Cours Services Réseaux

Mise à jour 2014

Auteurs

Dr. Samuel OUYA, **LIRT-ESP** (Laboratoire Informatique Réseaux et Télécommunications Ecole Supérieure Polytechnique)

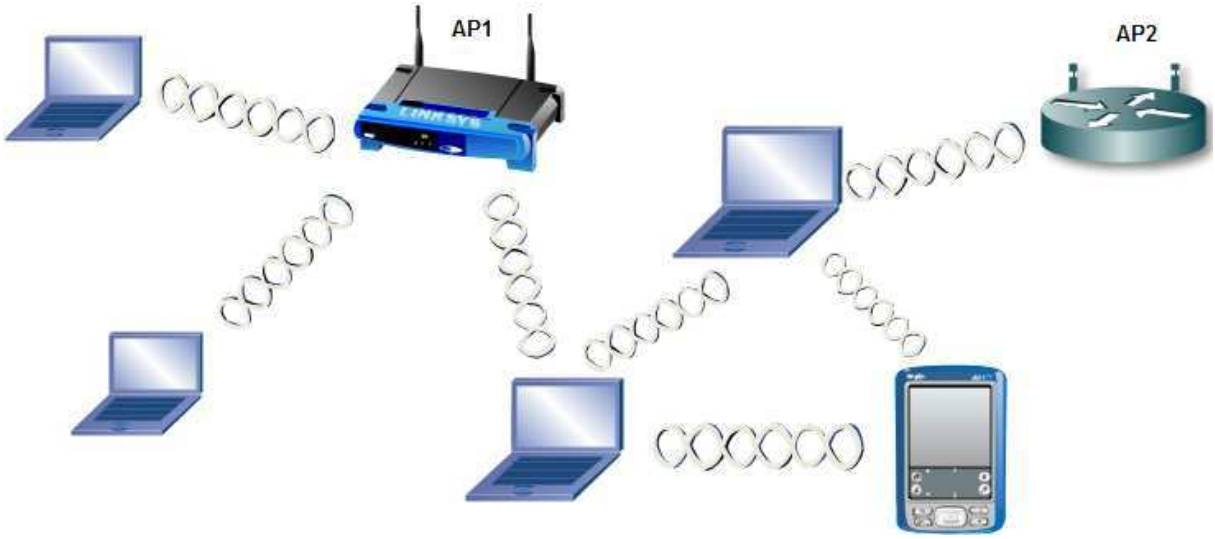
Yvan KALIA, **Laboratoire Réseaux & NGN EC2LT** (Ecole Centrale de Logiciels Libres et des Télécommunications)

Avant-propos

Le système d'information constituant la dorsale de l'entreprise regorge différentes applications au bon fonctionnement de celle-ci. Le contexte des applications et services sont généralement utilisés pour désigner un usage au sein de l'entreprise. Ce cours s'adresse aux étudiants en informatique et télécoms leur permettant de maîtriser les concepts des services déployés dans les réseaux d'entreprise. Ce document constitue un support de cours pour ces derniers et un mémento pour un administrateur système, ainsi que les ingénieurs systèmes. Le cours sera abordé sur deux environnements, sous Linux et également en environnement Cisco. Ce cours est organisé en chapitres afin d'appréhender les différents concepts des services déployés dans les réseaux d'entreprise. Des études de cas pour appuyer les scénarii.

Chapitre 1 : Introduction au réseau Wi-Fi

La norme 802.11 définie la norme des réseaux locaux sans fil dont les supports de communication sont basés sur les faisceaux hertziens. Un client wi-fi est immobile dans la cellule où il se trouve (une cellule est une zone géographique délimitée où l'on peut capter les signaux d'une antenne et émettre des signaux vers cette antenne). Les réseaux Wi-Fi travaillent à une vitesse de 11Mbits/s à 54Mbits/s avec une fréquence de 2,4GHz. Cette possibilité de communication par voie hertzienne donne lieu à des communications directe station à station, ou en passant par un point d'accès (AP, Access Point). Dans un réseau sans fil, pour qu'un signal soit reçu correctement, sa portée ne doit pas dépasser 50m (donnée théorique) dans un environnement de bureau ou 500m dans un vide (sans obstacle), soit sur plusieurs kilomètres avec une antenne directive. En générale, la station peut capter le signal sur 20m (dans les faits) dans un environnement de bureau.



Architecture d'un réseau Wi-Fi

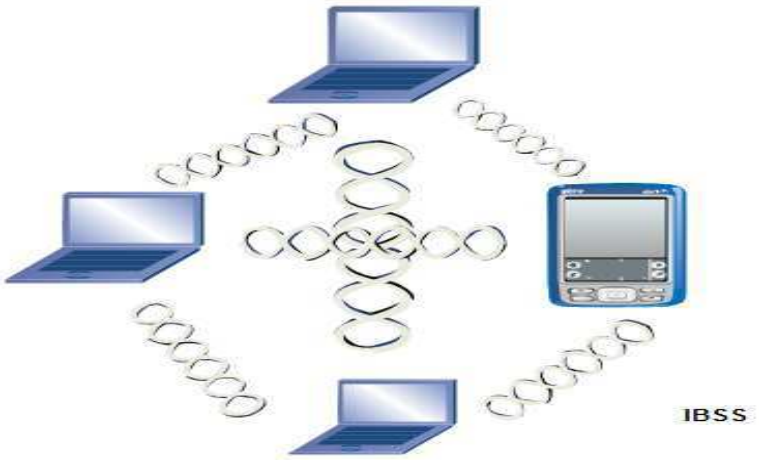
Les topologies des réseaux Wi-Fi

Les réseaux sans fil de type Wi-Fi exploitent deux modes de connexion :

- Le mode Ad-Hoc
- Le mode Infrastructure

Le mode Ad-Hoc

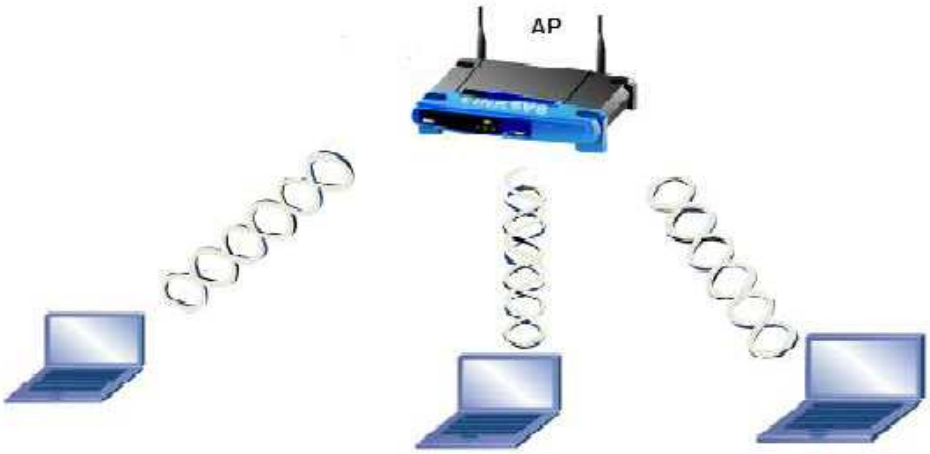
L'exploitation des voies hertziennes a été à l'origine des architectures des réseaux sans fil, donnant naissance à la notion des cellules. Un réseau Ad-Hoc est un ensemble de terminaux (stations) directement reliés entre eux formant une cellule appelé IBSS (Independent Basic Set Service). Le mode ad-hoc permet aux stations de communiquer sans l'intervention d'une infrastructure quelconque telle qu'un point d'accès. Chaque terminal peut ainsi établir une communication avec un autre dans un IBSS.



Mode AdHoc

Le mode Infrastructure

Un réseau en mode infrastructure est un ensemble de terminaux reliés entre eux par l'intermédiaire d'un AP. les communications entre les stations ne sont pas directes. Le mode infrastructure est utilisé dans un environnement de bureau. Ce mode est défini pour offrir aux différentes stations des services spécifiques sur une zonne donnée. L'ensemble composé par les stations et l'AP forme une cellule appelée BSS (Basic Set Service).



Mode Infrastructure

Etude de cas 1 : Mise en place d'un réseau Wi-Fi sous Linux

De manière générale, pour se connecter à un réseau Wi-Fi, il faut connaitre :

- Le nom du réseau (SSID)
- Le mode de fonctionnement de la radio

Il ya plusieurs modes de fonctionnement de la radio :

- Le mode AdHoc
- Le mode Manage
- Le mode Master
- Le mode Monitor

Cette étude de cas montre le déploiement d'un réseau Wi-Fi sous Linux à travers la ligne de commande. Les ordinateurs sont équipés à la fois d'une interface Ethernet et d'une interface supportant le Wi-Fi. Avant de mettre en place ce réseau, il faut s'assurer que votre ordinateur supporte le mode Wi-Fi. Linux dispose de l'outil ''iwconfig'' (interface wireless configuration) qui affiche les informations à propos de l'interface wifi. Voici un etrait du résultat de cette commande sur une station Linux.

```
# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11bg  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated  Tx-Power=off
          Retry  long limit:7  RTS thr:off Fragment thr:off
          Encryption key:off
Power Management:off
```

L'exécution de la commande ''iwconfig'' affiche les informations contenues dans les paramètres suivants :

- **IEEE 802.11bg** : Norme Wi-Fi supportant les technologies b et g
- **ESSID** (off/any) : Ce paramètre renseigne sur le nom du réseau (SSID)
- **Mode** : Le mode utilise
- **Access Point** (Associated/Not-Associated): il n'ya pas de point d'accès auquel la station a communiqué (Not-Associated)
- **Encryption key** (on/off): Paramètre indiquant si le réseau est protégé ou non.

Pour scruter les réseaux avoisinants, on peut se servir de l'outil ''iwlist''. La commande ''iwlist'' exécutée sur la station terminale, donne la liste des réseaux avoisinant ainsi que les puissances d'émissions. Un extrait de la commande ''iwlist'' est donné dans le tableau ci-dessous.

```
# iwlist wlan0 scan
wlan0     Scan completed :
          Cell 01 - Address: 64:70:02:F9:82:D6
                    Channel:6
                    Frequency:2.437 GHz (Channel 6)
                    Quality=70/70  Signal level=16 dBm
                    Encryption key:off
                    ESSID:"uvs"
                    Bit Rates:1 Mb/s; 2 Mb/s;5.5Mb/s; 11Mb/s; 6Mb/s
                              9 Mb/s; 12 Mb/s; 18 Mb/s
                    Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
                    Mode:Master
```

Nous allons maintenant configurer un réseau Wi-Fi en mode Ad Hoc pour deux stations terminales.

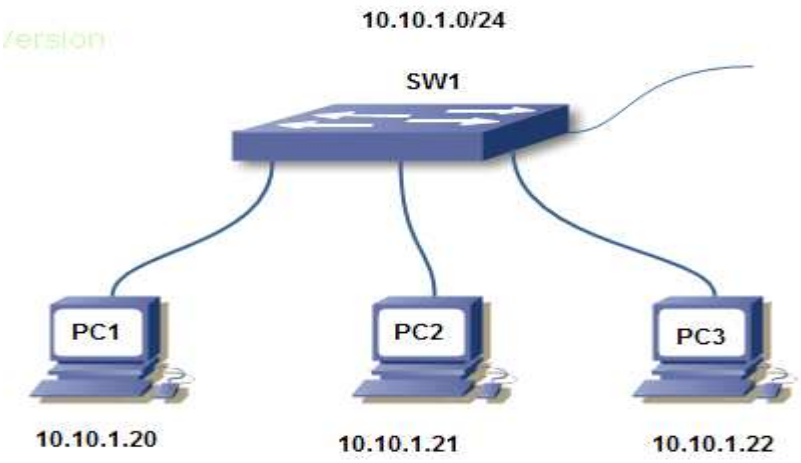
Nous avons de deux outils : iwconfig (pour créer le réseau) et ifconfig

Il faut tout d'abord commencer à désactiver l'interface wifi pour vous rassurer que votre station ne tente pas de se synchroniser avec un point d'accès dans.

```
# ifconfig wlan0 down
# iwconfig wlan0 mode AdHoc
# iwconfig wlan0 essid "UVS"
# iwconfig wlan  key 0123456789
# ifconfig wlan0 up
# ifconfig wlan0 192.168.3.20 netmask 255.255.255.0
```

Etude de cas 2 : Mise en réseau (Filaire)

Dans ce cas de figure, nous allons montrer comment mettre en réseau deux stations terminales sous Linux. Tous les paramétrages sont manuels. Nous allons considérer l'architecture suivante avec comme adresse réseau 10.10.1.0/24.



Nous allons configuer chacune des stations terminales en ligne c'est-à-dire leur donner les éléments TCP/IP (adresse IP, masque, passerelle, DNS, …).

Pour configuer less éléments TCP/IP sur une station terminale sous linux, on utilise la commande ‘’ifconfig’’. Cette commande prend en argument le nom de l’interface sur laquelle on veut fixer les éléments TCP/IP, l’adresse IP, et le masque de sous réseau. Elle peut s’employer sans option pour consulter toutes les interfaces présentes sur le système.

Extrait de configuration de la station terminale PC1.

```
# ifconfig –a
# ifconfig eth0 10.10.1.20 netmask 255.255.255.0
```

Le processus de configuration sur PC2 et PC3 sera le même à la limite de changer les adresse IP pour éviter un éventuel conflit. Après ses configurations, vous pouvez tester la connectivité par un ping.

Nous envisageons ajouter une route par défaut (passerelle par défaut). Pour ajouter une route par défaut, nous allons manœuvre la commande ‘’route’’ comme suit :

```
#
# route add default gw 10.10.1.254
```

Nous allons utiliser les ACLs pour interdire tout ping vers la station PC1

```
# iptables -A INPUT -p icmp -j REJECT
# iptables -D INPUT -p icmp -j REJECT  // cette regle annule la regle
precedente definie)
```

Toutes ces manipulations sont basiques et ne nécessite pas de connaissance approndies. Nous allons introduire les services réseaux. Quand on parle de services réseaux, on pense aux services qu’on déploie en réseau. Lorsqu’on service est déployé, c’est pour une utilisation précise. Mais la question pendante ici est comment se connecter à un service en réseau ?

Pour se connecter à un service en réseau, il faut préciser les paramètres suivants :

- L’adresse IP du service
- Le port d’écoute du service
- Le protocole de transport utilisé.

Mais en général, on ne précise pas tous ces paramètres pour accéder à un service. Prenons par exemple l'accès à un serveur web. L'accès à un service web se fait via le protocole http.

http://ip_serveur

Nous accédons au serveur alors qu'on n'a pas précisé tous les paramètres. Ceci fonctionne, car sous Linux il ya un fichier appelé ''/etc/services'' qui contient toutes les informations à propos des services, les ports d'écoute ainsi que les protocoles de transport utilisés.

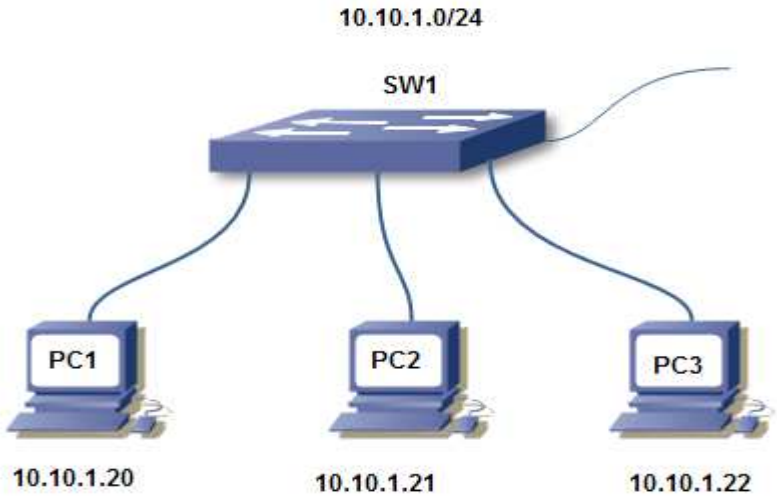
Extrait du fichier /etc/services

```
# less /etc/service

# service-name  port/protocol  [aliases ...]   [# comment]

ftp             21/tcp
ftp             21/udp          fsp fspd
ssh             22/tcp                          # The Secure Shell
(SSH) Protocol
ssh             22/udp                          # The Secure Shell
(SSH) Protocol
telnet          23/tcp
telnet          23/udp
```

Etude de cas 3 : Connexion à distante

La connexion à distance sur des systèmes UNIX/Linux se fait par le biais des protocoles ''Telnet'' ou SSH. Le protocole Telnet a montré ses limites (connexion non sécurisée) face aux mécanises de sécurité (envoie de mot de passe en clair), c'est pourquoi sous Linux, on empêche au super utilisateur (root) de se connecter à distance. Pour pallier aux insuffisances de telnet, SSH (Secure Shell) permet de se connecter à distance via un shell sécurisé. Nous allons montrer dans cette étude de cas comment se connecter sur un système distant par SSH.

Nous allons reconsidérer l'architecture de l'étude de cas 2 pour notre scénario.



Syntaxe

 # ssh nom_user@ip_disant

Nous utilisons le compte ‘’ec2lt’’ pour se connecter sur la station PC2 à partir de PC1

```
[root@localhost ~]# ssh ec2lt@10.10.1.21
```

Appliquons quelques règles d’ACLs pour empêcher toute connexion SSH sur PC2.

```
# iptables -A INPUT -p tcp --dport 22 -j REJECT
```

Observons ce qui suit lorsque PC1 tente une connexion par SSH sur PC2

```
[root@localhost~ # ssh ec2lt@10.10.1.21
ssh: connect to host 192.168.0.113 port 22: Connection refused
```

Nous allons interdire cette fois-ci la connexion à la station PC3

```
# iptables -A INPUT -p tcp --dport 22 -s 10.10.1.22 -j REJECT
```

Chapitre 2 : Les service DHCP

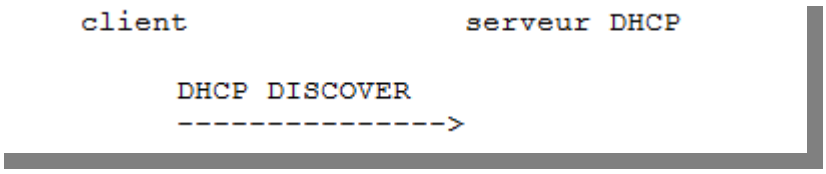
Le protocole DHCP (Dynamic Host Control Protocol), protocole défini dans les RFC 1531, 1534, 2131 et 2132 décrit les processus par lesquels une station terminale procède pour avoir les éléments TCP/IP et se mettre en réseau. DHCP écoute sur le port 67 (coté serveur) et sur le port 68 (coté client). Il est implémenté dans l'ancienne version du protocole (IPv4) et dans la version 6 appelé DHCPv6. L'intérêt d'utiliser le protocole est de permettre aux stations terminales d'obtenir les éléments TCP/IP de façon automatique et dynamique.

Fonctionnement du service DHCP

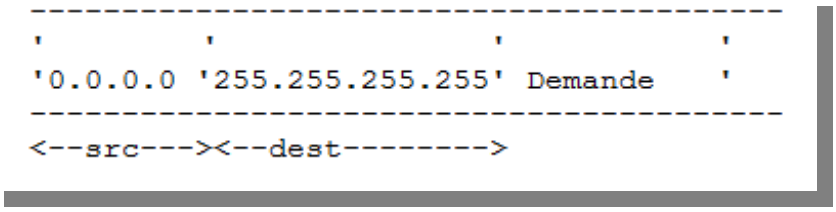
Pour comprendre le fonctionnement du service DHCP, nous allons étudier les scénarii suivants :

Scénarion1 : Première découverte du client

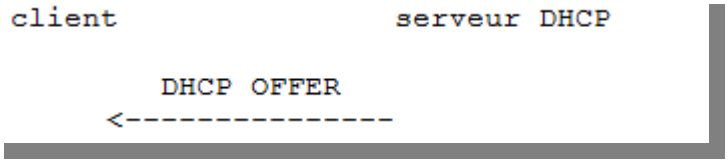
A sa première connexion sur le réseau, une station terminale émet des requêtes sur le réseau pour obtenir les éléments de connexion.



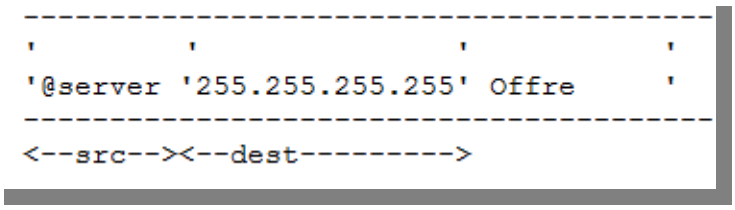
Un client envoie par diffusion sur le réseau un message appelé ''DHCPDISCOVER''. N'ayant pas encore d'éléments de connexion, ce message a pour adresse source 0.0.0.0 et pour adresse de destination 255.255.255.255. Voici le format d'un paquet DISCOVER



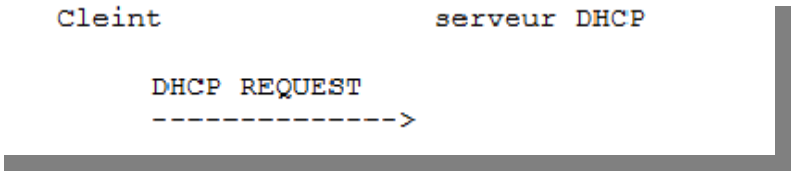
Scénario 2 : réponse du serveur



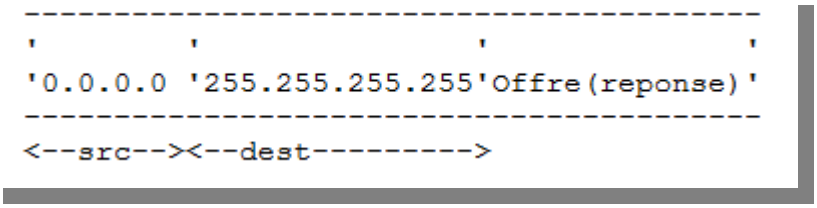
Un serveur DHCP se trouvant dan le réseau répond par diffusion avec un message DHCP OFFER.



Scénario 3 : Première réponse du client

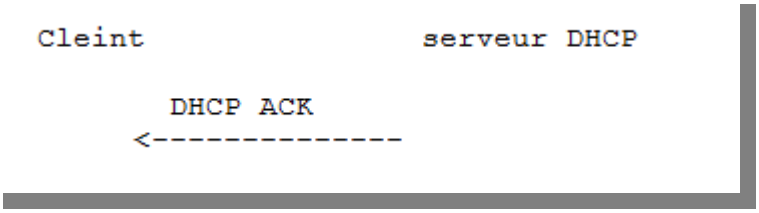


Le client répond par diffusion avec un message DHCP REQUEST pour notifier aux autres serveurs dont les offres n'ont pas été retenues de ne plus tenter de proposer des offres.

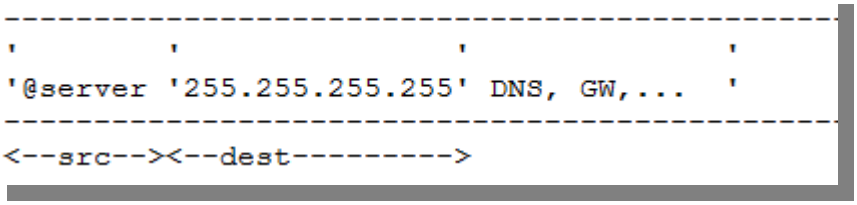


Scénario 4 : Confirmation de l'offre

Le serveur envoie de nouveau un message d'accusé de réception au client pour lui confirmer l'utilisation des éléments de connexion reçus dans un message ''DHCP ACK''



Jusque là, le client n'a pas encore tous les éléments possibles. C'est dans ce paquet que le client recevra le masque du sous-réseau, les adresses des serveurs DNS, et celle de la passerelle ainsi que d'autres paramètres.

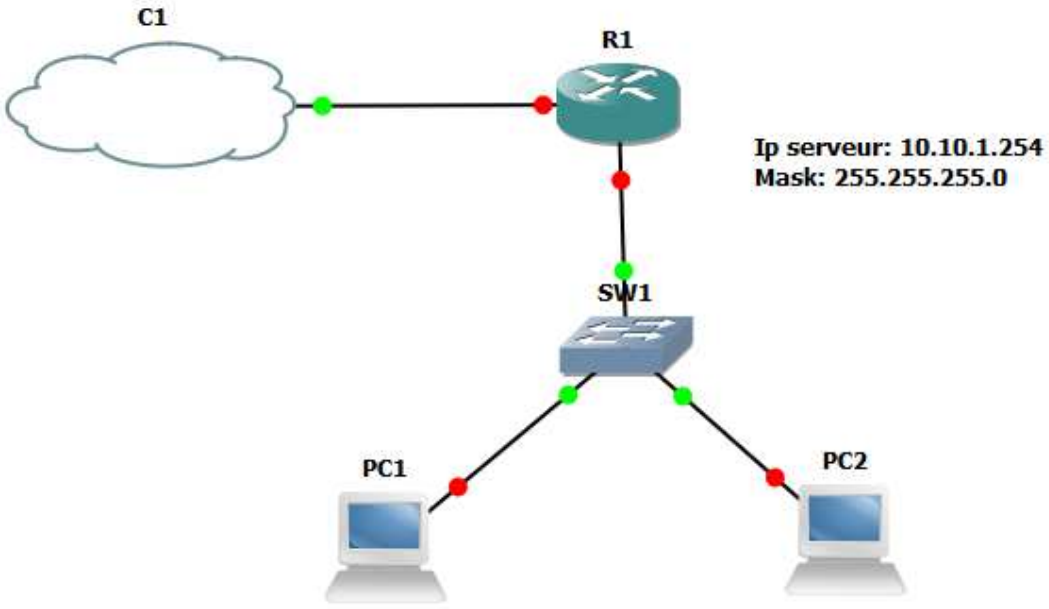


NB: Les éléments TCP/IP fournis par un serveur DHCP a un client porte le nom de bail. les baux ont une durée minimale et maximale de validité. Un client DHCP qui a épuisé les 50% de la durée maximale de validité de son bail se doit de le renouveler auprès du serveur DHCP. Sous Linux, le système conserve les informations sur les dernières utilisations des baux dans le fichier /var/lib/dhcp.lease. Voici un extrait du fichier d'une station de travail.

```
lease {
  interface "eth0";
  fixed-address 192.168.1.91;
  option subnet-mask 255.255.255.0;
  option dhcp-lease-time 86400;
  option routers 192.168.1.1;
  option dhcp-message-type 5;
  option dhcp-server-identifier 192.168.1.1;
  option domain-name-servers 213.154.95.126,213.154.64.13;
  renew 2 2014/03/25 20:25:46;q
  rebind 3 2014/03/26 08:02:46;
  expire 3 2014/03/26 11:02:46;
}
```

Etude de cas 2.1 : Mise en œuvre du service DHCP sous Cisco

Cette étude de cas est consacrée à la mise en place d'un serveur DHCP sur un routeur Cisco. Nous allons monter un LAB sous GNS3. Une documentation montrant l'installation et les premiers pas avec GNS3 se trouve sur le site www.ec2lt.sn, consulter l'onglet « Rapports ». Voici l'architecture que nous avons montée dans le LAB0.



La présence du nuage dans le LAB permet de relier le réseau monté dans le LAB à un réseau local. Pour tester l'attribution des éléments TCP/IP, nous avons relié par un câble la station sur laquelle le LAB a été monté à une autre station terminale.
Nous allons commencer par configurer notre serveur.

```
R1#configure terminal
R1#(config)#ip dhcp pool uvs_ec2lt
R1#(dhcp-config)#network 10.10.1.0 /24
```

Avec cette configuration, les hôtes du réseau peuvent déjà émettre des requêtes DHCP et obtenir les éléments TCP/IP. Cette configuration minimale est trop basique. Il faut préciser une plage. Pour définir une plage, il faut créer une classe.

```
#configure terminal
(dhcp-config)#class ec2lt
(config-dhcp-pool-class)#address range 10.10.1.100 10.10.1.200
```

Nous allons préciser l'adresse de la passerelle en utilisant la commande « default routeur @IP_PASSERELLE

```
#configure terminal
(dhcp-config)#ip dhcp pool uvs_ec2lt
#(dhcp-config)# default routeur 10.10.1.254
```

Il faut aussi préciser l'adresse des serveurs DNS s'il y'en a beaucoup, sinon préciser l'unique que vous disposez par la commande « dn server @IP_DNS »

```
#configure terminal
(dhcp-config)#ip dhcp pool uvs_ec2lt
#(dhcp-config)# dns server 10.10.1.254
```

Etude de cas 2.2 : Configuration des stations terminales

Nous avons transformé des routeurs en stations terminales. Pour obtenir les éléments de mise en réseau automatiquement, nous allons exécuter la commande « ip address dhcp » sur les stations terminales.

#PC1

```
PC1(config)#interface fa0/0
PC1(config-if)#ip address dhcp
*Apr  1 17:54:42.407: %DHCP-6-ADDRESS_ASSIGN: Interface
FastEthernet0/0 assigned DHCP address 10.10.1.1, mask
255.255.255.0, hostname PC1
PC1#ping 10.10.1.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.1.2, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
8/83/144 ms
```

PC2

```
PC2(config)#interface fa0/0
PC2(config-if)#ip address dhcp
*Apr  1 17:58:56.795: %DHCP-6-ADDRESS_ASSIGN: Interface
FastEthernet0/0 assigned DHCP address 10.10.1.2, mask
255.255.255.0, hostname PC2
PC2#ping 10.10.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.1.1, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
76/152/264 ms
```

Etude de cas 2.3 : Mise en œuvre d'un relais DHCP

Cette étude de cas illustre des principes de fonctionnement d'un relais DHCP. Peut-on se poser la question pourquoi avoir un relais alors qu'un serveur DHCP suffirait ? Il s'agit ici de contourner certaines difficultés en réseaux. Admettant que le réseau a été segmenté (sous-réseaux) et que ces sous-réseaux sont séparés par des routeurs. Or, un routeur limite les domaines de diffusion. Pour cela, les requêtes DHCP envoyés depuis les clients n'atteindront pas les serveurs DHCP, car ceux-ci sont injoignables. La solution idéale ici est de mettre en place un relais DHCP sur chaque segment du réseau. Celui-ci va écouter les requêtes et les relaye au serveur DHCP dédié. Nous allons monter deux sous-réseaux Rx1 (réseau de l'EC2LT) et Rx2 (réseau de l'ESMT) qui ont respectivement les adresses suivantes : 192.168.2.0/24 et 172.16.1.0/24. Notre architecture sera montée dans GNS3.

Chapitre 3: Le service DNS

Dans un réseau IP, deux stations terminales A et B ne peuvent communiquer que si l'une des machines connaisse l'adresse IP de l'autre. Pour que la machine A détermine le réseau de la machine B, elle applique son masque à l'adresse IP de B. Si l'adresse réseau de Best égale à l'adresse réseau de A, alors les deux machines peuvent communiquer. Au cas contraire la machine A remet le paquet à sa passerelle

Centre de résolution

Avant, pour effectuer la résolution, on utilisait un fichier texte (host.txt). Grâce à ces fichiers, on faisait une correspondance entre une adresse IP et un nom d'hôte. Chaque utilisateur du réseau devrait donc télécharger ce fichier et le stocker dans le fichier /etc/hosts. L'inconvénient de ce système est qu'il était difficile de faire la mise à jour, non seulement une augmentation de la taille de fichier due au nombre d'hôtes dans le réseau (plus on fait d'enregistrement, plus la taille du fichier augmente. Ce système de résolution était centralisé. Pour pallier à cela, il fallait trouver un système de résolution dynamique.

Le service DNS

DNS (Domain Name System) est un service de résolution de noms d'hôtes permettant de faire correspondre une adresse IP à un nom de machine (FQDN) et vice versa. L'objectif du DNS est de décentraliser le centre de baptême en vue d'avoir un service de résolution de noms sous forme d'arborescence dont la racine est symbolisée par un point (•). Définir les domaines du premier niveau (constitués de deux caractères pour représenter les pays ou trois pour désigner les organisations).

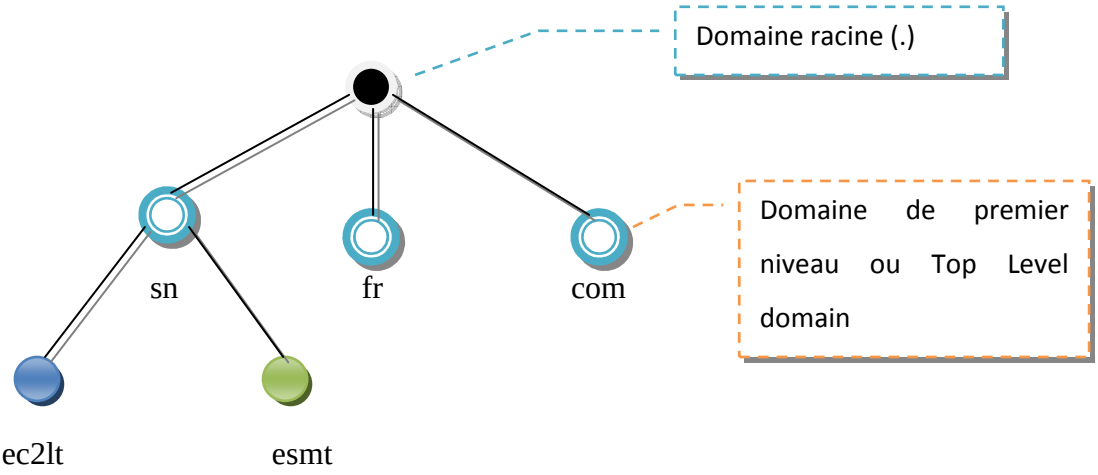


Figure 3.1 : Hiérarchie de noms DNS

Hiérarchie des protocoles DNS

L'arborescence DNS est organisée comme suit :

- Au sommet de l'arborescence se trouve le domaine racine, symbolisé par un point (.).
- Ensuite, nous avons les domaines supérieurs dits domaine de premier niveau (TLD=Top Level Domaine). Ils sont généralement représentés par le type d'organisation ou la localité du domaine (.ca, .fr, .com , .sn, ...).
- Après le TLD, s'imbriquent les domaines inférieurs dits de deuxième niveau. Ce sont généralement les noms d'entreprises ou d'organismes possédant le domaine (par exemple ec2lt, esmt, rtn ...).
- Enfin, un sous domaine peut être utilisé pour subdiviser le domaine. Mais généralement on trouve à ce stade le nom d'hôte qu'on désire contacter (www, poste1 ...).

Ci-dessous quelques règles énoncées pour bien élaborer une hiérarchie de noms de domaine. Un nom de domaine ne doit comporter de caractère point(.)

- Un nom sous_domaine_complet doit respecter ce format : ***sub_domain_name.domain_name***

Exemple : **ec2lt.sn ou esmt.sn ou rtn.sn**

- une machine dans un domaine doit avoir un nom d'hôte appelé FQDN (Full Qualify Domain Name) ***nomhote.nomcomplet.nomdomaine***

Exemple: poste1.ec2lt.sn

Concepts Liés au DNS

- **Serveur primaire d'un domaine**: machine sur laquelle on baptise ou les autres machines ou on renseigne les informations du domaine.
- **Enregistrement:** toutes informations stockées sur un serveur DNS
- **Serveurs secondaires**: serveurs qui sont appelés à faire une copie des informations du domaine et les garder

Les différents types d'enregistrement

- Enregistrement de type **NS**(Name server): permet de déclarer les serveurs DNS d'un domaine.

Exemple**:** ***ec2lt.sn. IN NS poste1.ec2lt.sn***. : Le serveur DNS du domaine ec2lt.sn est la machine poste1.ec2lt.sn

- Enregistrement de type **A**: permet de faire la résolution directe

Exemple: ***poste1.ec2lt.sn. IN A 192.168.1.10 :*** la machine poste1.ec2lt.sn a pour adresse IP 192.168.1.10

- Enregistrement de type **CNAME** (Canonical Name ou nom canonique)**:** permet de surnommer ou de créer un alias pour un nom d'hôte.

Exemple: ***www.ec2lt.sn. IN CNAME poste1.ec2lt.sn***. : La machine www.ec2lt.sn a pour véritable nom pose1.ec2lt.sn

- Enregistrement de type **MX** (Mail eXchange): permet d'indiquer les serveurs de messagerie du domaine avec leur ordre de priorité.

Exemple: ***ec2lt.sn. IN MX 2 poste3.ec2lt.sn.***

ec2lt.sn. IN MX 1 poste8.ec2lt.sn.

- Enregistrement de type **PTR** (Point To Record): permet de faire la résolution inverse

Exemple: ***10.1.192.in-addr.arpa. IN PTR poste1.ec2lt.sn.***

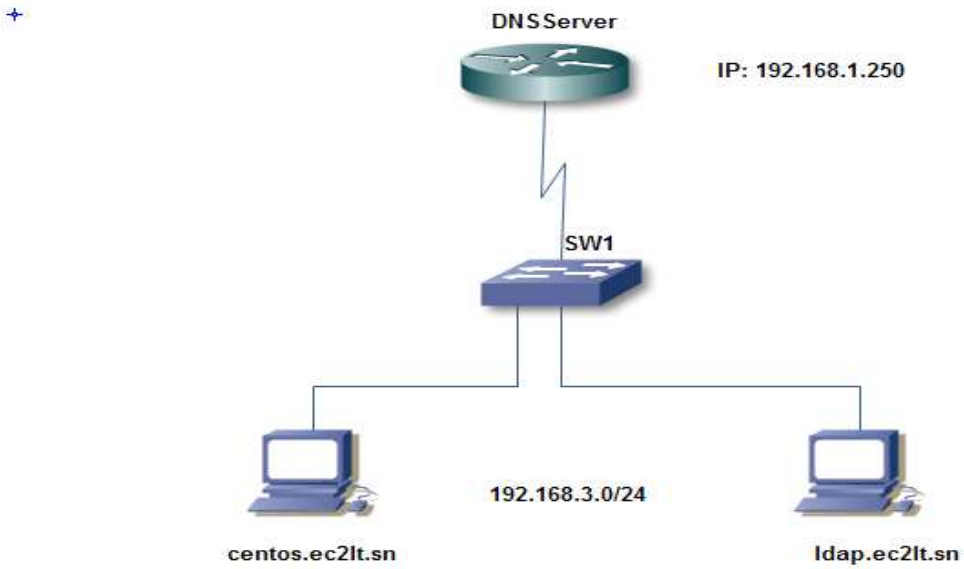
- Enregistrement de type **SOA** (State Of Autority): renferme 7 informations à savoir:
  - nom du serveur DNS primaire du domaine
  - l'adresse e-mail de l'administrateur du domaine:
  - le numéro de série d'information stockée
  - durée de rafraichissement (indique au serveur secondaire quand est-ce qu'il va revenir pour copier des informations de mise a jour
  - durée de réessaie (indique au serveur secondaire le nombre de fois il doit revenir pour prendre des infos)
  - durée d'expiration: somme des durées de réessaie (retry) pour indiquer au serveur secondaire de ne plus essayer de venir.
  - durée minimale de validité des informations: temps pendant lequel le secondaire doit continuer à servir pendant que le serveur principal n'est pas disponible.

Voici un extrait des premières lignes d'un fichier de zone pour le serveur DNS du domaine EC2LT.

```
; BIND data file for local loopback interface
;
$TTL    604800
ec2lt.sn. IN      SOA     server.ec2lt.sn. test.ec2lt.sn. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800)       ; Negative Cache TTL
```

Etude de cas 3.1 : Mise en œuvre du service DNS sous Cisco

Nous allons dans cette étude de cas montrer comment mettre en place un service de résolution de nom en environnement Cisco (routeurs Cisco). Nous allons monter une architecture expérimentale avec un serveur de nom. Ensuite nous allons prolonger cette étude de cas en combinant le service DNS et le service DHCP.



Nous allons monter cette architecture dans notre LAB sous GNS3

Comment fait-on pour transformer un routeur Cisco en un serveur de noms ? Voici quelques étapes de mise en œuvre :

- activer le routeur en tant que serveur de nom (ip dns-server)
- indiquer l'adresse IP du serveur DSN (ip name-server @IP)
- activer le routeur en tant que serveur de nom principal avec les enregistrements de type SOA (ip dns primary en2lt.sn soa dns.ec2lt.sn yvan.ec2lt.sn)
- activer l'enregistrement de NS (ip host ec2lt.sn ns dns.ec2lt.sn)
- activer l'enregistrement de type A (ip host dns.ec2lt.sn), idem pour les hôtes centos et ldap.

Voici un extrait de la configuration du serveur DNS primaire. Nous ne gérons pas encore le serveur secondaire du domaine EC2LT (plus tard).

```
hostname DNSServer
ip dns-server
ip name-server 192.168.1.250
ip dns primary ec2lt.sn soa dns.ec2lt.sn  yvan.ec2lt.sn
ip host ec2lt.sn ns  dns.ec2lt.sn
ip host dns.ec2lt.sn 192.168.1.250
ip host ec2lt.sn mx 1  msg.ec2lt.sn
ip host msg.ec2lt.sn  192.168.1.20
ip host ldap.ec2lt.sn 192.168.1.30
```

Etude de cas 3.2 : Configuration du client hôte nommé centos

Avec cette configuration, seul le serveur a l'habilité de résoudre les noms d'hôtes, car tous les enregistrements sont faits à son niveau. Pour que cela soit pris en compte coté clients, il faut ajouter la configuration du serveur au niveau de chaque hôte. Vous remarquerez qu'on donné un autre FQDN à la machine « centos » au lieu de « centos.ec2lt.sn » nous l'avons enregistrer en tant que « msg.ec2lt.sn ».

```
centos(config)#interface fa0/0
centos(config-if)#ip address 192.168.1.20 255.255.255.0
centos(config-if)#no shutdown
centos(config)#ip name-server 192.168.1.250
centos(config)#ip domain-lookup
```

Résultat sur la console d'un ping vers ldap.ec2lt.sn

```
centos# ping ldap.ec2lt.sn
Translating "ldap.ec2lt.sn"...domain server (192.168.1.250)
[OK]
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.30, timeout is
2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
36/153/204 ms
```

Etude de cas 3.3: Configuration du client « ldap »

Idem comme pour l'hôte centos, le FQDN de l'hôte « ldap » est « ldap.ec2lt.sn »

```
ldap(config)#interface fa0/0
ldap(config-if)#ip address 192.168.1.30 255.255.255.0
ldap(config-if)#no shutdown
```

Nous allons aussi ajouter le serveur DNS à l'hôte ldap

```
ldap(config)#ip name-server 192.168.1.250
ldap(config)#ip domain-lookup
```

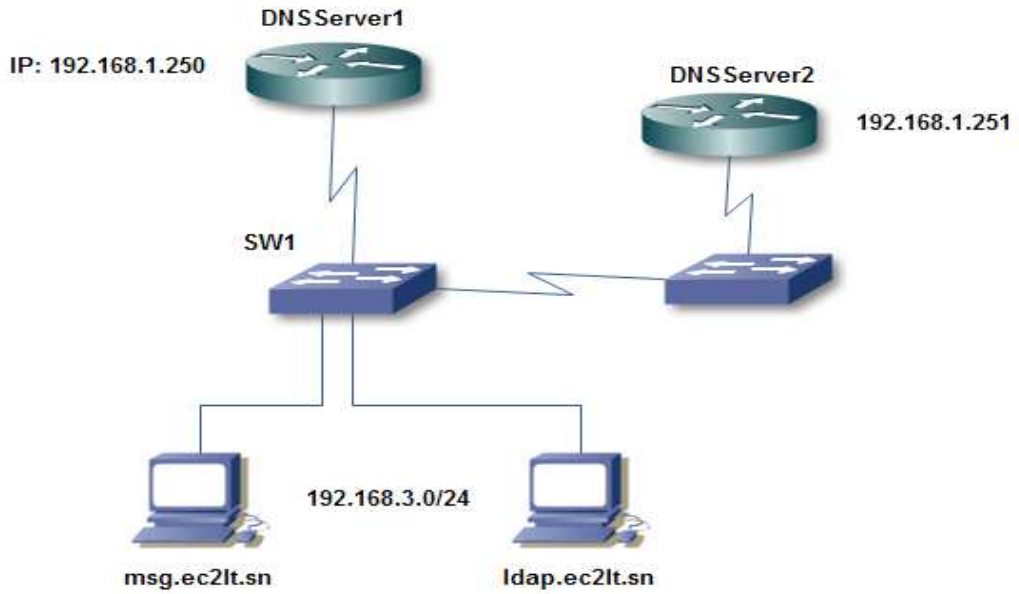
Résultat à l'écran d'un ping vers « msg.ec2lt.sn »

```
ldap#ping msg.ec2lt.sn

Translating "msg.ec2lt.sn"..domain server (192.168.1.250)
[OK]

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.20, timeout
is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max
= 80/125/272 ms
ldap#ping dns.ec2lt.sn

Translating "dns.ec2lt.sn"..domain server (192.168.1.250)
[OK]

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.250, timeout
is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
156/194/272 ms
```

Etude de cas 3.4 : Mise en œuvre d'un DNS secondaire

Dans l'étude de cas 3.1, nous avons montré comment mettre en œuvre un serveur DNS primaire pour la résolution directe. Pour des raisons de disponibilité en réseau, nous allons prévoir un serveur de secours pour le domaine principal.

Dans le LAB que nous avons monté, le serveur DNS primaire est appelé DNSServer1 (192.168.1.250) et le secondaire DNSServer2 (IP : 192.168.1.251).




```
DNSServer2(config)#ip dns server

DNSServer2(config)#ip domain name ec2lt.sn

DNSServer2(config)#ip dns primary ec2lt.sn soa dns.ec2lt.sn
yvan.ec2lt.sn

DNSServer2(config)#ip name-server 192.168.1.250
192.168.1.251

DNSServer2(config)#ip host ec2lt.sn ns 192.168.1.250

DNSServer2(config)#ip host msg.ec2lt.sn 192.168.1.20

DNSServer2(config)#ip host ldap.ec2lt.sn 192.168.1.30
```

Ensuite, il faut ajouter le DNSServer2 dans la liste des serveurs DNS à contacter au niveau du client.

```
centos(config)#ip name-server 192.168.1.250 192.168.1.251
```

```
ldap(config)#ip name-server 192.168.1.250 192.168.1.251
```

Pour tester, nous allons arrêter le serveur primaire (DNSServer) et tenter de joindre un hôte par son nom.

```
centos#ping ldap.ec2lt.sn
Translating "ldap.ec2lt.sn"...domain server (192.168.1.250)
(192.168.1.251) [OK]

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 192.168.1.30, timeout is 2
seconds:

.!!!!

Success rate is 80 percent (4/5), round-trip min/avg/max =
124/171/220 ms
```

Nous voyons bien la station terminale a envoyer les requêtes de résolution vers son serveur principal, puis vers le secondaire.

Etude de cas 3.5 : Mise en œuvre du service DNS sous Linux

Jusque là nous avons travaillé en environnement Cisco. Cette fois, nous allons manipuler un peu les outils d'administration système sous Linux. Lorsqu'on change d'environnement, on

cherche à s'adapter. Vous avez remarqué qu'en environnement Cisco, il n'y avait pas de packages supplémentaires. Or, là il faut installer les packages, cette tâche peut paraître compliquer pour certains, mais en réalité ce n'est pas le cas. Il suffit de savoir quel package il faut, pour tel service et comment l'installer ? Vous allez vous familiariser aux commandes Linux (très facile à retenir si l'on pratique une ou deux fois). Nous travaillons sous Linux, mais sur une version d'Ubuntu 12.04. Le package du service DNS est appelé « bind9 » veuillez consulter le site officiel d'Ubuntu www.ubuntu.org (pour plus de détails).
Dans un terminal, nous allons exécuter ceci.

```
#apt-get install bind9
```

Pour configurer un DNS sous Linux, il faut :

- déclarer la zone dans le fichier /etc/bind/named.conf.default-zones
- créer le fichier de zone dans le répertoire /etc/bind9
- gérer l'appartenance et les droits sur le fichier de zone ainsi que les options.

Etude de cas 3.5.1 : Déclaration de la zone

La déclaration de la zone se fait dans le fichier /etc/bind/named.conf.default-zones. La déclaration d'une zone doit respecter le format suivant :

```
zone "zone_name" {
        type master/slave;
        file "name_of_zone_file";
};
```

Voici la déclaration de zone pour le domaine ec2lt.sn

```
zone "ec2lt.sn" {
        type master;
        file "/etc/bind/db.ec2lt";
};
```

Il faut ensuite créer le fichier de zone. Le fichier de zone c'est le fichier qui contient toutes les informations du domaine. Des exemples de ce fichier existe dans le répertoire /etc/bind, une copie d'un exemplaire renommé conforme au nom que vous avez indiqué dans lors de la déclaration de la zone.

Extrait du fichier de déclaration de zone pour le domaine ec2lt.sn

```
; BIND data file for local loopback interface
;
$TTL    604800
ec2lt.sn. IN      SOA     server.ec2lt.sn. test.ec2lt.sn. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
ec2lt.sn.          IN      NS      server.ec2lt.sn.
server.ec2lt.sn.   IN      A       192.168.2.90
www                IN      CNAME   server.ec2lt.sn.
```

Le service DNS est lancé par le daemon « bind » (programme ou utilisateur chargé de lancer le service sur le système. En générale, vous exécuter les tâches d'administration sous les droits de « root ». Lorsque vous créer les fichiers en tant que super-utilisateur, ceux-ci lui appartiennent. Si les fichiers appartiennent à « root », bind aura du mal à exécuter le programme. Pour cela il faut changer l'appartenance du fichier ainsi que les droits positionnés sur le fichier.

```
# cd /etc/bind && chown bind:bind db.ims && chmod 777 db.ec2lt
```

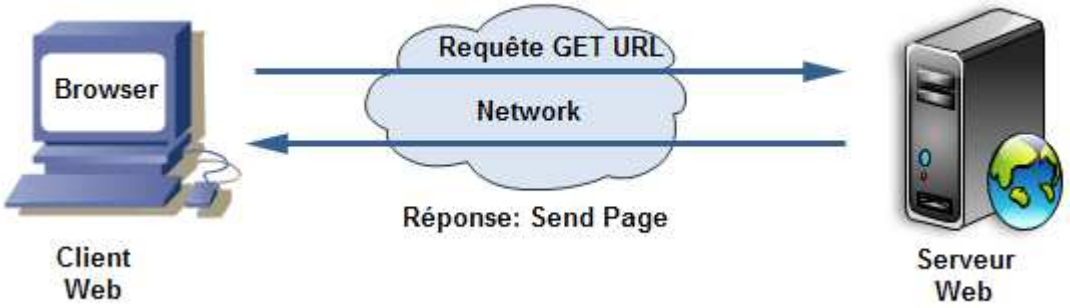
Ensuite, il faut modifier le resolver (client DNS). Lorsqu'il n'y a pas un serveur DHCP dans le réseau, et que l'on déployé le service DNS, il faut alors modifier le /etc/resolv.conf (fichier de configuration du client DNS) de chaque client ou a défaut, écrire un script Shell ou Perl pour exécuter cette tâche.

Chapitre 4 : Le service Web

Le web est devenu le moyen rapide d'accès aux ressources numériques par les usagers au sein des écoles, entreprises et institutions. Ces ressources hébergées sur des serveurs web sont accessibles en local ou via Internet. On désigne souvent le web par WWW (Word Wide Web) qui signifie tout simplement la toile. Les pages web hébergées sur les serveurs web sont écrites en HTML et exploitées par le protocole HTTP (HyperText Transfer Protocol). Il existe plusieurs types de serveurs web, mais ce chapitre est consacré à l'installation, et la mise en œuvre d'un serveur web apache sous Linux. Des études de cas sont traitées pour faciliter l'apprentissage.

4.1 Fonctionnement d'un serveur Web

Un serveur est un serveur qui héberge au moins un site web. Un site est un ensemble de pages web crée par un utilisateur (en général un programmeur). Le site web est accessible via un protocole de communication qui après traitements nécessaires affichent la page d'accueil du site. Il faut un protocole de communication (compris par le client et le serveur) et un protocole de transport pour l'envoie des pages web. Un serveur web fonctionne suivant une architecture client/serveur (requête/réponse). Le serveur web est accessible à partir d'un client web appelé « navigateur ».



L'utilisateur se saisie d'une application cliente (navigateur, browser) et formule une requête au serveur via ce format : http://address_server. Cette requête sera transmise par le serveur via un protocole de transport (TCP). Le serveur recevra la requête recherche l'emplacement de la ressource demandée et renvoie le résultat au protocole de transport TCP qui se chargera de le faire parvenir au client.

Pour comprendre le fonctionnement d'un serveur web, il faut connaitre les différents types de ressources. Il ya plusieurs types de ressources :

- URL (Uniform Resource Locator) : spécifie la localisation de la ressource de façon unique.
- URN (Uniform Resource Name) : c'est un mécanisme de nommage des ressources. L'URN est de la forme : urn : <NameSpace> :<SpecificString>. NameSpace est l'identificateur de nommage et le SpecificString est la chaine de caractère désignant la ressource
- URI (Uniform Resource Identifier) : c'est l'identifiant universel de la ressource. L'URI = URL + URN. Une URI est de la forme :
  <protocole>://<adresse serveur> :port/<chemin>/<ressource>

4.2 Le protocole HTTP

HTTP (Hyper Text Transfer Protocol) est le protocole de communication utilisé sur le web entre un client web et un serveur web. Le protocole HTTP a été défini dans la RFC 1945 et 2068 comme protocole de transfert de document (pages web) et de soumission de formulaire sur le web. HTTP utilise trois méthodes pour les échanges de documents : GET, POST et HEAD. La méthode GET est utilisée pour formuler les demandes de pages web. La méthode POST est utilisée pour la soumission de formulaire (On peut utiliser la méthode GET aussi). Pour récupérer les informations sur les documents ou gérer les proxy/cahe, HTTP utilise la méthode HEAD. D'autres méthodes tels PUT, LINK, UNLINK, DELETE sont aussi utilisées pour l'envoie des documents ou la gestion de site. Dans la suite, nous allons nous concentrer sur les différentes études de cas.

Etude de cas 4.2.1 : Découverte d'apache

Nous allons concentrer une petite attention à la découverte d'apache. Apache est l'un des serveurs web le plus utilisé à cause de sa simplicité (en grande partie grâce au protocole HTTP). Le binaire du programme est appelé « httpd » sous les distributions RedHat et versions et « apache2 » sous les distributions Debian et versions.

L'installation d'apache ne nécessite aucun effort, juste exécuter la commande suivante :

```
# apt-get install apache2
```

Le fichier principale de configuration d'apache se trouve dans le répertoire /etc/apache2/apache2.conf mais les fichiers complémentaires se trouvent dans le dossier /etc/apache2. Pour bien fonctionner, apache implémente certains modules des applications. Le fichier de configuration des modules se trouvent dans le répertoire /etc/apache2/modules.conf et les modules disponibles sont dans le fichier /etc/apache2/mods-avaible et les les modules chargés sont contenus dans le fichier /etc/apache2/mods-enable. Pour démarrer apache, il suffit de lancer le script de démarrage *apache2* et lui passer l'argument *start* ou *restart*.

```
# service apache2 restart/start
# /etc/init.d/apache2 restart/start
```

Voici un extrait du fichier des modules disponibles d'apache

```
# vim /etc/apache2/mod_avaible

authn_file.load       dir.conf           mime_magic.conf
ssl.load
authnz_ldap.load      dir.load           mime_magic.load
status.conf
authz_dbm.load        disk_cache.conf    negotiation.conf
status.load
authz_default.load    disk_cache.load    negotiation.load
substitute.load
authz_groupfile.load  dump_io.load       php5.conf
suexec.load
```

Apache a besoin des modules pour l'exécution des scripts PHP. Pour cela, nous allons installer le package PHP (version 5).

```
# apt-get install php5
```

PHP (mod_php) est un module important pour apache, car il permet d'exécuter les scripts CGI depuis un serveur apache. La fonction phpinfo() ; dans un script PHP permet d'afficher les informations sur les modules PHP dans un navigateur.

Nous allons créer un petit script appelé *info.php*.

```
# vim /var/www/info.php
<?php
phpinfo();
?>
~
```

Puis dans le navigateur http://localhost/info.php

Etude de cas 4.2.2 : Site virtuels par adresse

Un serveur web peut gérer plusieurs sites web. Ces sites web sont appelés serveurs virtuels. Pour créer un site virtuel sous apache, il faut vérifier que le module "mod_vhost_alias" qui permet de créer un nombre énorme de sites virtuels identifiables par leur document racine, leur adresse IP et leur numéro de port. Dans cette étude de cas nous allons montre comment créer des serveurs virtuels ou sites virtuels?

Sous Ubuntu, le fichier de configuration d'apache se trouve dans le répertoire /ect/apache2.conf et les fichiers relatifs au serveur virtuel se trouve dans les fichiers /etc/apache2/sites-availlables (dossier qui contient la liste des sites virtuels disponibles) ou /etc/apache2/sites-enable (dossier qui contient la liste des sites virtuels activés). Un site virtuel par adresse, est identifie par une adresse IP. Les sites virtuels sont crées dans les directives <*VirtualHost*> ..... <*/VirtualHost*> comme suit:

```
<VirtualHost 192.168.1.90>
        ServerAdmin webmaster@localhost

        DocumentRoot /var/www/siteaddr

</VirtualHost>
```

Puis on active le site en utilisant l'outil a2ensite nom_site

```
# a2ensite servervirt1
Enabling site servervirt1.
To activate the new configuration, you need to run:
service apache2 reload
```

L'exécution de cette commande créée un lien dans le dossier des sites actives "/etc/apache2/sites-enable. Il faut ensuite créer un contenu de la page dans le dossier racine du site indiqué par le paramètre *DocumentRoot /var/www/siteaddr*. On peut faire une copie du fichier index.html par defaut dans ce dossier, puis dans le navigateur http://192.168.1.90

Etude de cas 4.2.3 : Sites virtuels par nom

Un serveur web peut héberger plusieurs serveurs virtuels (module mod_vhost_alias). Pour cette étude de cas, nous allons créer deux serveurs virtuels basés sur le nom respectivement monsite1.ec2lt.sn et monsite2.ec2lt.sn pointant sur une même adresse IP. Le principe de création est le même que pour le précédent. Commençons par créer les dossiers racines des sites *monsite1* et *monsite2*

```
# mkdir –p /var/www/monsite1 && vim index.html
# mkdir –p /var/www/monsite2 && vim index.html
```

Contenu du fichier index.html pour monsite1. Nous allons taper un peu de code HTML.

```
<html>
     <head><title> SITE 1 </title>
     </head>
     <body>
          <center><h1>Mon premier site</h1>
               <p> Site virtuel par nom </p>
          </center>
     </body>
</html>
```

Nous allons enregistrer le fichier sous index.html

```
<html>
     <head><title> SITE 2 </title>
     </head>
     <body>
          <center><h1>Mon deuxieme site</h1>
               <p> Site virtuel par nom </p>
          </center>
     </body>
</html>
```

Ensuite, il faut créer le fichier du site dans /etc/apache2/sites-avaible

```
# cd /etc/apache2/sites-avaible && vim monsite1
NameVirtualHost 192.168.1.90 :80

<VirtualHost *:80>
        ServerAdmin admin@im.sn
        ServerName  monsite1.ec2lt.sn
        DocumentRoot /var/www/monsite1
</VirtualHost>
```

Extrait du fichier de serveur virtuel monsite2

```
# cd /etc/apache2/sites-avaible && vim monsite2
NameVirtualHost 192.168.1.90 :80
<VirtualHost *:80>
        ServerAdmin admin@im.sn
        ServerName  monsite2.ec2lt.sn
        DocumentRoot /var/www/monsite2
</VirtualHost>
```

Activons les deux sites par *a2ensite*

```
# a2ensite monsite1
# a2ensite monsite1

# service apache2 restart
```

Modifions ensuite le fichier /etc/hosts pour pointer l'adresse vers monsite1.ec2lt.sn et monsite2.ec2lt.sn ou faire un enregistrement dans le fichier de zone du serveur DNS.

```
monsite1        IN      CNAME   server.ec2lt.sn.
monsite2        IN      CNAME   server.ec2lt.sn.
```

Puis dans le navigateur *http://monsite1.ec2lt.sn* pour afficher la page d'accueil de monsite1 et *http://monsite2.ec2lt.sn* pour afficher la page d'accueil de monsite2.

Etude de cas 4.2.4 : Site par dossier

Un site par dossier n'est pas complexe à déployer. Il suffit de créer un dossier qui contiendra toutes les pages web dans le répertoire racine d'apache (/var/www). Nous allons créer un dossier appelé « siteuvs » auquel nous allons donner les droits 755. Dans ce dossier nous allons créer un fichier index.html dans lequel nous allons mettre un petit script HTML comme suit :

```
<html>
     <head><title> SITE UVS </title>
     <body>
          <center><h1>Du nouveau au Senegal</h1>
               <p>UVS, l'universite  proche de chez vous</p>
          </center>
     </body>
</html>
```

Dans le navigateur http://192.168.1.90/siteuvs le contenu de la page ci-dessus doit apparaitre dans la page d'accueil.

De ce fait, on peut copier le dossier d'un site et placer dans le répertoire racine d'apache.

Etude de cas 4.2.5 : Site personnel

Dans cette étude de cas, nous allons montrer chaque utilisateur peut avoir la possibilité d'héberger une page personnelle au niveau du serveur. De ce fait, pour accéder à la page de l'utilisateur, celui-ci devra taper dans le navigateur : http://ipserver/~nom_user. A partir de là, le serveur web recevant la requête vers le répertoire de l'utilisateur. Pour cela, pour un début nous devoir agir sur utilisateur système existant, mais dans la suite nous montrerons quelques techniques permettant d'éviter de créer à chaque fois un utilisateur système. Il faut d'abord s'assurer que le module « userdir » est chargé (activé) sinon le faire manuellement.

```
# a2enmod userdir
Enabling module userdir.
To activate the new configuration, you need to run:
service apache2 restart
```

Après le chargement du module, il faut maintenant le configurer. Un coup d'œil dans le dossier des modules disponibles pour identifier le fichier qui contient les paramètres du module.

```
<IfModule mod_userdir.c>
        UserDir public_html
        UserDir disabled root

        <Directory /home/*/public_html>
                AllowOverride FileInfo AuthConfig Limit
Indexes
                Options MultiViews Indexes
SymLinksIfOwnerMatch IncludesNoExec
        </Directory>
</IfModule>
```

Le « UserDir » pointe vers un référentiel « public_html », il s'agit d'un répertoire qu'il faut créer dans le répertoire de base de l'utilisateur puis nous allons créer dans ce dossier un page d'accueil. Notre utilisateur système ici s'appelle « test »

```
$su test
[test@ubuntu ~]$ mkdir public_html && cd public_html
[test@ubuntu ~]$ $chmod -R 755 public_html && cd public_html
```

Mettons ce bout de page HTML dans le dossier public_html

```
<html>
     <head><title> SITE PERSO </title>
     </head>
     <body>
          <center><h1>Bonjour</h1>
           <p>Bienvenu sur ma page Perso</p>
          </center>
     </body>
</html>
```

On redémarre apache puis dans le navigateur http://192.168.1.90/~test ou http://server.ec2lt.sn/~test

4.3 Apache et les modes proxy

Le service web est le plus utilisé en entreprise. Les utilisateurs ont une certaine facilité à accéder aux ressources informatique à travers, que ce soit en local ou à distance. Le système d'information peut parfois être objet d'une attaque. Pour ce faire, il est important de déployer des certaines règles de sécurité (filtrage) au sein du réseau. Des serveurs proxy peuvent être déployés pour régler cette question. On proxy désigne un ordinateur appelé serveur mandataire qui se situe a la frontière entre le client et le serveur web dédié. Un serveur proxy peut jouer les rôles suivants :

- hébergement de pages web en local
- protection contre les intrusions
- masquage des informations concernant votre système

En mode sans proxy, le client accède directement aux ressources mises disposition au niveau du serveur. Il n'y a pas un intermédiaire (voir figure)

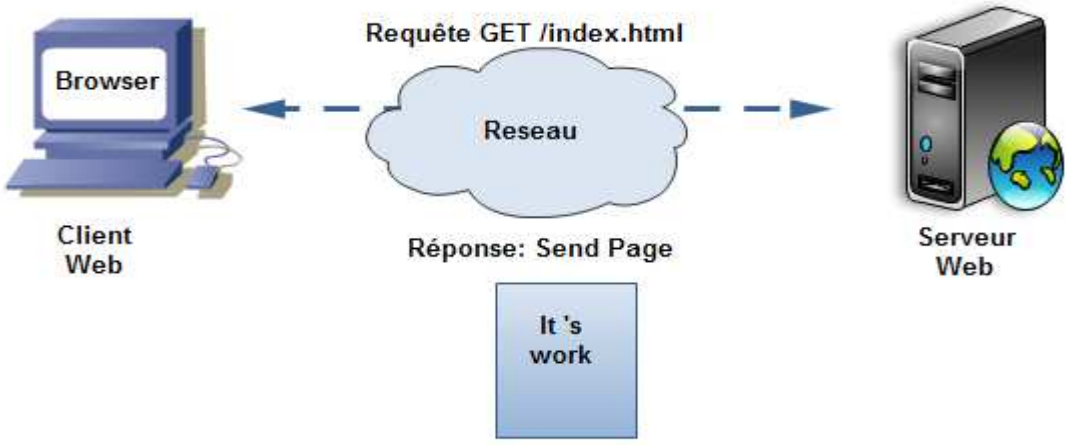


Figure 4.2 : Fonctionnement sans proxy

Par contre le fonctionnement en mode fait intervenir un équipement intermédiaire (le serveur proxy) qui se ment en frontal du serveur web de l'entreprise. D'après ce schéma, le serveur web de l'entreprise est caché derrière le proxy. En ce moment l'utilisateur ne voit que le proxy. A partir de là, toute ressource sollicitée et qui n'est pas accessible directement, c'est le serveur proxy qui est chargé de les traiter.

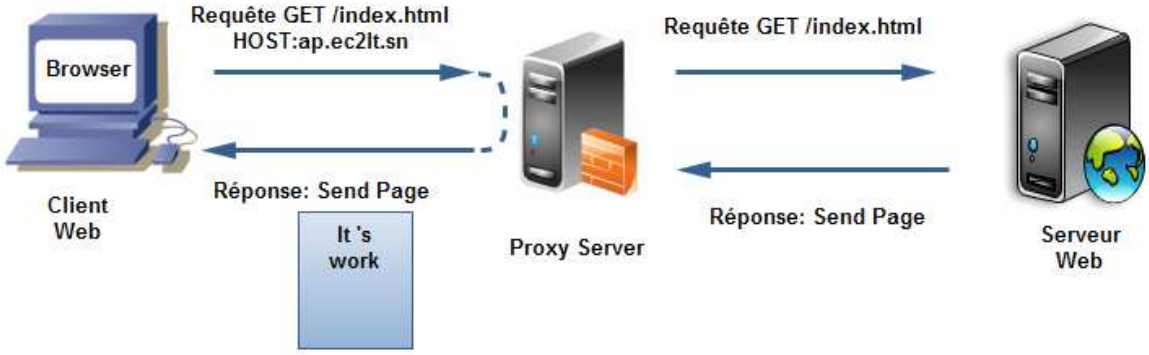


Figure 4.3 : Fonctionnement en mode proxy

Etude de cas 4.3.1 : Mise en œuvre d'un proxy sous apache

Cette étude de cas montre comment peut-on mettre en place un proxy en frontal d'apache ? Nous ne traitons d'autres cas plus complexe, ces cas seront traités plus tard. On considérera l'architecture de fonctionnement en mode proxy ci-dessus à partir de laquelle nous allons mener les expériences. Il ya plusieurs modes proxy, mais nous utiliserons que les modes *ProxyPass et ProxyPassReverse*.

ProxyPass est une directive de mandatement d'apache permettant de traduire les requêtes venant sur un lien a un serveur mandataire capable de faire les redirections. Ce cas de figure est fréquent en entreprise.

ProxyPassReverse permet de repartir la charge sur plusieurs serveurs. Le serveur frontal accepte les requêtes et les redirige aux serveurs terminaux. La directive ProxyPassReverse laisse apache adapter l'en-tête de la réponse. Si ProxyPass a été utilisé, elle réécrit les en-têtes provenant du serveur mandataire de telles sorte qu'il semble provenir d'un même serveur.

Tout d'abord il faut charger les modules nécessaires pour les modes proxy : il s'agit des modes *proxy* et *proxy_http.*

```
# a2enmod proxy
# a2enmod proxy_http
```

Nous allons créer un site virtuel sur le werveur web1 tandisque le serveur web2 n'est pas accessible directement. L'utilisateur ne connait pas l'adresse du serveur web2 mais va tenter d'accéder à la page *http://monsite1.ec2lt.sn* sur le serveur proxy, celui-ci va maintenant grâce aux directives ProxyPass et ProxyPassReverse rediriger la requête vers le serveur web2.

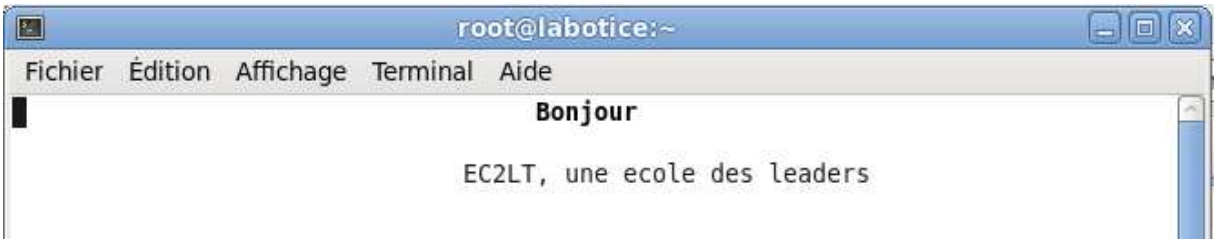
Au départ le site http://monsite1.ec2lt.sn afficherai le contenu de la page suivante :

```
# cat /var/www/monsite1/index.html
<html>
     <body>
          <center>  <h1>Du nouveau au Senegal</h1>
               <p>UVS, l'universite  proche de chez vous</p>
          </center>
</body>
</html>
```

Nous allons maintenant cacher le serveur web2 derrière le serveur web1 et nous inndiquerons u serveur web1 que si un utilisateur tape dans son navigateur http://monsite1.ec2lt.sn qu'il redirige la requête vers le serveur web2 à l'adresse http://web2.ec2lt.sn et on affichera la page d'accueil du serveur web2.

```
<VirtualHost *:80>
        ServerAdmin admin@im.sn
        ServerName  monsite1.ec2lt.sn
        DocumentRoot /var/www/monsite1
        ProxyPreserveHost on
        ProxyRequests   off
        ProxyPass / http://web2.ec2lt.sn:80/
        ProxyPassReverse / http://web2.ec2lt.sn:80/
   <Proxy *>
        AddDefaultCharset off
        Order deny,allow
        Allow from all
   </Proxy>
</VirtualHost>
```

Capture de l'écran d'un utilisateur qui s'est conecté en mode console.



Et depuis le web