

## Network booting putting the pieces together

- recall that tftp can be used to boot diskless machines
- there exists an open source project [gppe](http://etherboot.org) (<http://etherboot.org>) which provides a GPL implementation of a tftp client, which can be installed on
  - a USB device
  - a hard disk
  - floppy disk
- gppe can also boot using http and obtain a boot menu from the server!

## GNU PXE

- GNU PXE (aka gppe) implements a PXE (Preboot Execution Environment)
- however it does more than implement a PXE
- extended to provide a flexible network bootloader
  - supports protocols: tftp, nfs, http, ftp
  - can boot Linux, FreeBSD, multiboot, EFI (OSX), Windows CE images
- it also works well with syslinux (and it can download graphical boot menu from the server)
  - allowing the user to choose further boot images

## Booting via http

- much more efficient than booting from tftp
  - visually compare the boot time of a tftp thin client against a http thin client
  - tftp is using a stop and wait (idle RQ) protocol
  - http is using tcp (a sliding window protocol)

## Booting via http

- provides flexibility, probably slightly easier to configure a web server (apache) than a tftp server
- one less port to manage (if you need to provide http anyway)
- much easier to debug!
  - where is the boot image?
  - is it visible from the client?
  - these questions are trivial to solve using http!
  - a little more challenging is tftp is used

## Building gpxe on Debian

- download various useful tools

```
$ sudo apt-get install syslinux mtools git
```

- now download the source code

```
$ git clone git://git.etherboot.org/scm/gpxe.git
```

- build the code

```
$ cd gpxe/src
$ make
```

## Add a configuration file for gpxe

- emacs mybuild-http

```
#!/bin/bash
make clean
make DEBUG=gpxe_call,gpxe_file,gpxe_tftp,gpxe_preboot,\
gpxe_undi,gpxe_udp,gpxe_loader \
EMBEDDED_IMAGE=../contrib/scripts/glam-http.gpxe
```

## Add a configuration file for gpxe

- now create the glam-http.gpxe script

```
#!/gpxe
dhcp net0
chain http://193.63.129.1/tftp/i386/glam-j203.gpxe
```

- once this works, you could try out using

- http instead of tftp

- notice that this script is embedded inside the data section of the gpxe binary

- which will end up as a boot image on your USB or CDROM, or hard disk

## Add a configuration file for gpxe

- you should be able to build gpxe by typing:

```
$ bash mybuild-http
```

- note that the ../contrib/scripts/glam-http.gpxe script references a file glam-j203.gpxe which is held on the server 193.63.129.1

## Add a configuration file for gpxe

- the contents of this file are:

```
#!/gpxe
dhcp net0
kernel http://193.63.129.1/ltsp/i386/vmlinuz ro \
root=/dev/nfs ip=dhcp boot=nfs nfsroot=193.63.129.1:/opt/
initrd http://193.63.129.1/ltsp/i386/initrd.img
set root-path iscsi:193.63.129.1:nfs:::/opt/ltsp/i386
boot vmlinuz
```

- note how easy it is to modify all clients!
  - change this server file so that the client boot using http or tftp
  - or change all clients to access dhcp on the second Ethernet interface
  - or use a different NFS server (it would be a minor code change to implement load balancing)

## Installing gpxe on the local hard disk

- maybe you want to provide multi boot clients (Window/Linux)
- syslinux can be used on a spare partition
  - can normally be contained within about 10 MB size
- install syslinux ([follow instructions](http://etherboot.org/wiki/syslinux) (<http://etherboot.org/wiki/syslinux>)) and adapt appropriately for the hard drive rather than USB

## Installing gpxe on the local hard disk

- add a file `syslinux.cfg` in the new partition
- on our clients in J203 we have three partitions
  - partition 1 is for system use (Norton ghost)
  - partition 2 is for Windows (whatever flavour)
  - partition 3 is 30 MB which consists of syslinux (7% full)
- note a copy of `gpxe.krn` must also be placed into this partition

## syslinux.cfg

```
default vesamenu.c32

prompt 0
noescape 1
allowoptions 0
totaltimeout 150

MENU TITLE Please choose a system to boot

LABEL windows
MENU DEFAULT
MENU LABEL windows
kernel chain.c32
append hd0 2

LABEL gpxe
MENU LABEL linux
kernel /gpxe.krn
```

## Conclusion

- gpxe is very powerful and customisable
  - much of the configuration is held on the server
  
- syslinux works well and is resource lean
  
- it might be possible to make utilise boot menus across the network
  - although we have purposely not done this, why?
  
- in J203 the file `glam-j203.gpxe` is downloaded via tftp
  - would this present a performance problem?

## Interesting video

- [google talk](http://www.youtube.com/watch?v=GofOqhO6VVM) (<http://www.youtube.com/watch?v=GofOqhO6VVM>)