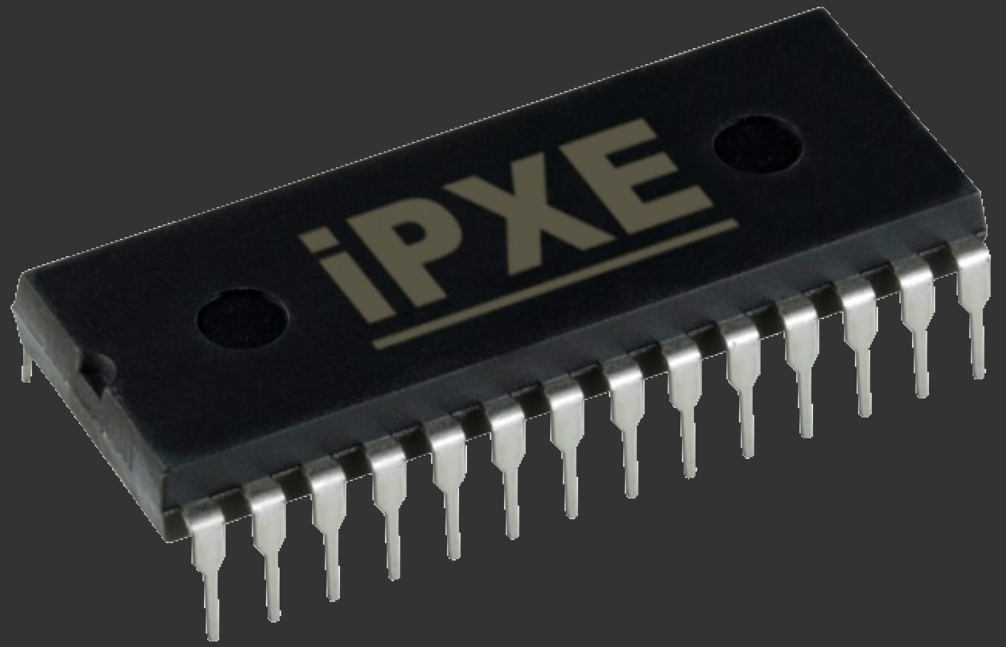
iPXE

Fonctionnalités, applications

iPXE

- Un firmware/bootloader PXE open source (GPLv2)
- https://ipxe.org



Parlons d’abord de PXE

- *Pre-boot eXecution Environment, aka pixie*
- 3 étapes :
  1. Le client recherche une adresse IP via DHCP
     - la réponse du serveur contient le nom du fichier de démarrage et éventuellement le nom ou l’adresse du serveur de démarrage
  2. Le client télécharge le fichier de démarrage via TFTP
  3. Le client exécute le fichier de démarrage
- La spécification PXE est incluse dans UEFI

Il y a pas que iPXE...

- PXELINUX (SYSLINUX)
- gPXE (Etherboot)

  iPXE est un fork de gPXE, qui n’est plus développé depuis 2010

Il fait quoi de plus iPXE ?

- Il permet de démarrer depuis/via :
  - HTTP
  - iSCSI
  - Fiber Channel over Ethernet (FCoE)
  - ATA over Ethernet (AoE)
- Il fonctionne sur réseaux filaires, sans-fil et même InfiniBand (quand on les moyens, pourquoi s'en priver ?)
- « Pimp my NIC » : on peut flasher la ROM d'une carte PXE avec son propre firmware iPXE tuné
- **<killerfeature>Il est scriptable</killerfeature>**

C’est quoi la recette ?

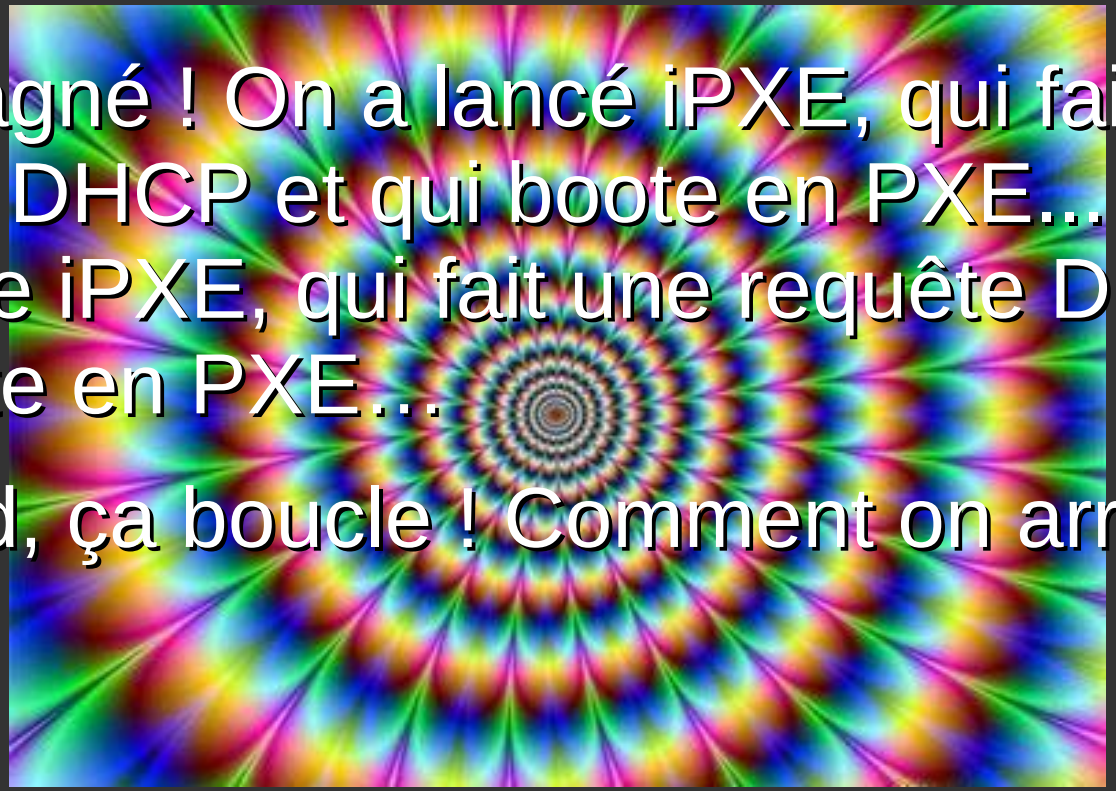
Ingrédients :

- Une machine équipée d’une carte réseau qui parle PXE
- Un serveur DHCP
- Un serveur TFTP

La recette facile : le chainloading...

- On télécharge iPXE depuis http://boot.ipxe.org/ :
  - undionly.kpxe pour les clients « old school » (BIOS)
  - ipxe.efi pour les clients « hype » (UEFI)
- On pose le tout à la racine du serveur TFTP
- On configure le serveur DHCP :
  - next-server : <ip ou hostname du serveur TFTP> (attention au DNS)
  - boot-filename : <le nom du fichier de démarrage>
- Et on fait démarrer le client sur la carte réseau...

OK, et après ?



- C’est gagné ! On a lancé iPXE, qui fait une requête DHCP et qui boote en PXE... et qui recharge iPXE, qui fait une requête DHCP et qui boote en PXE…
- Damned, ça boucle ! Comment on arrête ce truc ?

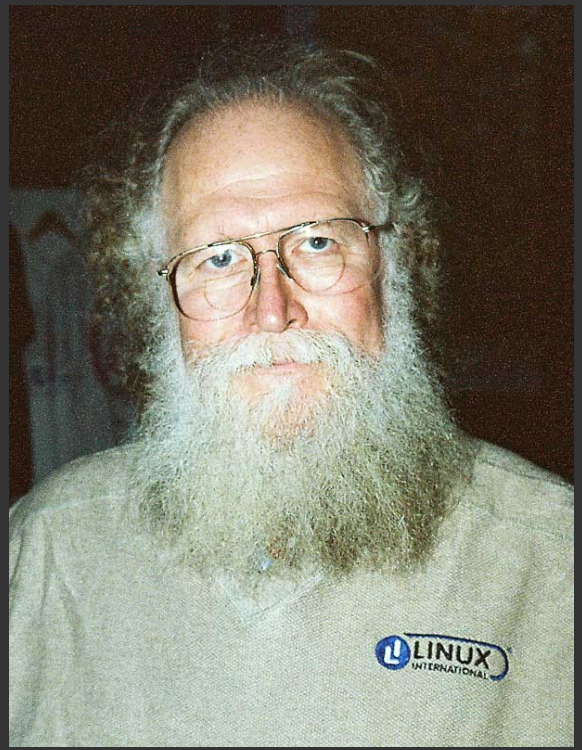
La ruse version « Bob le bricoleur »

- Avec ISC-DHCPd :

```
if exists user-class and option user-class = "iPXE" {
    filename "http://mon.serveur.web/mon_script.ipxe";
} else {
    filename "undionly.kpxe";
}
```

- Il y a la même pour MS DHCP mais il y trop de copies d'écran, clickodrome oblige, ça rentre pas sur un seul slide alors je vous mets le lien vous pourrez voir par vous même : https://ipxe.org/howto/msdhcp#pxe_chainloading

La ruse version « Même pas peur de build moi-même »




```
git clone git://git.ipxe.org/ipxe.git
cd ipxe/src
make bin/undionly.kpxe EMBED=mon_script.ipxe
```

- Pré-requis : un OS installé avec un shell, git, gcc, binutils, make, perl, liblzma or xz header files, mtools
  cf. https://ipxe.org/download#source_code
- Le script *mon_script.ipxe* est embarqué dans le firmware iPXE, et exécuté directement

Un script iPXE simple

```
#!ipxe

shell
```

- NB : on peut aussi appuyer sur *Ctrl*+*B* pendant le démarrage d'iPXE pour lancer un shell

Et c’est là que ça devient intéressant…

```
iPXE 1.0.0+ -- Open Source Network Boot Firmware -- http://ipxe.org
Features: HTTP iSCSI DNS TFTP AoE bzImage COMBOOT ELF MBOOT PXE PXEXT

iPXE> help

Available commands:

  echo            exit            isset           goto            help
  shell           autoboot        config          dhcp            pxebs
  ifopen          ifclose         ifstat          imgfetch        module
  initrd          kernel          chain           imgload         imgargs
  imgexec         boot            imgstat         imgfree         login
  show            set             clear           route           sanboot


Type "<command> --help" for further information

iPXE> ifstat
net0: 52:54:00:12:34:56 on PCI00:03.0 (closed)
  [Link:up, TX:0 TXE:0 RX:0 RXE:0]
iPXE> dhcp
DHCP (net0 52:54:00:12:34:56).. ok
iPXE> route
net0: 10.0.0.168/255.255.255.0 gw 10.0.0.1
iPXE> _
```

Commandes disponibles

- Boot
- Interfaces réseau
- Images de démarrage
- SAN
- Paramètres de configuration
- Contrôle de flux « basic » : isset, iseq, goto, exit
- Certificats
- Console
- Formulaires
- Utilitaires : ping, nslookup, echo, shell, reboot, poweroff, etc...
- Et d’autres : https://ipxe.org/cmd

Ok, mais pour faire quoi ?

- Installer un OS par le réseau :
  - sur le disque local
  - sur une cible iSCSI ou FCoE
- Démarrer depuis le réseau sur :
  - un environnement WinPE
  - une image disque
  - un LiveCD
- On peut également remplacer la ROM PXE par défaut de VMware par iPXE (Virtualbox utilise déjà son propre firmware iPXE)

Quelques applications d’iPXE

- FOG project
- netboot.xyz
- iPXE + Porteus Kiosk
- Bootman

FOG project

- *« A free open-source network computer cloning and management solution »*
- https://fogproject.org/
- Utilisait initialement PXELINUX, remplacé par iPXE depuis la version 1.0

netboot.xyz

- *« ...a way to PXE boot various operating system installers or utilities from one place... »*
- https://netboot.xyz/
- Une boite à outils pour installer des OS ou lancer des utilitaires directement depuis le web

HOWTO netboot.xyz

- Le script iPXE qui va bien :

```
#!ipxe
dhcp
chain --autofree https://boot.netboot.xyz
```

- Une démo vite fait (si tout va bien)
- Un petit tour sous le capot chez GitHub :
  - Installation de Debian GNU/Linux : https://github.com/antonym/netboot.xyz/blob/master/src/debian.ipxe
  - Le menu « couteau suisse » : https://github.com/antonym/netboot.xyz/blob/master/src/utils.ipxe

iPXE + Porteus Kiosk

- « Porteus Kiosk is a lightweight Linux operating system which has been restricted to allow only use of the web browser »
- http://porteus-kiosk.org/
- iPXE est utilisé pour démarrer les postes via HTTP
- Les postes sont configurés à l'identique via un fichier de configuration centralisé
- A votre disposition pour un complément d'infos : emmanuel.lestrelin@univ-amu.fr

IPXE + Porteus Kiosk : applications

- Postes de consultation du catalogue dans toutes les BU AMU
- Mise en mode « examen sur portail web » de salles informatiques (en production à AMU aussi)
- Postes d’affichage dynamique à partir d’un simple site web
- Et vous avez certainement d’autres idées...

Bootman

- « Un gestionnaire de boot (Boot Manager) en réseau »
- Pas encore de site…
  Au départ j'avais pensé le mettre chez GitHub, mais bon finalement peut-être pas en fait...
- Python + Django + BootStrap + iPXE
- A votre disposition pour un complément d'infos : emmanuel.lestrelin@univ-amu.fr

Principe de Bootman (1)

- *Computer* : en gros une adresse MAC
- *Action* : un script iPXE
- *Task* : une *Action* qui peut être exécutée par un *Computer* dans un intervalle de temps donné

Principe de Bootman (2)

- Le *Computer* s'adresse au serveur Bootman via HTTP lors de son démarrage
- Bootman cherche si une *Task* valide est programmée pour le Computer et lui renvoie l'*Action* correspondante, sinon l'*Action* par défaut spécifiée pour le *Computer*
- Les *Task* exécutées sont historisées puis supprimées
- Les *Task* obsolètes (non exécutées) sont conservées (mais supprimables)

Applications de Bootman

- *Chainloading* vers un serveur de déploiement d'OS PXE : MDT/WDS (en production chez nous), SCCM, FOG, Altiris, etc...
- *Chainloading* vers netboot.xyz et tout ce qu'il sait faire...
- Avec un peu d'imagination, ou de copier/coller depuis netboot.xyz, pas mal de trucs sympas...


Des réactions ?