MOULAY Karim
FILLOT Denis
VINCENT Matthieu

23 mai 2007

Système et Réseaux II

Rapport de Travaux Pratique

Licence 3 Informatique

Université de Bourgogne

Sommaire

Index des Figures

Introduction

Ce rapport intervient dans le cadre des Travaux Pratiques du module de Système et Réseaux II de second semestre de licence Informatique. Il a pour but de résumer nos manipulations réalisées en TP afin qu'une personne le consultant puisse effectuer les mêmes opérations, mais aussi expliquer les notions qui sont manipulés et approfondir certains points pour éclairer le lecteur sur ce qui est mis en œuvre.

Le sujet de nos TP est l'installation et la configuration d'un serveur dans un réseau local, le serveur étant lui-même relié à d'autre afin de simuler le réseau d'un site d'une entreprise par exemple dans un groupe possédant plusieurs emplacement. Les réseaux locaux possèdent leur propre connexion Internet, il faut que le serveur puisse y avoir accès mais aussi chaque poste client. La machine serveur doit fournir les services habituellement utilisés dans un réseau c'est à dire l'adressage automatique de chaque machine, l'authentification par compte utilisateur sur les postes clients, un serveur de fichier lié aux comptes clients, etc., …

Pour mener à bien ce rapport, nous allons dans un premier temps nous intéressé à l'installation et la configuration du système d'exploitation. Ensuite nous verrons comment installer des programmes à partir de packages ou de sources ainsi que la configuration de services tel que DHCP, Apache…

Installation et configuration du système d'exploitation du serveur.

Installation de Debian Sarge

Le système d'exploitation désigné pour le serveur est la distribution Linux appelée Debian Sarge. Cette version de Linux est réputée pour être très fiable et efficace, elle est de plus déclinée dans d'autre distribution comme Ubuntu ou Knoppix, plus axées sur la bureautique et le grand publique.

Le nom Sarge de cette distribution est donné à la version 3.1, elle fait suite à celle appelée Woody et est sortie en juin 2005 après 3 ans de développement.
Son installation se fait par démarrage sur le CDROM où est gravé l'ISO que l'on peut télécharger sur Internet (à l'adresse : http://www.debian.org/). Le programme chargé d'installer le système nous demande en premier lieu la langue du programme parmi une liste conséquente ainsi que le pays pour enfin choisir un jeu de caractère spécifique.

L'étape suivante est le choix du nom de la machine, ce qui n'est pas très important en soit mais qui peut permettre d'identifier facilement le serveur. Nous avons choisi de nommer la machine en fonction de notre position dans la salle : « SRII-5 » puisque nous sommes le groupe 5. Ce groupe sera utilisé dans la plupart des numérotations d'identification (IP, domaine…). Ce nom est utile sur le réseau local pour l'atteindre plus facilement. La configuration réseau peut d'ailleurs être effectuée maintenant ou une fois l'installation effectuée. Nous avons choisit de faire cette configuration tout de suite en choisissant l'interface qui sera connecté au réseau de l'Université, nous reverrons plus tard à quoi cela correspond. Les différents champs sont pour nous :

Adresse IP :	172.31.20.25
Masque réseau :	255.255.255.0
Passerelle (Gateway) :	172.31.20.1
Serveur de nom (DNS) :	172.31.21.35

Précisons que nous avons une machine avec plusieurs interfaces réseaux et que dans ce cas il est important de se repérer entre chaque carte pour avoir une bonne configuration.

L'installation se poursuit durant un certain temps au bout duquel le programme nous demande le logiciel que nous voulons pour gérer le démarrage de l'ordinateur (boot), nous choisissons Grub. Ce choix importe peu étant donné qu'un seul système d'exploitation ne sera présent sur la machine.

A ce niveau l'installation du système en tant que tel est terminée mais il nécessite quelques configurations à effectuer au démarrage, un programme se lance automatiquement à cet effet. Ce dernier nous souhaite tout d'abord la bienvenue puis nous demande un mot de passe pour le compte super utilisateur (root), nous avons choisit « admin » ce qui est à éviter car très vulnérable au niveau sécurité (les

premières lettres d'une phrase avec un chiffre est un très bon mot de passe en général) mais que nous utilisons par simplicité mnémotechnique.

L'étape suivante consiste en la création d'un compte utilisateur classique, qui n'aura pas tous les droits que le super utilisateur peut avoir. L'utilisation de ce compte est conseillée par rapport au compte root car il a accès à certaines fonctions qu'après une confirmation ce qui peut permettre d'éviter des erreurs. Nous avons choisi de créer un compte qui a les caractéristiques :

- Nom complet : Etudiant
- Identifiant (login) : etudiant
- Mot de passe : etudiant

Le programme nous demande ensuite de quelle façon nous voulons faire les mises à jour du système. Comme nous disposons d'un accès à Internet par la passerelle précédemment configurée, nous choisissons le protocole http avec le miroir France et la première adresse que le programme nous propose : `ftp://ftp2.fr.debian.org`, la configuration d'un proxy peut-être faite à ce moment, nous n'avons pas de tel équipement à configurer. Pour terminer la configuration du système, nous ne demandons pas de programme à installer ni de configuration du logiciel Exim v4 (Serveur de Mail SMTP).

Configuration du système

Le système est maintenant prêt à démarrer et nous demande de nous identifier, nous utilisons le compte super utilisateur pour cela afin d'avoir un maximum de possibilité sur le système, nous sommes conscients que ce mode d'utilisation comporte des risques et qu'il ne nous demande peu de confirmation, en cas de suppression de fichiers importants par exemple.

Nous accédons alors à un shell et nous pouvons effectuer différents tests en particuliers pour vérifier que les paramètres réseaux sont bons. Nous effectuons en premier lieu des commandes permettant de vérifier l'état du réseau tel que :

`ping -b 172.31.20.255` *Demande à toutes les machines présentes sur le réseau de répondre à un paquet et calcule le temps de réponse,*

`arp -a` *Donne les adresses présentent dans la table ARP,*

`ifconfig -a` *Présente la configuration réseau de l'ordinateur, l'option -a permet de lister les interfaces qui ne sont pas actives.*

Interconnexion avec les autres ordinateurs de la salle :

Maintenant que nous avons vu comment installé et configurer le système d'exploitation de notre serveur, nous allons voir comment le connecter aux machines présentes sur le réseau local des autres serveurs ainsi que celle qui dépendent de notre réseau : les postes clients.

Nous avons encore deux interfaces disponibles sur notre ordinateur, une pour chaque réseau. Pour configurer une des interfaces, il faut modifier le fichier

`/etc/network/interfaces` avec un éditeur de texte comme `VI` ou `nano`. Nous préférerons le second car il nous apparaît comme plus convivial. Il peut être lancé par la commande :

```
nano /etc/network/interfaces
```

Pour ajouter une nouvelle interface, il faut se placer à la fin du fichier et entrer les lignes suivantes :

```
auto eth0
iface eth0 inet static
     address 192.168.0.5
     netmask 255.255.255.0
     network 192.168.0.0
     broadcast 192.168.1.255
```

La première ligne permet de démarrer automatiquement cette interface au démarrage du système. Les lignes suivantes définissent la manière dont les cartes sont configurées. Ici nous avons choisi une configuration statique pour les interfaces en paramétrant l'adresse IP, le masque réseau, le réseau et l'adresse de diffusion (broadcast).

Pour les autres interfaces réseaux, il faut de plus ajouter les informations suivantes :

```
auto eth1
iface eth1 inet static
     address 192.168.5.1
     netmask 255.255.255.0
     network 192.168.5.0
     broadcast 192.168.5.255

auto eth2
iface eth2 inet static
     address 172.31.20.25
     netmask 255.255.255.0
     network 172.31.20.0
     broadcast 172.31.20.255
```

Une fois le fichier modifié, il faut relancer le réseau en redémarrant l'ordinateur ou en lançant la commande :

```
/etc/init.d/networking restart
```

Cette dernière permet de relancer les services réseaux sans avoir à redémarrer l'ordinateur. La vérification de la configuration de la carte réseau peut se faire en tapant les commandes :

```
ifconfig -a
ping <adresse d'un autre PC sur le réseau>
```

La première commande permet de savoir si la configuration est activée sur la carte. Si l'interface configurée n'a pas les adresses définies, il peut y avoir un

problème dans l'écriture du fichier comme une faute de frappe par exemple. La seconde permet d'atteindre une autre machine sur le réseau, si aucune réponse n'est fournie cela peut venir d'un problème matériel (câble non branché ou défaillant, carte réseau endommagée…). Une commande `ping` peut être lancée sur un autre poste pour vérifier la communication réciproque des postes.

Nous avons choisi une adresse statique pour le réseau entre client et serveur et entre les serveurs. Le plan de configuration réseau est présenté page suivante (Figure 1).

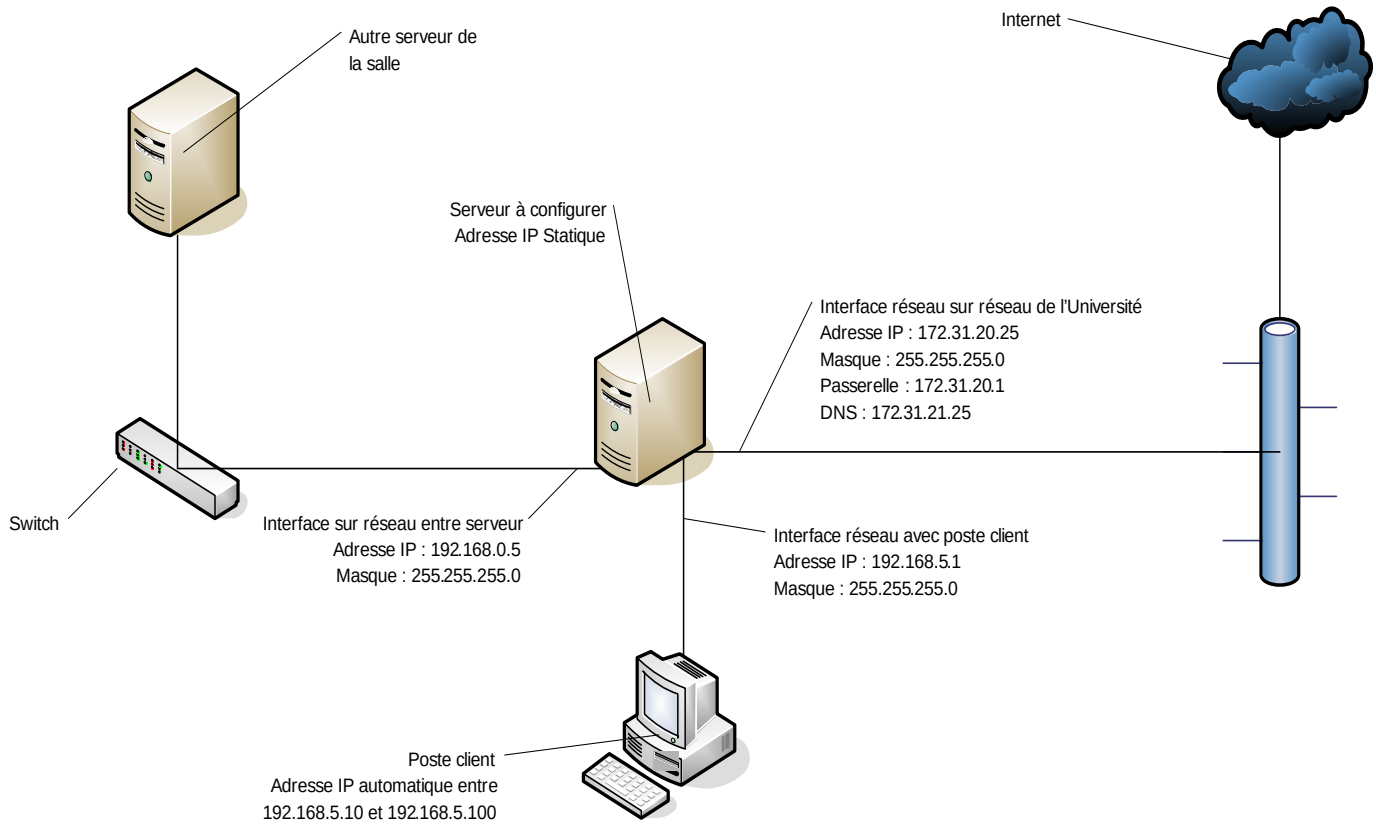

Figure 1 - Plan du réseau

Légende : affichage du plan réseau de l'installation avec les plages et adresses IP des différentes interfaces et réseaux.
Note : le poste client est en fait représentatif d'un réseau de machine, comme l'indique sa plage d'adresse attribuée dynamiquement par le serveur DHCP.

Pour que les postes clients aient accès aux postes reliés aux autres serveurs, il faut configurer des « routes » réseaux sur le poste serveur qui sert de passerelle. Les routes peuvent être ajoutées par la commande :

```
route add –net <adresse du réseau> netmask <adresse du masque réseau> gw <adresse du serveur du réseau> dev <nom de l'interface reliée au réseau>
```

Chaque réseau de la salle est accessible ainsi, les routes qui sont ajoutées de cette manière sont automatiquement placées dans la configuration du serveur et resteront actives après redémarrage de l'ordinateur. Cette commande doit être répétée pour chaque réseau.

Pour que le transfert des paquets des clients vers d'autres machines soit activé, il faut effectuer la commande :

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Les différents postes peuvent se voir, ce qui peut être testé en faisant plusieurs commandes `ping` vers et depuis différents postes déjà configurés.

La commande `iptables` permet d'autoriser ou de refuser les paquets à accéder à Internet, on peut ainsi fixer des règles de routage pour permettre à certains logiciels d'accéder ou non au Web. Dans un premier temps, nous avons ajouté des règles permettant à tous les programmes et à tous les postes d'accéder à Internet. Nous avons créé un fichier qu'il faut exécuter à chaque redémarrage. Pour configurer des tables automatiquement au démarrage, il faut placer cette configuration dans un fichier. Les commandes que nous avons en premier lieu entrées sont :

```
iptables -p INPUT ACCEPT
iptables -p OUTPUT ACCEPT
iptables -p FORWARD ACCEPT
iptables -F
iptables -t nat -F
iptables -t nat –A POSTROUTING -s 192.168.5.0/24 -j MASQUERADE
```

Ces réglages sont uniquement là pour exemple, car leur effet est de laisser tous les trafics autorisés, ce qui n'est pas la solution de sécurité recherchée au final. La version définitive est disponible en annexe 2.

Pour que le fichier puisse être interprété par le shell, il faut lui attribuer les droits d'exécution avec la commande :

```
chmod a+x iptables
```

Ces règles concernant les routes et les tables de filtrage peuvent être ajoutées dans un script qui s'exécutera au démarrage du système. Pour cela, nous créons un nouveau fichier dans lequel nous insérons les différentes commandes définies ci-dessus. Ce fichier est à placer dans le répertoire `/etc/init.d` et pour qu'il soit exécuté dans le mode de fonctionnement multi-utilisateur avec prise en compte du réseau, il faut faire un lien symbolique de ce fichier vers le répertoire `/etc/rc2.d`.

Ces niveaux d'utilisation correspondent chacun à un nombre de services démarré sur le système, le changement de niveau a donc pour conséquence le lancement ou l'arrêt de service de manière groupé. Ces scripts s'exécutent au lancement de l'ordinateur dans un ordre précis en lien avec le nom du fichier (une lettre puis deux chiffres qui le déterminent), il faut donc choisir le nom de notre script en fonction de ceux qui sont déjà présent. Comme le notre a besoin que les premiers services du noyau soient lancés, nous avons choisi un numéro qui vient après la plupart des scripts (à partir de 50 par exemple).

De plus, les scripts sont lancés par le système qui leur passe en paramètre l'argument "`start`", il faut donc placer les règles dans une structure de contrôle qui gère ce paramètre. Lors d'un changement de niveau d'utilisation, le système lance les scripts en leur passant l'argument `stop`, il faut donc aussi gérer ce cas. Cette gestion peut se faire par la structure `switch – case` par exemple.

Tous les utilisateurs du système doivent avoir les droits d'exécution sur ce fichier, il est important de mettre ces droits avec la commande :

```
chmod a+x <nom du fichier>
```

Pour tester ce script, il est possible de les exécuter directement en faisant :

```
. /<chemin et/ou nom du fichier>
```

Si des routes ont déjà été ajoutées, il faut redémarrer l'ordinateur afin de les désactiver, on peut le faire avec la commande `reboot` qui ne demande pas de confirmation lorsque l'on est connecté en root. Après le redémarrage, il est possible de voir les tables en demandant la commande `route`, les différentes routes que l'on a définies dans le script. On peut aussi essayer d'accéder à Internet depuis le client, s'il y a accès les tables de filtrage sont opérationnelles. Il est possible d'affiner ces filtrages en restreignant l'accès à Internet qu'à certains programmes, nous verrons plus tard comment le faire après les avoir installés.

Installation et configuration de nouveaux logiciels

Sous les environnements Unix, les programmes peuvent être installés de différentes façons. Nous allons détailler dans ce rapport la procédure à suivre pour installer à partir de « package » puis à partir des sources du logiciel.

Installation

Grâce au système des packages

L'installation et la mise à jour de package se fait en plusieurs étapes. La première consiste à mettre à jour la liste des logiciels que l'on peut télécharger sur le miroir de mise à jour que l'on a choisi, cela se fait avec la commande :

```
apt-get update
```

Il est possible ensuite de mettre à jour les packages qui sont installés sur le système ainsi que les drivers grâce à commande :

```
apt-get upgrade
```

Cette commande peut demander un redémarrage de l'ordinateur et terminer les installations.

Pour installer de nouveaux packages, il faut en premier lieu connaître le nom exact du package à installer, il peut être différent du nom général du logiciel car il est possible que la version que l'on veuille installer comporte un nom spécifique. Pour connaître les packages en lien avec celui que l'on veut installer, on peut entrer la commande :

```
apt-cache search <nom du package>
```

Elle permette d'avoir une idée des package en rapport avec celui recherché dans la liste que nous avons mise à jour précédemment. Une fois que l'on sait lequel des ces logiciels est à installer, il suffit d'entrer la commande :

```
apt-get install <nom du package>
```

Le système télécharge automatiquement ce qu'il a besoin pour le logiciel et commence automatiquement le processus d'installation. Si le package a des dépendances ou s'il contient d'autres logiciels nécessaires à celui que nous voulons installer, le logiciel nous demande si l'on veut les installer. Refuser risque de poser des problèmes dans l'utilisation du logiciel voulu.

Nous avons utilisé cette procédure pour installer les logiciels `traceroute` (outils permettant de connaître le chemin d'un paquet réseau), `links2` (navigateur en mode texte), `host` (donne le nom du serveur et l'adresse IP d'une adresse) et

`ssh` (permet d'ouvrir une session sur un poste à distance). Lors de l'installation d'un package, il se peut que le processus demande s'il doit remplacer un package précédent par celui que l'on veut installer pour éviter les conflits, ce fut le cas pour `host`, nous avons du désinstaller `bind9-host` pour l'installer.

Exemple d'utilisation de ces logiciels :

`traceroute http://www.google.com` *Affiche la liste des sauts à faire pour atteindre l'adresse voulue*

`links2 http://www.google.com` *Affiche la page Internet voulue en mode texte*

`host http://www.google.com` *Donne le nom du serveur et l'adresse IP correspond à l'adresse passée en paramètre*

Lors de l'installation de `ssh`, un processus plus important est lancé pour aider à sa configuration. Lors de la première étape, il nous est demandé si nous voulions installer seulement la version 2 ce auquel nous répondons par la négative pour avoir les deux versions. Puis si nous voulons utiliser le principe de bit SETUID qui permet d'être identifiée automatiquement sur certaines machines grâce à un échange de clés cryptées uniques ce qui peut-être intéressant et que nous acceptons. L'installation demande si nous voulons installer le serveur `ssh` qui sera utile si nous voulons paramétrer le serveur à distance.

Une fois ce logiciel installé, on peut tester son utilisation par la commande :

`ssh login@172.31.21.34` *Connexion au serveur Kundera de l'université avec le login choisi.*

L'ordinateur que nous cherchons à joindre peut être protégé par mot de passe. Si l'on se connecte sur l'adresse donnée, nous accédons au contenu de notre répertoire présent sur les ordinateurs de l'université.

Il est possible de supprimer un package installé grâce à la commande :

```
apt-get remove <nom du package>
```

De même, on peut consulter la liste des packages installés sur le système par l'exécution de la ligne :

```
dpkg -l
```

Lors des TP, nous avons utilisé ce système pour installer DHCP, NFS et NIS. Leur configuration sera abordée dans la partie suivante.

A partir des sources

Lors de nos TP, nous avons installé plusieurs programmes qui sont fréquemment utilisés sur les serveurs Unix en entreprise, c'est-à-dire Apache

(serveur HTTP), PostgreSQL (système de gestion de base de données) et un interpréteur PHP. Ces logiciels forment un ensemble permettant de diffuser sur Internet des pages web, Apache, dont le contenu dépend de données contenues dans la base, PostgreSQL, qui est elle-même interrogée par un langage de programmation, PHP.

Ces différents logiciels peuvent s'installer de manière similaire en suivant une procédure « type », il existe pour certains des différences lors de certaines étapes que nous traiterons au cas par cas ensuite. La méthode d'installation peut se résumer ainsi :

Téléchargement des sources compressées
Décompression de l'archive
Configuration des options pour la compilation
Compilation du programme
Installation des fichiers à l'emplacement prévue

Enfin nous détaillerons dans la partie suivante ce qu'il faut réaliser pour configurer et utiliser ces programmes.

Procédure d'installation générale

L'installation d'un programme avec Debian sans passer par les packages nécessite d'avoir les sources du logiciel que l'ont peut généralement télécharger sur Internet à l'adresse du distributeur. Pour les logiciels qui nous intéressent, les adresses où l'on peut télécharger les sources sont :

`http://httpd.apache.org/download.cgi`	Apache
`http://www.postgresql.org/ftp/binary/`	PostgreSQL
`http://www.php.net/downloads.php`	PHP

Différentes versions sont généralement disponible pour chaque logiciel et correspondent à des ajouts de fonctionnalité ou des corrections de bugs. Il faut alors choisir une version qui répond à nos besoins. Pour Apache et PostgreSQL, les différentes versions correspondent à des différences dans le développement et dans les fonctions apportées. Pour PHP, on ne peut en général télécharger uniquement la dernière version car elle correspond à la version la plus stable et sécurisée qui existe, les versions précédentes présentant des problèmes ne sont plus proposées.

Nous avons choisi les versions :

1.3.37 pour Apache
8.2.3 pour PostgeSQL
5.2.2 pour PHP (dernière version disponible pour php5 au moment de la configuration du serveur)

Une fois que les sources ont été téléchargées, il est préférable de les regrouper dans un même répertoire pour les retrouver facilement en cas d'erreur. Nous avons choisi de créer un répertoire `/src` directement à la racine de l'arborescence.

Les archives sont donc copiées dans ce dossier et sont décompressées à l'aide de la commande :

```
tar xzvf <nom complet de l'archive>
```

Les lettres qui suivent "tar" correspondent à l'ordre que nous voulons. De manière détaillée, nous pouvons détailler les différentes options utilisées :

x	Extraction des fichiers
z	Décompression des fichiers avec gzip
v	Détaille les fichiers extraits
f	Précise que l'ont passe le fichier à extraire ensuite

L'étape suivante correspond à la configuration de la compilation et de l'installation finale. Cette étape est réalisée à l'aide de la commande `configure` auquel on passe différentes options permettant par exemple de définir le chemin où sera installé le programme en fin de processus, de lier certains programme entre eux, d'utiliser ou de ne pas utiliser telle ou telle bibliothèque pour le programme…

Les options sont propres à chaque programme, on peut tout de même indiquer l'option générique qui permet de définir l'emplacement d'installation :

```
--prefix=<chemin vers le répertoire>
```

La commande `configure` vérifie que les programmes dont dépend celui que l'on va installer sont présents sur le système. Il arrive en effet qu'il manque une bibliothèque de compilation nécessaire au bon fonctionnement du programme. Il est alors possible d'installer le programme grâce au système de package développer plus haut. Lors de l'installation des programmes, nous avons eu à installer les packages

Après cette configuration, il est dès lors possible de lancer la compilation grâce à la commande :

```
make
```

Le système nous affiche alors nombre de message, la compilation est en cours et demande un certain temps pour s'exécuter entièrement. A la fin de l'opération, le programme nous annonce le succès de l'opération, nous n'avons alors plus qu'à effectuer la dernière opération à l'aide de la commande :

```
make install
```

Cette dernière étape s'exécute plus rapidement que la précédente et permet de recopier les fichiers qui ont été compilés à l'emplacement que l'on a défini lors de la configuration.

Configuration des programmes installés

Après avoir vu comment installer des programmes grâce aux packages et à partir de leur source, nous allons voir la configuration de chacun des services installés.

DHCP

La recherche sur le package DHCP nous renvoie un large choix de programme, nous avons choisi `dhcp3-server`. Lors du lancement du package, nous avons précisé l'interface à utiliser, dans notre cas `eth1` connectée aux clients, puis validé pour terminer l'installation.

La configuration du service DHCP passe par la modification du fichier `dhcpd.conf` qui se trouve dans le répertoire `/etc/dhcp3/`. Il faut alors ajouter une nouvelle plage d'adressage automatique :

```
subnet 192.168.5.0 netmask 255.255.255.0
{
     range 192.168.5.10 192.168.5.100 ;
     option routers 192.168.0.5 ;
     option domain-name-servers 172.31.21.35 ;
}
```

On lance alors le serveur DHCP avec la commande :

```
/etc/init.d/dhcp3-server restart
```

Pour tester l'attribution automatique d'une adresse IP sur les clients, il faut redémarrer une machine cliente pour vérifier la bonne configuration de la carte réseau.

Il est aussi possible de vérifier à partir du serveur qu'une adresse a été attribuée en détaillant les dernières lignes du fichier `/var/log/messages`.

Apache

Nous allons maintenant traiter Apache. Pour l'installer, il faut créer un dossier qui accueillera les fichiers compilés, nous avons choisit le répertoire `/usr/local/apache-1.3.37`. Les options de compilation sont :

```
. /configure --prefix=<répertoire où installer>
     --enable-module=so
```

Une fois cette configuration effectuée, il faut faire la commande `make` puis `make install`. Les fichiers compilés seront alors copiés dans le répertoire défini lors de la configuration, pour nous : `/usr/local/apache-1.3.37`.

Pour configurer le démon httpd, il faut éditer le fichier dont le chemin est pour nous : `/usr/local/apache-1.3.37/conf/httpd.conf`. Les différentes modifications que nous avons apportées sont :

Enlever le commentaire de la ligne contenant `BindAdress *`, correspond aux adresses IP pour lesquels le serveur réagira

Attribuer le port d'écoute : `Port 80`, d'autres ports peuvent ici être définis

Définir l'utilisateur (ainsi que son groupe) qui sera utilisé pour lancer le démon httpd avec :

User nobody
Group nobody

Nous proposons l'utilisateur `nobody` que nous créons uniquement pour ce logiciel. L'utilisateur en question doit avoir les droits sur les fichiers qu'il aura à utiliser, il faudra donc lui attribuer.

Entrer l'adresse de l'administrateur :

`ServerAdmin admin@server.com`

Fixer le répertoire qui sera le répertoire de base du site Internet

`DocumentRoot /usr/local/www`

Pour ajouter l'utilisateur et le groupe lui correspondant, il faut en premier lieu ajouter le groupe grâce à la commande :

```
groupadd nobody
```

Ensuite pour créer l'utilisateur et lui affecter le groupe, on peut utiliser :

```
adduser nobody nobody
```

Il peut être intéressant d'avoir un lien symbolique vers le répertoire qui contient la version d'Apache que l'on utilise, de manière à ne pas changer d'autres paramètres que ce lien symbolique lors d'un changement de version. Ce lien est fait par la commande :

```
ln -s /usr/local/apache-1.3.37 /usr/local/apache
```

Le service est maintenant prêt à être lancé grâce au script et au paramètre :

```
/usr/local/apache/bin/apachectl start
```

Nous n'avons pas affecté de nom de domaine lors de la configuration, le service nous prévient de ceci par message.

PostgreSQL

Après avoir installé Apache, nous avons installé le Système de Gestion de Base de Données PostgreSLQ. Pour ce programme, nous avons utilisé les options de compilation suivante :

```
--prefixe=/usr/local/postgesql-7.4.5
--with-java
--enable-thread-safety
```

Lorsque de cette configuration, le programme nous a averti qu'il manquait des programmes nécessaires pour l'utilisation de PostgreSQL : `readline` et `zlib`, nous avons utilisé le mécanisme de package pour les installer. Plusieurs packages pouvaient correspondre à la désignation que nous avions des programmes, nous avons choisi respectivement `libreadline5-dev` et `zlib1g-dev`, la présence de "`-dev`" dans leur désignation nous permet de savoir que ces packages peuvent être utiles lors de développement de logiciels et donc lors de la compilation d'autres programmes. Une fois ces packages installés, la compilation et l'installation a pu avoir lieu comme décrit plus haut.

Après cette première étape, nous avons configuré le programme afin de lancer le service. Le démon PostgreSQL se lance avec un utilisateur particulier : `postgres`, il faut donc en premier lieu vérifier qu'il a bien été créé lors de l'installation en changeant d'utilisateur avec la commande :

```
su postgres
```

Ou en vérifiant dans le fichier `/etc/passwd` s'il y est. Cela peut être fait avec la commande :

```
cat /etc/passwd | more
```

S'il n'y est pas, comme ça a été le cas pour nous, il faut le créer comme pour l'utilisateur nobody pour Apache avec la commande :

```
adduser postgres
```

Il est dès lors possible de changer d'utilisateur avec la commande ci-dessus. Il est d'ailleurs conseillé d'avoir deux terminaux afin de ne pas avoir à changer sans cesse d'utilisateur lors des manipulations qui vont suivre. L'ajout d'un terminal se fait en appuyant sur la combinaison de touches "Alt + F#", # représentant le numéro de terminal à créer. Nous pouvons ensuite passer de l'un à l'autre en utilisant "Alt + tab".

La première étape de configuration à effectuer pour PostgreSQL est d'effectuer la commande en utilisateur postgres :

```
make check
```

Là où sont placées les sources pour vérifier si l'ensemble des fonctionnalités du programme ne pose aucun problème. Le programme vous confirme qu'aucun problème n'a été décelé, nous pouvons dès lors continuer la configuration. Il se peut que l'utilisateur postgres n'ait pas les droits sur le dossier des sources. On peut le rendre propriétaire de ce dossier en utilisateur root grâce à la commande :

```
chown -R /src/postgres-8.2.3
```

Il est aussi préférable de créer un lien symbolique comme pour Apache, pour simplifier les prochaines étapes, la commande pour réaliser cela est :

```
ln -sf /usr/local/postgres-8.2.3 /usr/local/postgresql
```

Le paramètre `-f` permet d'effacer celui qui existait s'il existait. Son utilisation est donc à préconiser en particulier lors d'un changement de version.

Pour pouvoir lancer le service PostgreSQL, il faut en premier lieu initialiser la base de donnée à partir d'un script du programme. Pour cela il faut créer le dossier qui contiendra les bases de données grâce à la commande :

```
mkdir /usr/local/postgresql/data
```

Il faut rendre l'utilisateur postgres propriétaire de ce dossier comme précédemment puis lancer le script d'initialisation du dictionnaire de données avec ce même utilisateur grâce à la commande :

```
/usr/local/postgresql/bon/initdb -D
```

Si aucun problème ne survient, le script nous indique que tout s'est bien déroulé en nous affichant "`Success`" suivit des chemins à partir desquels il est possible de lancer le service. Si on essaie de lancer le service maintenant, il devrait répondre par un message d'erreur nous précisant qu'aucun fichier de log n'a été créé. Pour cela, il faut repasser en mode super utilisateur, se placer dans le répertoire `/var/log` (grâce à `cd`) et créer un fichier vide avec la commande :

```
touch postgres
```

Un ficher postgres est ainsi créé, il faut ensuite l'affecter à l'utilisateur postgres avec :

```
chown postgres postgres
```

Il est alors possible de lancer le service postgres avec la commande :

```
/usr/local/postgresql/bin/postmaster –D \
    /usr/local/postgresql/data start
```

Le programme nous indique que le serveur démarre par un message. Lorsque le serveur est démarré, il est possible de vérifier la présence du processus grâce à la commande :

```
ps -ef
```

On peut constater l'apparition de nouveau processus liés à PostgreSQL. Il est possible d'accéder à un terminal en lançant le programme :

```
/usr/local/postgresql/bin/psql
```

On peut alors vérifier le bon fonctionnement du service en manipulant le concept de base de données, c'est-à-dire : créer une table, insérer des données, effectuer des requêtes…

PHP

Enfin, après avoir installé PostgreSQL, nous allons voir la procédure à suivre pour implémenter PHP, le service chargé d'interpréter les pages Internet qui comportent du code php.

Nous avons utilisé les options de configuration suivantes :

```
--prefix=/usr/local/php-5.2.1
--with-psql=/usr/local/postgresql
--with-apsx=/usr/local/apache/bin/apsx
--with-zlib --with-jpeg --with-png --with-gettext
--with-gd2
```

Comme pour les programmes précédents, la première permet de fixer le répertoire où sera installé le programme une fois compilé. Les options suivantes définissent comment sera utilisé PHP par rapport aux autres logiciels installés mais aussi par rapport à ce qu'il est possible de définir avec php. La liste complète des options supportées par les versions 3 et 4 de PHP est disponible à cette adresse : `http://dionysos.univ-lyon2.fr/~miguet/docPhp4/install.configure.html`.

Lors de cette première étape de configuration, le programme nous a averti qu'il manquait xml2 pour que l'installation puisse s'opérer. Nous avons alors obtenu ce programme grâce au mécanisme des packages en choisissant celui nommé `libxml2-dev`. On peut alors recommencer la configuration puis, si aucun autre problème ne survient, demander la compilation avec `make` puis l'installation avec `make install`.

La configuration de PHP passe par la modification du fichier de configuration du serveur httpd qui se trouve au chemin : `/usr/local/apache/conf/httpd.conf`. Il faut alors ajouter la ligne :

```
AddType aplication/x-httpd-php .php
```

Vers les autres lignes concernées par le AddType. Pour que ce changement soit pris en compte, il est nécessaire de redémarrer le serveur Apache grâce aux commandes :

```
/usr/local/apache/bin/apachectl stop
/usr/local/apache/bin/apachectl start
```

Il est possible de tester les programmes que nous avons installés en créant un fichier `index.php` là où nous avons choisi de placer nos pages, dans notre cas `/usr/local/www`. Son contenu (édité avec `nano` par exemple) sera :

```
<html>
<head>
<title>Page de test</title>
</head>
<body>
<h1>Hello World !</h1>
<?php phpinfo() ?>
</body>
</html>
```

Cette page, accessible en utilisant `links2` avec la commande `links2 http://localhost/index.php` devrait afficher le message de bienvenue et les informations de la version de PHP dans le cas où tous les modules Apache et PHP sont correctement actifs.

Lors de la première installation de ces programmes, nous ne sommes pas arrivés à faire fonctionner Apache de manière à ce qu'il interprète les commandes PHP. Nous avons repris ces manipulations plus tard avec une nouvelle version de PHP (passage de la version 5.2.1 à 5.2.2) en ne changeant rien. Le code PHP est alors été interprété. Le changement de version a pu corriger des problèmes liés à notre version de Debian puisque nous n'avons pas fait d'autres modifications.

NFS

Le service NFS (Network File System) permet le partage de données entre un serveur et un client. Ce service est particulièrement efficace lorsque les utilisateurs risquent de changer de poste à l'intérieur d'un réseau, ils pourront dès lors retrouver leurs fichiers sur n'importe quelle machine reliée au serveur.

Nous avons choisi d'installer le service NFS grâce aux packages. Nous avons installé les packages :

`portmap`	gère les appels de procédures à distance (RPC),
`nfs-common`	gère les transactions de type NFS avec le client,
`nfs-kernel-server`	noyau du serveur.

Les deux packages étant déjà installés, nous avons alors du le configurer avant de le lancer. En premier lieu, il faut créer ou choisir un répertoire qui accueillera les fichiers partagés, la commande pour créer un répertoire est :

```
mkdir <nom du répertoire>
```

La configuration du serveur NFS passe par la modification du fichier `/etc/exports` dans lequel on ajoute la ligne correspondant au répertoire qui sera partagé suivit de paramètres. Nous avons choisi de créer un répertoire "partage" à la racine de l'arborescence et de partager ce répertoire en lecture et écriture à tous les postes du réseau. Dans le but de faciliter le fonctionnement de NIS, il est préférable de partager également le répertoire `/home` afin que chaque utilisateur retrouve son environnement de travail sur les postes clients. Les lignes à ajouter sont donc pour nous :

```
/partage 192.168.5.0/255 (rw)
/home 192.168.5.0/255(rw)
```

Les machines du réseau sont identifiées par le CIDR qui leur correspond.

Nous pouvons alors lancer le serveur pour le tester. La commande pour le lancement est :

```
. /etc/init.d/nfs-kernel-server start
```

On peut alors tester sur le client en montant le répertoire partagé dans le système de fichier. Cela se fait en modifiant sur le client le fichier `/etc/fstab` qui regroupe les différents périphériques qui seront montés au démarrage (disquette, CD-ROM, …). Nous rajoutons donc la ligne :

```
192.168.5.1 :/partage    /home2    nfs  defaults  0    0
192.168.5.1 :/home       /home     nfs  defaults  0    0
```

Le premier champ est celui du point de montage, c'est-à-dire là où existe le répertoire d'origine. Pour nous, cela correspond à l'adresse du serveur suivit du répertoire partagé.

Le second champ correspond au répertoire où sera monté le partage sur le client, il nous a fallut créer ce répertoire avant de le monter.

Le troisième paramètre correspond au type de système que l'on monte, il peut être automatique ou même ignoré dans certain cas, ici de type NFS.

Le quatrième paramètre correspond à des options de montage comme le type de système de fichier, nous utiliserons les options par défaut.

Les champs cinq et six correspondent respectivement à la fréquence de sauvegarde et à l'ordre de vérification du système de fichier. Nous ne voulons pas de sauvegarde et laissons la valeur par défaut pour la vérification.

Pour réinitialiser la table de montage présent dans ce fichier, nous utiliserons la commande :

```
mount -a
```

Nous pouvons alors vérifier que si un fichier est créé sur le client, il apparaît sur le serveur. Le partage et la sauvegarde des fichiers sont alors opérationnels. Attention toute fois à ne pas oublier d'autoriser les droits de lecture/écriture à tous les utilisateurs sur le répertoire partagé du serveur.

NIS

Comme pour le programme NFS, nous avons choisi d'installer NIS grâce au système des packages, celui que nous installons s'appelle `nis`. Il nécessite au préalable l'installation d'autres packages puis démarre un programme d'installation complet où en premier lieu, nous avons à entrer le nom du domaine. Nous choisissons `SRII-5` compte tenu de notre position dans la salle et du numéro de notre groupe. Le programme nous informe que nous avons d'autres paramètres à faire après l'installation.

Nous allons récapituler les différentes modifications à effectuer pour paramétrer le serveur en indiquant le fichier sur lequel faire ces changements et en expliquant dans la mesure du possible à quoi ces fichiers correspondent:

- `/etc/default/nis` :

```
NISSERVER=master
NISCLIENT=true
```

Indique que le poste correspond au serveur mais sera aussi son propre client.

- `/etc/defaultdomain` :

```
SRII-5
```

Correspond au nom de domaine du serveur par défaut.

- `/etc/yp.conf` :

Ajout de la ligne :

```
domain    SRII-5    server    localhost
```

Permet à la partie cliente de NIS présente sur le poste de trouver quel est son domaine, cette modification sera à faire sur chacun des postes clients.

- `/etc/ypserv.securenets` :

Commentaire de la ligne :

```
0.0.0.0             0.0.0.0
```

Et ajout de la ligne :

```
255.255.255.0       192.168.5.0
```

Précise quelles sont les adresses qui recevront les identifiants grâce à NIS plutôt que de permettre à toutes les machines de les recevoir.

Nous pouvons alors correctement démarrer le serveur grâce à ces changements avec la commande :

```
/etc/init.d/nis start
```

L'initialisation du service doit être faite afin d'assurer son fonctionnement à l'aide de la commande :

```
/usr/lib/yp/ypinit -m
```

Il est alors possible d'ajouter un nouvel utilisateur et de lui affecter un répertoire de travail directement grâce à la commande :

```
adduser –home /home/<nom> <nom>
```

Le système nous demande le mot de passe que nous souhaitons pour cet utilisateur ainsi que d'autres informations telles que son nom, son numéro de téléphone, etc…

La mise en correspondance des utilisateurs du système et de NIS se fait en exécutant :

```
/var/yp/make
```

Pour s'identifier sur le client, il faut modifier le domaine du système en éditant le fichier `/etc/yp.conf` et ajouter la ligne :

```
domain    SRII-5          server          192.168.5.1
```

Il faut de plus modifier le fichier `/etc/sysconfig/network` pour préciser le domaine NIS en inscrivant :

```
NISDOMAIN=SRII-5
```

Lors de ces deux réglages, il faut commenter les autres options déjà existantes. Dans notre exemple, les clients étant partagés, il est préférable de ne pas détruire le travail des autres utilisateurs.

Scripts exécutés au démarrage

Pour que les règles de routages et de filtrages ainsi que les services que nous implantons sur le serveur soient actifs automatiquement lors du démarrage, nous devons créer des fichiers de scripts et les placer dans le répertoire adéquat pour que le système les reconnaisse.

Nous avons choisi de placer tous les fichiers que nous implantons dans le répertoire `/etc/init.d` et de créer un fichier par service, nous obtenons donc les fichiers `route`, `iptables`, `apache` et `postgresql`. Les services DHCP, NIS et NFS, du fait de leur installation par packages, sont automatiquement insérés dans les scripts de démarrage. Le cas de PHP ne nécessite pas de script car il est démarré automatiquement par Apache.

Les scripts sont normalisés pour être utilisés par les commandes standards `start`, `stop` et `restart`. Ils suivent donc un modèle qui peut être résumé ainsi :

```
#!/bin/bash

case "$1" in
        start)
                #instruction de lancement
                ;;
        stop)
                #instruction d'arrêt
                ;;
        restart)
                $0 stop
                $0 start
                ;;
        *)
                exit 1
esac

exit 0
```

Les scripts `route` et `iptables` sont présents en annexe. Le script d'`apache` est bâti sur le modèle ci-dessus, les instructions de lancement et d'arrêt sont :

```
/usr/local/apache/bin/apachectl start #démarrage
/usr/local/apache/bin/apachectl stop  #arrêt
```

Le script pour PostgreSQL pose quelques difficultés, étant donné que ce service doit être lancé avec l'utilisateur postgres et non en mode root. Pour cela, il existe un script de démarrage fourni avec les sources. Conformément à notre installation, les options à modifier dans ce script sont :

```
prefix=/usr/local/postgressql
PGDATA="/usr/local/postgressql/data"
PGUSER=postgresql
PGLOG="/var/log/postgres"
```

Ouverture vers une solution en entreprise

Pour faire une installation de ce type, il faut normalement avoir une architecture différente. En effet, nous devrions avoir une passerelle qui se place entre l'extérieur (Internet) et notre réseau local. Cette passerelle contiendrait tous les règles de filtrages pour contrôler les entrées et sorties (le trafic depuis et vers Internet). D'autre part, il aurait fallut avoir un serveur qui gère DHCP et le routage directement relié à la passerelle. Selon l'installation des réseaux des "filiales" de l'entreprise, il est peut être nécessaire de placer une autre passerelle entre le réseau local et les réseaux de ces filiales. Le but de ces passerelles est de sécuriser notre réseau, car si elles n'existent pas et n'ont pas de pare-feu performant, le réseau peut être vulnérable.

Dans le cas idéal de leur placement dans le réseau, il faut également leur installer un système approprié comme une distribution de type BSD (OpenBSD par exemple). En effet, l'installation d'une Debian est relativement simple à effectuer, ayant pour conséquence une incertitude quant au contenu exact des différents programmes et compilateurs installés. Ce qui est dangereux car un utilisateur mal intentionné peut en profiter pour compiler ses propres programmes directement sur le système et allant jusqu'à mettre en danger le réseau. Pour cela il faut plutôt favoriser les systèmes à compiler, comme Gentoo, et non ceux avec un noyau générique comme Debian. De plus, dans le cas d'un tel système, si la configuration est optimale, on peut minimiser les demandes de ressources systèmes.

Pour les différents services à mettre en place, il aurait fallut idéalement placer un serveur pour chaque service. Soit un serveur dédié à Apache et PostgreSQL (ou Apache et MySQL, le cas échéant), un autre gérant les informations échangées entre l'intérieur et l'extérieur, la passerelle et un dernier qui supporte NIS (ou Samba) et NFS pour gérer les identifications et les transferts de fichiers. Nous pourrions éventuellement ajouter des serveurs de sauvegarde et de journalisations des évènements. L'avantage de cette configuration est que nous pouvons combiner plus de services sans rencontrer de problèmes, tel que l'incompatibilité de compilateurs dans le cas d'une installation sur un poste unique. De plus, ceci nous permet d'installer le système le plus approprié à chaque service.

Le coût de ce type d'installation est généralement élevé, mais il existe une solution alternative : l'installation sur le poste principale d'un serveur virtuel. Ce procédé permet d'installer plusieurs systèmes d'exploitations qui peuvent s'interconnecter entre eux et sont lancés par un unique système qui va les gérer. Cette solution permet de relativiser les risques en relançant rapidement les sauvegardes des serveurs corrompus. Mais le système hôte reste vulnérable, il ne faut donc pas négliger sa protection. De plus, cette méthode nécessite que la machine qui le supporte soit performante pour pouvoir lancer tous les systèmes et éviter la surcharge.

Pour mettre en place cette solution, il existe le logiciel VServer. Très intéressant car les systèmes émulés utilisent le noyau de l'hôte, il ne nécessite pas un espace disque virtuel, exploite le même système de fichier que l'hôte optimisant aussi l'espace disque occupé. Ceci peut alors être utilisé pour mettre en commun des fichiers de configuration et autre. Pour plus de détails sur le sujet, Wikipédia a un article qui l'aborde :

`http://fr.wikipedia.org/wiki/Vserver`

Nous avons donc tenté de faire ce type d'installation. Malheureusement, avec Debian, ce genre de service n'est pas incorporé par défaut (alors qu'il existe sous Solaris). Nous avons donc tenté de mettre à jour le noyau avec les composants adéquats :

`kernel-package` : les outils de compilation du noyau

`util-vserver` : les outils de manipulation des VServers

`vserver-debiantools` : contient des utilitaires permettant la création de VServers

Cette mise à jour peut à présent être faite. Pour cela, il nous faut télécharger leurs sources et les recompiler.

A partir des instructions données sur différents sites sur Internet, nous avons tenté d'effectuer ces manipulations. Mais il existe des différences sur les résultats escomptés et ceux que nous obtenons, la compilation du noyau ne peut pas aboutir. Cette tentative d'amélioration a du être abandonnée. Et nous avons conservé l'installation simple avec un unique système pour gérer tous les services.

Conclusion

Ces TP nous ont permis d'apprendre de nombreux mécanismes concernant Debian, en particulier l'installation de programmes grâce aux différentes méthodes d'installation par packages ou compilation des sources. Nous avons de plus mis en pratique l'aspect de configuration de différents services tel que DHCP, PostgreSQL, NIS…

La rédaction de ce rapport nous a permis de synthétiser les différentes opérations que nous avons effectuées. Nous l'avons conçu pour qu'il puisse être consulté et exploité par une personne n'ayant que peu de notions d'administration serveur mais aussi pour faire ressortir les choix que nous avons fait dans le but de faciliter les opérations de maintenance. Dans cet esprit, nous avons insisté sur les points clés de la mise en œuvre, des tests de bon fonctionnement et des opérations de résolution des problèmes possibles.

De plus, le fait d'établir un tel document nous permettra à l'avenir de nous y reporter lorsque l'on devra remettre en pratique ces notions. Ce peut être aussi pour nous un document de référence lors de rectification de paramètres.

Référence Internet

Debian Sarge :
`http://www.framasoft.net/article2603.html`
Configuration Apache :
`http://www.linux-kheops.com/doc/redhat72/rhl-rg-fr-7.2/s1-apache-config.html`
Installation et configuration PostgreSQL, script de démarrage :
`http://stessy.developpez.com/postgresql/installation/`
`http://www.postgresqlfr.org/?q=node/632`

Cours de M. Eric Leclerc - L3 Informatique - Dijon

Annexes

Annexe 1 – Script de démarrage route

```
#!/bin/bash
#
# Regles de routage
#

start() {
        #ajout des regles de routage

        #regle(s) pour le reseau IEM
        route add default gw 172.31.20.1 dev eth2

        #regle(s) pour le reseau d'interconnexion
        route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.0.1 dev eth0
        route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.0.2 dev eth0
        route add -net 192.168.3.0 netmask 255.255.255.0 gw 192.168.0.3 dev eth0
        route add -net 192.168.4.0 netmask 255.255.255.0 gw 192.168.0.4 dev eth0

        #regle(s) pour le reseau prive
        route add -net 192.168.5.0 netmask 255.255.255.0 dev eth1
}

stop() {
        #suppression des regles de routage
        route -f
}

restart() {
        #restoration des regles de routage
        stop
        start
}


case "$1" in
        start)
                echo "Lancement des regles de routage"
                start
                ;;
        stop)
                echo "Arret des regles de routage"
                stop
                ;;
        restart)
                echo "Redemarrage des regles de routage"
                restart
                ;;
        *)
                echo $"Usage : $0 {start|stop[restart}"
                exit 1
esac

exit 0
```

Annexe 2 – Script de démarrage iptables

```
#!/bin/bash
#
# Regles de filtrage
#

public="eth0"
prive="eth2"
inter="eth1"
adrprive="192.168.3.0/24"

start2() {
        #ajout des regles de filtrage
        iptables -P INPUT DROP
        iptables -P FORWARD DROP
        iptables -P OUTPUT DROP
        iptables -F
        iptables -t nat -F

        iptables -A INPUT -i lo -j ACCEPT
        iptables -A OUTPUT -o lo -j ACCEPT

        iptables -A INPUT -i $public -j ACCEPT
        iptables -A OUTPUT -o $public -j ACCEPT
        iptables -A FORWARD -i $public -j ACCEPT

        iptables -A INPUT -i $inter -j ACCEPT
        iptables -A OUTPUT -o $inter -j ACCEPT
        iptables -A FORWARD -i $inter -j ACCEPT

        iptables -A INPUT -i $prive -j ACCEPT
        iptables -A OUTPUT -o $prive -j ACCEPT
        iptables -A FORWARD -i $prive -j ACCEPT


        #translation d'adresse
        iptables -t nat -A POSTROUTING -s $adrprive -j MASQUERADE
}

start() {
        #ajout des regles de filtrage
        iptables -P INPUT DROP
        iptables -P FORWARD DROP
        iptables -P OUTPUT DROP
        iptables -F
        iptables -t nat -F

        iptables -A INPUT -i lo -j ACCEPT
        iptables -A OUTPUT -o lo -j ACCEPT

        #iptables -A INPUT -i $public -j ACCEPT
        #iptables -A OUTPUT -o $public -j ACCEPT
        iptables -A FORWARD -i $prive -o $public -j ACCEPT
        iptables -A FORWARD -i $public -o $prive -j ACCEPT

        #routeur vers Internet
        #iptables -A OUTPUT -o $public -p tcp --destination-port http -j ACCEPT
        #iptables -A INPUT -i $public -p tcp -m state --state ESTABLISHED,RELATED \
-j ACCEPT

        #reseau prive vers internet
        #iptables -A FORWARD -i $prive -o $public -p tcp --dport http -j ACCEPT
```

```
        #iptables -A FORWARD -i $public -o $prive -p tcp -m state --state \
ESTABLISHED,RELATED -j ACCEPT

        #reseau prive vers routeur
        iptables -A OUTPUT -o $prive -d $adrprive -j ACCEPT
        iptables -A INPUT -i $prive -s $adrprive -j ACCEPT

        #translation d'adresse
        iptables -t nat -A POSTROUTING -s $adrprive -j MASQUERADE
}

stop() {
        #suppression des regles de filtrage

        iptables -F
        iptables -t nat -F
}

restart() {
        #restauration des regles de filtrage

        stop
        start
}

case "$1" in
        start)
                echo "Lancement des regles de filtrage"
                start
                ;;
        start2)
                echo "Acceptation de tous les packets"
                start2
                ;;
        stop)
                echo "Arret des regles de filtrage"
                stop
                ;;
        restart)
                echo "Redemarrage des regles de filtrage"
                restart
                ;;
        *)
                echo $"Usage : $0 {start|stop[restart}"
                exit 1
esac



exit 0
```