'informer la source lorsqu'un problème est rencontré par un paquet, entraînant son élimination au cours de sa transmission à travers le sous-réseau.

— Il s'agit donc d'une fonction complémentaire à IP pour gérer les anomalies rencontrées par ce dernier.

— Ce protocole est requis dans tous les routeurs, ce qui explique pourquoi il est placé dans la couche IP.

Un message ICMP est envoyé à la source dans les cas suivants :

— Le paquet ne peut pas atteindre sa destination

— La durée de vie (TTL) du paquet a expiré

— Il y a un paramètre incorrect dans l'en-tête IP

— Congestion du routeur ou de l'entité de destination

— Erreur de routage local

— Expiration du minuteur de réassemblage IP

Aucun message ICMP n'est envoyé dans les cas suivants :

— Problèmes avec un paquet contenant un message ICMP

— Problèmes avec des paquets broadcast ou multicast

— Problèmes avec des fragments de paquets autres que le premier fragment

Remarques importantes :

— Le protocole ICMP est associé à la couche 3 (couche réseau), mais il est encapsulé par IP, comme s'il faisait partie de la couche 4.

— L'ICMP est effectivement un protocole de la couche réseau (couche 3) dans le modèle OSI, mais il est encapsulé dans des paquets IP, ce qui pourrait donner l'impression qu'il fonctionne comme un protocole de la couche transport (couche 4).

— Le protocole ICMP ne corrige pas les erreurs ; il se contente de transmettre des messages de contrôle.

11.5.2 Types et Codes de Message ICMP

ICMP définit plusieurs types de messages, les plus courants étant entre 0 et 18 comme le montre le Tableau 11.1 (les principaux types de messages ICMP pour IPv_4 (définis dans le RFC 792)).

11.5.2.1 Types d'Erreur ICMP

— Type 3 : Destination Unreachable

— Type 4 : Source Quench (Deprecated)

— Type 5 : Redirect

— Type 11 : Time Exceeded

— Type 12 : Parameter Problem

Tableau 11.1 – ICMP message types

Type	Description	Catégorie	Codes
0/8	Echo Request/Echo Reply	Informational Message	0
3	Destination unreachable	Error Message	0-15
4	Source quench	Error Message	0
5	Redirect	Error Message	0-3
9/10	Router advertisement/Router Solicitation	Informational Message	0
11	Time exceeded	Error Message	0-1
12	Parameter problem	Error Message	0
13/14	Timestamp Request/Timestamp Reply	Informational Message	0
17/18	Address Mask Request/Address Mask Reply	Informational Message	0

11.5.2.2 Types Informatifs ICMP

— Type 0 : Echo Reply

— Type 8 : Echo Request

— Type 9 : Router Advertisement

— Type 10 : Router Solicitation

— Type 13 : Timestamp Request

— Type 14 : Timestamp Reply

— Type 17 : Address Mask Request

— Type 18 : Address Mask Reply

Remarque importante :

— Type 15 : “Information Request” (initialement utilisé pour “address requests” ; désormais obsolète)

— Type 16 : “Information Reply” (obsolète)

11.5.3 Messages ICMP courants

— **Destination Unreachable :** Envoyé lorsqu’un paquet ne peut pas atteindre sa destination.

— **Time Exceeded :** Indique que la durée de vie (TTL) d’un paquet a expiré.

— **Redirect :** Informe un hôte d’une route plus efficace pour envoyer le trafic.

— **Echo Request/Reply :** Utilisé pour des outils de diagnostic comme ping pour tester la connectivité.

Le Tableau 11.2 montre des types et codes de messages ICMP courants.

Tableau 11.2 – Types et codes de messages ICMP courants

Type	Code	Description
0	0	Echo Reply (response to a ping request)
3	0	Destination Unreachable-Network Unreachable
3	1	Destination Unreachable-Host Unreachable
3	2	Destination Unreachable-Protocol Unreachable
3	3	Destination Unreachable-Port Unreachable
3	4	Fragmentation Needed and DF set
4	0	Source Quench (congestion control, deprecated)
5	0	Redirect-Network Redirect
5	1	Redirect-Host Redirect
8	0	Echo Request (ping)
11	0	Time Exceeded-TTL Expired in Transit
11	1	Time Exceeded-Fragment Reassembly Time Exceeded
12	0	Parameter Problem-Pointer indicates the error
12	1	Parameter Problem-Missing required option

11.6 Structure d'un message ICMP

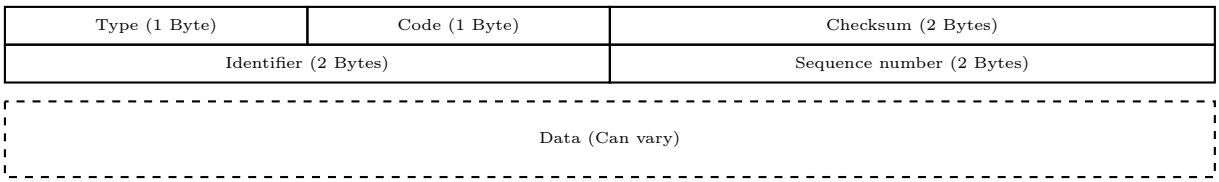


Figure 11.14 – En-tête ICMP (8 octets) + Données (peuvent varier)

La structure générale d'un message ICMP est la suivante :

— **Type (8 bits) :** Indique le type de message ICMP (par exemple, 0 pour Echo Reply, 3 pour Destination Unreachable).

— **Code (8 bits) :** Fournit un contexte supplémentaire pour le type (par exemple, pour Destination Unreachable, le code 0 indique réseau injoignable).

— **Checksum (16 bits) :** Utilisé pour vérifier les erreurs dans le message ICMP afin d'assurer l'intégrité des données.

— **Reste de l'en-tête (32 bits) :** Varie en fonction du type et du code du message. Ce champ peut contenir des informations telles que :

  - **Identifiant (16 bits) :** Utilisé pour associer les requêtes et les réponses (utilisé dans Echo Request/Reply).
  - **Numéro de séquence (16 bits) :** Utilisé pour suivre la séquence des requêtes et des réponses (utilisé dans Echo Request/Reply).
- **Données :** Contient des informations supplémentaires selon le type de message. Par exemple, une requête Echo inclut une charge utile (les données envoyées dans le ping).

Exemple pour Echo Request/Reply :

- **Type :** 8 (pour Echo Request) ou 0 (pour Echo Reply)
- **Code :** 0
- **Checksum :** Calculé pour la détection des erreurs
- **Identifiant :** Utilisé pour associer les requêtes et les réponses
- **Numéro de séquence :** Utilisé pour suivre la séquence
- **Données :** Champ optionnel contenant la charge utile

Remarque importante :
Pour IPv_6, le protocole ICMP est remplacé par ICMPv6.

11.7 Conclusion

Ce chapitre a permis d'explorer trois protocoles essentiels à la communication sur les réseaux IP, à savoir les protocoles ARP, RARP et ICMP. ARP et RARP facilitent la traduction des adresses entre les couches réseau et liaison de données, tandis que l'ICMP joue un rôle crucial dans la gestion des erreurs et la transmission d'informations de contrôle. Ces trois protocoles fondamentaux contribuent à assurer le bon fonctionnement et l'efficacité des échanges de données dans un environnement réseau.

Chapitre 12

Protocole DHCPv_4

12.1 Introduction

Dans ce chapitre, nous allons présenter le protocole DHCPv_4 (Dynamic Host Configuration Protocol). Ce protocole est essentiel pour la gestion dynamique des adresses IPv_4 dans les réseaux. Nous explorerons son fonctionnement, ses messages clés, et son rôle crucial dans la configuration automatique des dispositifs réseau.

12.2 Protocole DHCP

Le protocole de configuration dynamique des hôtes (DHCP) est un protocole de gestion de réseau utilisé sur les réseaux IP pour attribuer automatiquement des adresses IP et d'autres paramètres de configuration aux dispositifs (clients) connectés au réseau. En automatisant le processus d'attribution des adresses IP, le DHCP simplifie ainsi la gestion des grands réseaux et réduit le risque de conflits d'adresses.

12.3 Dans quelle couche opère le DHCP ?

Le DHCP appartient principalement à la couche Application (Couche 7), mais il interagit à la fois avec la couche Réseau (Couche 3) pour la gestion des adresses IP et avec la couche liaison de données (Couche 2) pour la communication initiale utilisant les adresses MAC lorsque l'IP n'est pas encore attribuée :

— **Couche Application (Couche 7) :** Le DHCP est un protocole de couche application qui fournit des services aux clients et aux serveurs pour la configuration automatique des adresses IP. C'est à ce niveau que la communication réelle entre le client DHCP et le serveur a lieu, en utilisant UDP (User Datagram Protocol) comme protocole de transport.

— **Couche Transport (Couche 4) :** Le DHCP utilise **UDP** comme protocole de transport, avec **le port 67 pour le serveur DHCP** et **le port 68 pour le client DHCP**.

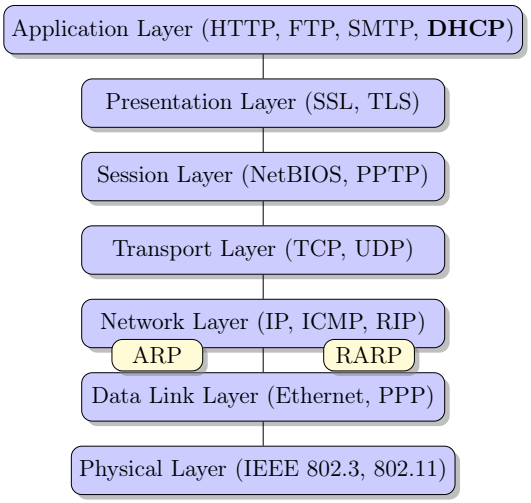


FIGURE 12.1 – Modèle OSI avec Exemples de Protocoles.

— **Couche Réseau (Couche 3) :** Étant donné que le DHCP est responsable de l'attribution et de la configuration des adresses IP, il interagit avec la Couche Réseau. Les adresses IP font partie de la Couche 3, et le DHCP fournit la configuration nécessaire pour que les dispositifs communiquent sur un réseau basé sur IP.

— **Couche Liaison de données (Couche 2) :** Aux premières étapes, lorsqu'un client DHCP n'a pas encore d'adresse IP, il communique avec le serveur en utilisant des messages de diffusion au niveau de la Couche Liaison de données (par exemple, en utilisant des adresses MAC pour la communication sur un réseau Ethernet).

Pour récapituler, on dira que :

— Le protocole DHCP permet aux nœuds de récupérer des paramètres de configuration depuis un serveur de configuration central.

— Une liaison fait référence à un ensemble de paramètres de configuration, comprenant au moins une adresse IP, qui sont associés ou attribués à un client DHCP.

— Les liaisons sont gérées par les serveurs DHCP.

— Les liaisons sont généralement valides seulement pour une durée limitée, appelée bail.

— La norme RFC 2131 précise les détails et les formats des messages pour le protocole DHCP.

— La norme RFC 3118 traite des considérations de sécurité liées au DHCP, en mettant particulièrement l'accent sur l'absence de mécanismes d'authentification dans le protocole.

12.4 Types de Messages DHCPv_4

La communication entre le client et le serveur suit quatre étapes principales, souvent mémorisées par l'acronyme **DORA** :

— Le message **DHCPDISCOVER** est un message de diffusion envoyé par les clients DHCP pour localiser les serveurs DHCP.

— Le message **DHCPOFFER** est envoyé par un serveur DHCP pour proposer un ensemble de paramètres de configuration à un client.

— Le message **DHCPREQUEST** est envoyé par le client au serveur DHCP en réponse à un précédent message **DHCPOFFER**, pour confirmer une liaison précédemment attribuée ou pour prolonger la durée de validité d'une liaison.

— Le message **DHCPACK (Accusé de Réception DHCP)** est envoyé par un serveur DHCP avec des paramètres supplémentaires au client en réponse positive à un **DHCPREQUEST**.

— Le message **DHCPNAK (Accusé de Réception Négatif DHCP)** est envoyé par un serveur DHCP pour informer le client que l'adresse IP proposée ou l'adresse IP demandée n'est plus valide. Par exemple :

  — **Offre :** Le serveur propose l'adresse IP 192.168.1.10 au client.

  — **Demande :** Le client accepte l'offre et envoie un DHCPREQUEST pour 192.168.1.10.

  — **Accusé de Réception Négatif :** Si, avant de traiter le DHCPREQUEST, le serveur découvre que 192.168.1.10 n'est plus disponible (peut-être a-t-elle été réattribuée à un autre client), il envoie un DHCPNAK au client. Le client devra alors recommencer pour obtenir une nouvelle adresse IP.

— Le message **DHCPDECLINE** est envoyé par le client au serveur DHCP pour indiquer que l'adresse IP proposée (ou une adresse IP qui a été assignée au client) est déjà utilisée sur le réseau. Le client utilise ce message s'il détecte un conflit d'adresse IP (par exemple, par ARP ou une autre méthode de détection de conflits réseau) et informe le serveur que l'adresse IP n'est pas appropriée, afin que le serveur propose une autre adresse.

— Le message **DHCPINFORM** est envoyé par un client DHCP à un serveur DHCP pour demander des informations de configuration sans avoir besoin d'obtenir une adresse IP. Ce message est utilisé lorsqu'un client a déjà une adresse IP (peut-être assignée statiquement) et a seulement besoin d'informations de configuration supplémentaires, telles que des serveurs DNS ou des passerelles par défaut. Le serveur répond avec un message DHCPACK contenant les informations de configuration demandées.

— Le message **DHCPRELEASE** est envoyé par un client DHCP pour informer le serveur DHCP que les paramètres de configuration ne sont plus utilisés. Aucun Accusé de Réception Explicite Requis : Le serveur n'est pas obligé d'envoyer un accusé de réception au client.

12.5 Bail DHCP

Pour prolonger le bail d'une liaison, DHCP attribue des adresses IP aux clients pour une période limitée, appelée le temps de bail. Si un client utilise encore l'adresse IP et en a besoin pour une période plus longue, il enverra un message DHCPREQUEST au serveur pour demander une prolongation du bail. Cela permet au client de continuer à utiliser la même adresse IP sans avoir besoin d'en recevoir une nouvelle.

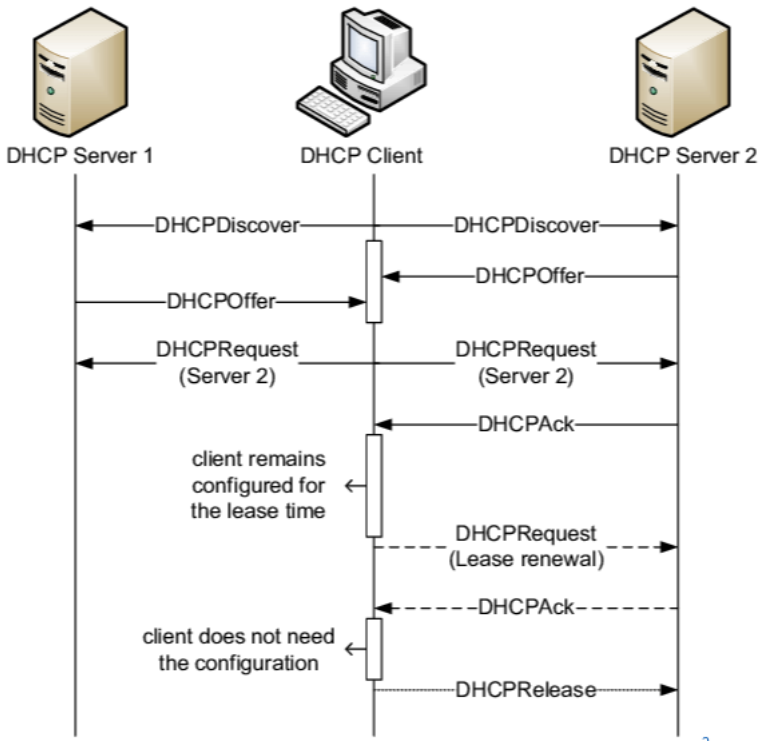


FIGURE 12.2 – Comment fonctionne le DHCP [30].

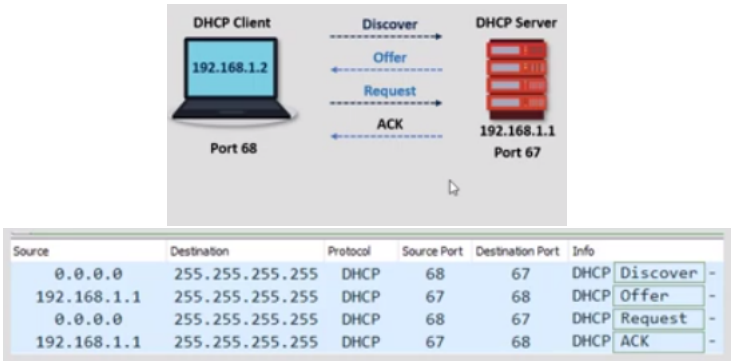


FIGURE 12.3 – Comment fonctionne le DHCP [31].

12.6 Options DHCP

En plus des adresses IP, DHCP peut fournir divers paramètres de configuration réseau, appelés options DHCP. Ceux-ci incluent :

— **Masque de Sous-Réseau :** Définit la taille du réseau et divise le réseau en sous-réseaux.

— **Passerelle par Défaut :** Spécifie le routeur que les clients utilisent pour accéder aux réseaux externes.

— **Serveurs DNS :** Fournit les adresses IP des serveurs DNS utilisés pour résoudre les noms

de domaine en adresses IP.

12.7 Conclusion

Ce chapitre a couvert les principaux aspects du DHCP, y compris les étapes de l'échange de messages, la gestion des baux, et les options de configuration fournies aux clients. Comme nous l'avons mentionné tout au début du Chapitre, grâce à l'automatisation du processus d'attribution des adresses IP, DHCP simplifie considérablement l'administration des réseaux et réduit le risque de conflits d'adresses.

Conclusion et perspectives

Conclusion

En résumé, ce polycopié sur l'"Architecture des Réseaux" offre une vue d'ensemble approfondie des concepts clés qui constituent la base des réseaux informatiques modernes.

Tout au long de ce document, nous avons exploré les principaux modèles de réseau, les protocoles de communication, le mécanisme du routage, ainsi que des techniques avancées telles que le NAT et les VLANs. Cela fournit une base solide pour les étudiants souhaitant approfondir leur compréhension de cette discipline en perpétuelle évolution.

La structuration en 12 chapitres a permis d'aborder de manière progressive les différents aspects de l'architecture des réseaux, facilitant ainsi l'apprentissage étape par étape. Chaque chapitre a été conçu pour développer les connaissances de manière cohérente, intégrant les fondements théoriques aux applications pratiques.

Perspectives

Pour les étudiants qui souhaitent poursuivre leur exploration, il est recommandé d'examiner plus en détail les domaines suivants, qui sont très prometteurs et ont révolutionné la discipline :

— **Les réseaux définis par logiciel (SDN) :** Découvrir comment ces nouvelles architectures influencent la gestion et l'évolution des réseaux modernes.

— **Les technologies de virtualisation des réseaux (NFV) :** Découvrez comment la virtualisation des fonctions réseau (VNF) et le découpage réseau (network slicing) transforment la gestion des infrastructures.

— **La sécurité des réseaux :** Appropriez-vous les techniques de sécurisation des communications et les mécanismes de protection contre les cybermenaces.

Bibliographie

[1] S Tanenbaum Andrew and J Wetherall David. *Computer networks*. Prentice Hall, 2011. i

[2] M. A. PERALDI-FRATI. La couche réseau : adresse ip. https://www.i3s.unice.fr/~map/Cours/LPSILADMIN/C3_Reseau_IP_CIDR_LPSIL.pdf, visité le 20.09.2022. ix, x, 6, 51, 52

[3] Sami Ahmed. Cisco world. https://www.youtube.com/watch?v=E_rqGiQhMC0, visité le 20.09.2022. ix, 13

[4] V.Mancuso G.Bianchi, G.Neglia. Subnetting & supernetting. https://dokumen.tips/documents/lecture-10-subnetting-supernetting-gneglia-vmancuso-outline-subnetting-variable.html?page=2, visité le 20.09.2022. ix, 16, 17

[5] Walid Saâd Eddine. Mega professional training. https://www.youtube.com/channel/UCBRqi3if1NrIgsbelhzwybA, visité le 20.09.2022. ix, x, 21, 23, 39, 68, 69

[6] Everest Academy. Routing protocol overview. https://www.youtube.com/watch?v=Qr7olj7MoiA, visité le 20.09.2022. ix, x, 21, 22, 25, 27, 28, 29, 92, 93

[7] André Vaucamps. *Cisco : protocoles et concepts de routage : configuration avancée des routeurs*. ENI, 2010. ix, 26, 28, 40

[8] Laurent SCHALKWIJK. Protocoles réseau v. https://www.irif.fr/~jch/enseignement/reseaux/reseaux5.pdf, visité le 23.02.2023. ix, 29, 31

[9] Monica Sharma. Advanced internet technologies :osi & tcp/ip. https://advancedinternettechnologies.wordpress.com/ipv4-header/, visité le 14.11.2023. ix, 45

[10] Yeda Center. Ip header - tos field. https://www.youtube.com/watch?v=WpL5wy3SuO0, visité le 14.11.2023. ix, 45

[11] AbdelAli ED-DBALI. Les réseaux : Couche réseau. https://www.univ-orleans.fr/lifo/Members/duchier/teaching/A1-ASR-Reseau/Ch3.pdf, visité le 20.09.2022. x, 52

[12] ipwithease. Ipwithease. https://ipwithease.com/what-is-a-mac-address/, visité le 14.11.2023. x, 62

[13] ciscopress. mplementing cisco ip switched networks (switch) foundation learning guide : Campus network architecture. https://www.ciscopress.com/articles/article.asp?p=2348266, visité le 11.07.2024. x, 66

[14] Cisco. Configuration de base des réseaux locaux virtuels. https://www.exoco-lmd.com/dlattach/?attach=7748, visité le 18.05.2024. x, 71

[15] Réussirsonccna. Réussir son ccna. https://reussirsonccna.fr/trunk-802-1q-et-isl-ce-quil-faut-savoir-pour-le-ccna/, visité le 11.07.2024. x, 72

[16] Sameh Ben Fredj & Philippe Antoine. Introduction à l'iot : Du capteur à la donnée. https://dokumen.tips/technology/introduction-a-liot-du-capteur-a-la-donneepresentation-mix-it2015.html?page=12, visité le 24.11.2022. x, 80

[17] Polytech Nantes Benoît Parrein. Réseaux, internet et routage. https://pagesperso.ls2n.fr/~parrein-b/courses/srt3A-reseaux-parrein.pdf, visité le 24.11.2022. x, xiii, 81, 86

[18] Adel Al Humaidi. Understanding ipv_6. https://www.youtube.com/playlist?list=PLFjr7QsNng23iQMTKmkctBf8gN55JXnJz, visité le 24.11.2022. x, 85, 87, 88, 89, 90

[19] Reseaux Certa. Exolab mise en oeuvre ipv_6. https://www.reseaucerta.org/sites/default/files/Pratique-IPv6-DHCPv6-01-Sujet.pdf, visité le 24.11.2022. x, 91

[20] Asmaa Boughrara. Cours réseaux (l2), usto. https://www.youtube.com/watch?v=hylnvT4Ox9Q, visité le 20.09.2022. x, xi, 97, 100, 103, 109

[21] IT-Connect.fr. Plate-forme de cours sur l'administration systèmes et réseau pour les professionnels de l'informatique. https://www.it-connect.fr/, visité le 01.06.2024. xi, 100, 102

[22] Mohamed Herak. Réseaux. https://www.youtube.com/watch?v=lfj5usdvAJE&list=PLh330TDu4_HAye58MbG6Yto0MDEAd9Icp, visité le 04.07.2024. xi, 109, 110

[23] Licence MIASHS. Support de cours réseaux. https://www.gipsa-lab.grenoble-inp.fr/~christian.bulfone/MIASHS-L3/, visité le 31.05.2024. xi, 111

[24] Guy Pujolle Avec la contribution de Olivier Salvatori. Cours réseaux et télécoms avec exercices corrigés. https://www.univ-orleans.fr/lifo/membres/duchier/teaching/A1-ASR-Reseau/Ch3.pdf, visité le 28.07.2024. xi, 122

[25] AbdelAli ED-DBALI. Chapitre 4 : Couche réseau. https://www.univ-orleans.fr/lifo/membres/duchier/teaching/A1-ASR-Reseau/Ch3.pdf, Last visited on 28.07.2024. xi, 123

[26] ROBIN Eric et all. Ccna 4 - essentiel. réseaux et technologies wan. https://www.academia.edu/31012926/CCNA_4_Essentiel_pdf, Last visited on 28.07.2024. xi, 124

[27] FORMIP. The three types of network address translation. https://www.formip.com/pages/blog/network-address-translation-les-3-types, Last visited on 28.07.2024. xi, 124, 125

[28] Samuel Herington. Network layer protocols : Arp, ipv4, icmpv4, ipv6, and icmpv6. https://https://slideplayer.com/slide/3366508/, Last visited on 11.07.2024. xi, 144, 145, 146, 147, 148, 150, 151, 152

[29] Luthfi Widyanto. Arp spoofing. https://www.slideshare.net/slideshow/arp-spoofing/14730788, Last visited on 11.07.2024. xi, 149

[30] Shih-Ting Yuan. Behind the scenes of instance failures : The impact of dhcp lease expiration. https://shihtiy.com/posts/instance-status-check-system-time-dhcp/, Last visited on 20.09.2022. xi, 160

[31] Walid Saâd Eddine. Dhcp protocol. https://www.youtube.com/watch?v=i0_4xr8E9wc, Last visited on 10.09.2024. xi, 160

[32] Mohamed Rjoubi. Configuration d'un routeur. https://independent.academia.edu/MohamedRjoubi, visité le 24.11.2022. 23

[33] Walid Saâd Eddine. Vtp. https://www.youtube.com/watch?v=g0_xiCb7V2c&t=6s, visité le 14.07.2024. 74

[34] WikiPédia. Proxy arp. https://fr.wikipedia.org/wiki/Proxy_ARP, Last visited on 11.07.2024. 148