



- Projet Master 2 Informatique -



Systeme de boot multiple hétérogène sur réseau

Tuteur :

Mr Hayel

Groupe 2 :

MOIROUX Etienne
JUANEDA Matthieu
LAWISKY Rodolphe



Présentation du projet

Description

Ce projet a pour objectif d'étudier un système de démarrage à distance, à travers un réseau local, sur plusieurs OS différents comme différentes versions de Linux (Ubuntu, Debian, Slackware, etc...) et Windows.

Ce projet expérimental pourra servir de maquette afin d'être déployé dans les salles réseaux du CERI.

L'objectif

Lors d'un TP réseau, le professeur a besoin pour gagner du temps sur le TP d'un certain nombre de postes avec des systèmes d'exploitation préinstallés avant le TP.

Pour le déroulement du TP dans les meilleures conditions, il y a certains prérequis :

- Les OS doivent être propres et optimisés.
C'est-à-dire sans configuration ou paramétrage particulier, sans aucun daemon lancés, etc... En gros, comme s'il venait d'être installé.
- Certains TP utilisent un certain nombre de packages d'outils ou d'applications (wireshark, dhcpd3, vlc, etc) qui peuvent être préinstallés sur chaque poste mais qui varient suivant les TP (Sécurité, Multimédia, Wifi, etc...)
- Les étudiants peuvent se loguer en root avec le même mot de passe pour chaque poste ainsi que disposer des droits root pour la configuration des clients.
- Les étudiants doivent aussi pouvoir accéder au disque dur local pour sauvegarder leur travail.

Notre projet est donc de centraliser sur un seul serveur un ou mieux plusieurs système(s) d'exploitation différents accessibles et bootable pour l'ensemble des clients du réseau.

Ces OS sont préinstallés et peuvent être modifiés (ajout ou suppression de package, etc...) sur un serveur.

Les postes clients auront donc la possibilité de choisir au démarrage parmi la liste des systèmes d'exploitation accessibles depuis le réseau ou parmi ceux installés sur le disque dur local.

Solutions techniques de boot en réseau

Qu'est-ce que le boot ?

Notre projet repose sur la possibilité pour un ordinateur de démarrer non pas sur une ressource locale, mais accessible depuis le réseau.

Petit rappel sur le « boot » :

L'ordinateur exploite un programme réduit (appelé Bootloader), permettant d'extraire un programme accessible via un périphérique de stockage permanent ou amovible. Ce dernier est typiquement le noyau du système d'exploitation, qui s'installera en mémoire vive et appellera lui-même des programmes applicatifs.

Sur PC, le BIOS se charge de lire le MBR (Master Boot Record), d'une taille de 512 octets, à partir de son lecteur de boot (généralement un disque dur, mais souvent aussi une disquette, un CDROM, ou une clef USB).

Cependant certains BIOS permettent aussi d'effectuer un démarrage PXE.

C'est ce mode opératoire que nous allons utiliser dans notre projet.

Démarrage PXE

PXE est l'acronyme de Pre-boot eXecution Enviroment.

C'est l'élément clef sur lequel repose tout notre projet.

PXE est donc un environnement regroupant plusieurs services / protocoles qui permet à une station de travail de démarrer depuis le réseau en récupérant une image de système d'exploitation qui se trouve sur un serveur.

L'image ainsi récupérée peut être le système d'exploitation brut ou bien le système d'exploitation personnalisé avec des composants logicielles (suite bureautique, utilitaires, packs de sécurité, scripts, etc...).

Une fois cette image "pré-chargée", elle peut éventuellement, en fonction des paramétrages passés à cette image sur le serveur, être installée sur la machine qui a booté en PXE.

Il permet également d'installer de manière automatique et à distance des serveurs sous divers OS.

Pour activer le PXE, il faut auparavant le configurer dans le BIOS si celui-ci le permet. L'option se trouve fréquemment dans un menu concernant la carte réseau.

Nous allons maintenant décrire l'ensemble des services à installer pour mettre en place un serveur pxe.

Premièrement, la station qui veut booter sur PXE doit avoir une adresse IP.

Elle va donc être attribuée par DHCP.

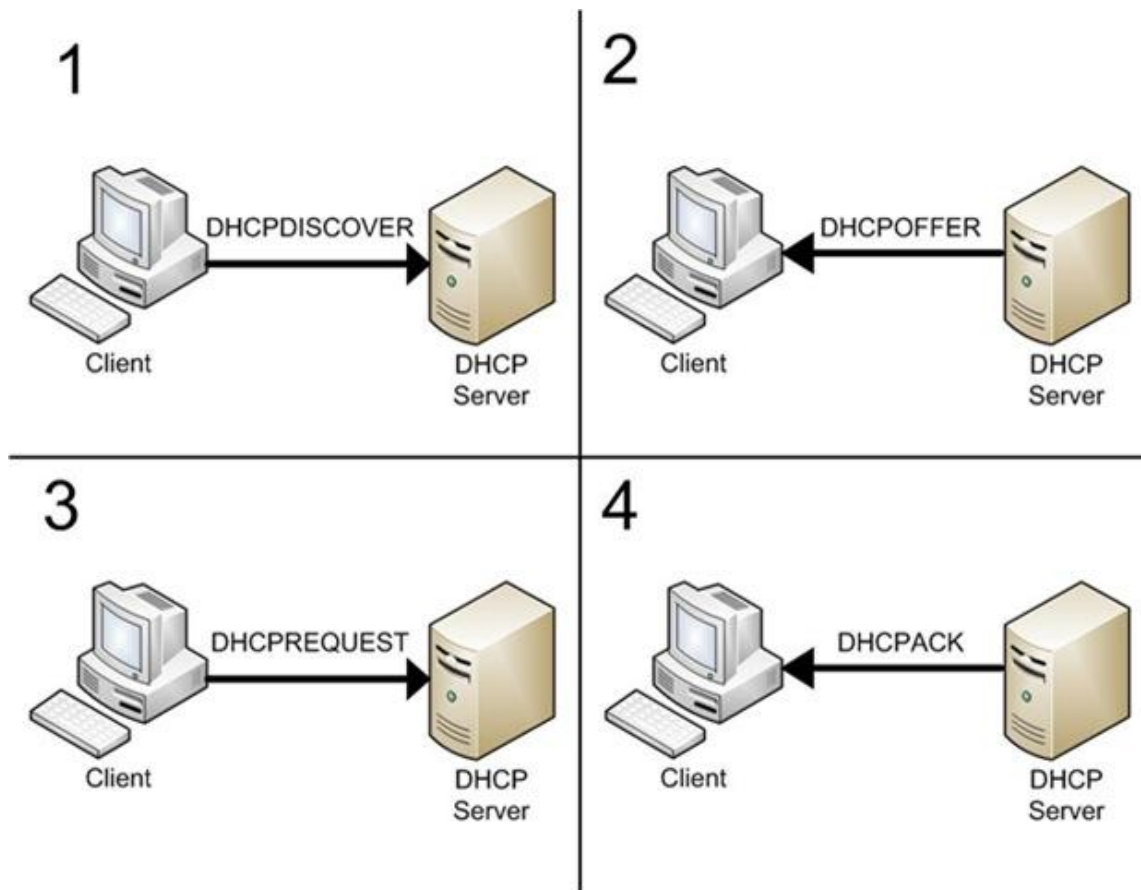
DHCP

DHCP (Dynamic Host Configuration Protocol) est un protocole réseau permettant à la base de configurer automatiquement les paramètres IP du client.

C'est-à-dire l'adresse IP du client, le masque de sous réseau, la passerelle par défaut et le ou les DNS.

Cela se déroule en 4 étapes

1. DHCPDISCOVER qui permet au client de découvrir les serveurs DHCP disponibles sur le réseau et demander une première configuration.
2. DHCPOFFER qui est la réponse du serveur, contenant les premiers paramètres.
3. DHCPREQUEST pour confirmer la demande d'attribution.
4. DHCPACK est la réponse finale du serveur, elle contient l'ensemble des paramètres IP et autres du client.



Ce service (dhcp3-server) va donc être installé sur notre serveur maquette. On le paramètre via le fichier dhcpd.conf.

DHCP va être aussi utile pour indiquer au client le nom d'un fichier et l'adresse du serveur TFTP sur lequel télécharger ce fichier une fois le paramétrage IP fini.

Un tutorial pour la configuration figure en annexe.

Une amorce PXE nécessite pour le client de récupérer sur un serveur de fichier distant un fichier exécutable nommé pxelinux.0.

On utilisera donc un serveur TFTP.

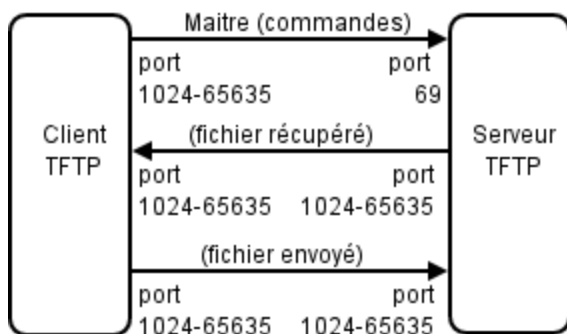
TFTP

TFTP est l'acronyme de Trivial File Transfer Protocol.
C'est un protocole simplifié de transfert de fichier.

Il fonctionne en UDP sur le port 69, au contraire du FTP qui utilise lui TCP.
Il n'y a donc pas de gestion de perte de paquet.

Il est rempli les même fonctions que FTP cependant il ne gère pas le listage de fichiers et ne dispose d'aucun mécanismes d'authentification ou de chiffrement.

Il faut donc connaître à l'avance le nom du fichier que l'on veut récupérer.



Nous l'avons donc installé le service tftpd sur le serveur maquette.

Les fichiers nécessaires pour l'amorce PXE seront stockés dans le dossier partagé /tftpboot situé à la racine.

Le premier d'entre eux est l'exécutable pxelinux.0.

Un tutorial pour la configuration figure en annexe.

PXELINUX

PxeLinux.0

L'exécutable qui sera téléchargé par les clients se nomme pxelinux.0, c'est un petit programme qui va lire un fichier de configuration, puis afficher un prompt pour que l'utilisateur choisisse son OS. Ce programme fait partie du projet SYSLINUX.

Il faut donc paramétrer pxeLinux.0 pour afficher les OS souhaités et l'action à effectuer par la suite. Afin de paramétrer ce fichier binaire il faut modifier le fichier pxelinux.cfg

PxeLinux.cfg

C'est le fichier de configuration de pxelinux.0.

Nous y avons écrit un texte de bienvenue pour le client ainsi que le nom des os.

Nous avons aussi paramétré l'action à faire une fois l'OS choisi.

Voici un exemple de PxeLinux.cfg. Ici le fichier n'affiche que l'OS ubuntu en option.

```
Display msgs/boot.msg => on affiche un message de bienvenu  
default local  
label Ubuntu    => choix de l'os  
kernel /xpe.0   => Action à effectuer « chargement du kernel ».  
timeout 100  
prompt 1
```

Un tutorial pour la configuration figure en annexe.

L'action qui suit permet donc ici de charger le kernel et de commencer à booter sur l'OS distant.

L'OS en lui-même n'est pas accessible par TFTP mais par un système de partage de fichiers plus complet.

En effet, après le démarrage du client, la réception et l'exécution de Grub, le téléchargement des images noyau et d'initrd, le système aura besoin d'un véritable système de fichiers.

Celui-ci sera monté à la racine et utilisé comme avec un disque local. Pour faire cela, nous allons utiliser NFS. Notez qu'il est possible d'utiliser d'autres systèmes de fichiers en réseau, mais nous resterons dans la configuration la plus classique ici.

NFS

C'est l'acronyme de Network File System.

C'est un protocole qui permet à un ordinateur d'accéder à des fichiers via un réseau.

Depuis sa troisième version, le principal protocole de transport utilisé est TCP. Contrairement à TFTP, NFS offre de nombreuses fonctionnalités telles que :

- Une gestion totale de la sécurité :
 - négociation du niveau de sécurité entre le client et le serveur
 - sécurisation simple, support de kerberos5, certificats SPKM et LIPKEY
 - Chiffrement des communications possible (kerberos 5p par exemple)
- Support accru de la montée en charge :
 - Réduction du trafic par groupement de requêtes (compound)
 - Délégation (le client gère le fichier en local)

- Systèmes de maintenances simplifiés :
 - Migration : le serveur NFS est migré de la machine A vers la machine B de manière transparente pour le client
 - Réplication : le serveur A est répliqué sur la machine B
- Reprise sur incidents
- Compatibilité : NFSv4 peut être utilisée sous Unix et maintenant également sous MS-Windows

Le service NFS s'installe très facilement et très rapidement via le paquet `nfs-kernel-server`. Bien sûr, vous pouvez également choisir `nfs-user-server` offrant plus de fonctionnalités mais étant plus "coûteux" en termes de ressources.

Côté configuration, le serveur NFS n'est pas très exigeant, une simple ligne dans `/etc/exports` précisant le dossier de partage et le sous réseau IP sur lequel il est accessible.

DRBL



DRBL (Diskless Remote Boot in Linux) permet de mettre à disposition un environnement système pour les clients qui peuvent booter en utilisant PXE/etherboot et l'ensemble des services décrits précédemment.

Le but est de déployer Linux (ou Windows) à de nombreux clients, évitant l'installation des systèmes d'exploitations "un à un".

Il permet également de changer la configuration logicielle de toutes les machines à partir de la machine serveur.

DRBL est un utilitaire puissant, utilisé dans de nombreuses écoles, des hopitaux

Taiwanais, ainsi qu'au CERI d'Avignon et est supporté par la communauté française de l'enseignement supérieure.

Le principe général de ce logiciel est de pouvoir utiliser des clients terminaux, sans système d'exploitation, et de les faire démarrer par le serveur à travers une connexion PXE.

DRBL fut créé afin de proposer une alternative à LTSP.

En effet LTSP permet la gestion et le fonctionnement de clients terminaux X légers. Ces terminaux transforment les signaux venant de la souris et du clavier, les envoient au serveur par le réseau, puis affichent sur leur écran le résultat renvoyé par le serveur.

Ces clients légers ne nécessitent ni disque dur, ni processeur puissant – on les appelle aussi « diskless clients », soit clients sans disque. Ils peuvent être des ordinateurs anciens, obsolètes ou peu puissants. Dépourvus de composants mécaniques mobiles, ils peuvent être plus économes et silencieux que des ordinateurs de bureau standards.

Avec DRBL, le serveur sert à envoyer les services au client à l'instar d'un serveur NFS et NIS. Tous les clients accèdent au serveur DRBL pour obtenir les authentifications, et les logiciels utilisent le processeur et la mémoire vive du client, à contrario de LTSP [1]. Si cela permet au serveur d'être relativement "léger" (un ordinateur actuel avec 2Go de RAM peut s'occuper d'une trentaine de PC sous Linux).

DRBL permet de cloner une image client et de la déployer à un ensemble de clients, à l'aide de Clonezilla server, comparable à Ghost server édition, True Image ou Rembo. Avec cette méthode, il est possible de déployer un système de 5,6Go sur 40 ordinateurs en 10 minutes.

Cependant, il est possible d'utiliser DRBL pour injecter un système ultraléger en RAM, comme Toutou Linux ou Damn Small Linux, FreeDOS permettant ensuite d'utiliser le client sans connexion PXE.

DRBL permet d'utiliser une machine d'une salle comme un client léger utilisant le système d'exploitation du serveur DRBL.

C'est cette fonctionnalité qui nous intéresse car elle répond en partie au cahier des charges de notre projet.

Les systèmes de fichiers supportés sont ext2, ext3, reiserfs, xfs, jfs, FAT, NTFS et HFS+. L'interface est très austère (textuelle), et les ordinateurs cibles doivent être dans un réseau privé. Si les guides en français ne foisonnent pas encore, quelques tutoriels sont disponibles sur le net.

DRBL va donc permettre aux clients de booter un OS installé et configuré sur notre serveur maquette.

Cependant DRBL ne répond pas à toutes nos attentes.

En effet, il permet de partager sur le réseau l'OS linux sur lequel il tourne mais il ne semble pas en mesure de partager plusieurs OS simultanément pour une amorce PXE.

De même il ne paraît pas possible de mettre à disposition un système d'exploitation Windows de cette manière.

La solution préconisée par DRBL est le SAN boot avec AOE ou iSCSI

SAN (AoE, iSCSI) Booting on DRBL

Le principe repose sur un réseau de stockage SAN (de l'anglais Storage Area Network), qui est un réseau spécialisé permettant de mutualiser des ressources de stockage.

Un SAN se différencie des autres systèmes de stockage tel que le NAS (Network Attached Storage) par un accès bas niveau aux disques. Pour simplifier, le trafic sur un SAN est très similaire aux principes utilisés pour l'utilisation des disques Internes (ATA, SCSI). C'est une mutualisation des ressources de stockage.

Les disques partagés n'apparaissent pas comme des volumes partagés sur le réseau. Elles sont directement accessibles en mode bloc par le système de fichiers des serveurs. En clair, chaque client voit l'espace disque d'un serveur SAN auquel il a accès comme son propre disque dur.

Cela fonctionne grâce à AoE (ATA over Ethernet) et iSCSI (internet SCSI), deux protocoles permettant le transport de commandes ATA et SCSI sur un réseau TCP/IP.

Les points forts du SAN :

- La Qualité de service (Qos) :
C'est l'argument majeur du SAN ! Le commutateur garantit un débit fixe de 8gb/s (précédemment 1gb/s, 2, 4) par lien en fibre optique, et assure le fait que la requête envoyée par un serveur a bien été reçue et prise en compte par les systèmes de stockage.
- La disponibilité :
Le SAN peut assurer la redondance du stockage, c'est-à-dire l'accessibilité au système de stockage en cas de panne de l'un de ses éléments, en doublant au minimum chacun des éléments du système. (haute disponibilité)
- L'hétérogénéité
Le SAN peut fonctionner dans un environnement complètement hétérogène, les serveurs Unix, Windows, Netware... peuvent tous rejoindre le SAN.

Cependant une telle solution requiert des moyens matériels nettement plus importants.