

Version 1.0

**Sebastian Schmelzer (sebastian.schmelzer@rz.uni-freiburg.de),
Dirk von Suchodoletz (dvs@openslx.com)**

**The project is made available under the GNU General Public License,
this paper follows the GNU Free Documentation License.**

OpenSLX is providing you a toolkit to simplify the setup of a network booted environment via PXE. It includes the preparation of the master images, the configuration of the boot servers as well as the live configuration on boot time of the client. With the introduction of the PreBoot environments on CD, USB and other media we are now able to leave our predefined local setup and run the network boot from anywhere.

Network Linux Anywhere

Introduction

OpenSLX aims to provide easy administration of large bunches of computers on the Linux desktop and larger cluster setups as a middleware solution. The project might be of interest in a wide field of application: schools, education, universities, grid number crunching clusters, corporations with a lot of office workplaces.

OpenSLX does not boot only within PXE environments but offers the option of a flexible anywhere boot: e.g. it makes possible to boot over wireless LAN with a small (invariant) boot system installed to the machine. Using rather the same framework it would be possible to boot clients over WANs (fast DSL and cable) connections too: the rising amount of available bandwidth for normal end users offers to boot the complete system over the Internet. Only a small addition to the standard boot loaders is required. These features are achieved by booting from CD/DVD as e.g. option parallel to the standard installation environment of a Linux distro, USB stick or by Linux/Windows standard boot loaders. The booted clients do not just run general login/kiosk mode, but are configurable, include local filesystem, USB stick mounting. This is implemented using a small initial Linux environment and kexec.

The operation and administration of a larger number of computers is quite time-consuming. This is also true for Linux installations. So that administration won't become the main task with your implementation and maintenance of a larger installation, there is a need for special technologies to get the problem under control. Apart from the classic method of software deployments and patch managements, Stateless approaches have the best chance to relieve the admin from boring routine tasks. On request, one installation can serve any number of client-machines over the network. On the client side there is no need for any special preparation of the machines: PXE, Etherboot or kexec-preboot are sufficient to allow the machine to boot from the network. Out of

this approach additional freedom of design is achieved: Local installations don't need to loose ground, but for example they can offer an alternative to the stateless operation during migration or mixed scenarios.

Usage Scenarios

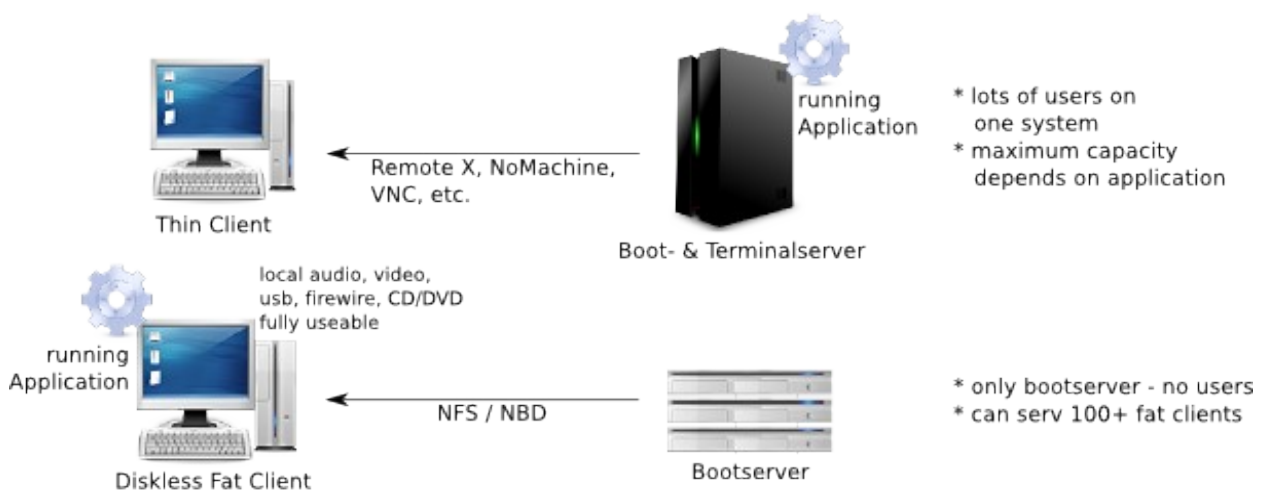
OpenSLX deals with GPL-Software that allows installation and easy administration on hard drive free Linux workstations. The project focuses on large installations with a similar deployment profile (not necessarily on machines with special hardware for production control etc.). This is currently implemented in schools or universities to have a highly flexible and stable pool environment for training courses, classes etc. On the established stateless-Linux-basis, another OS for training purposes, such as Windows or Linux, can effectively be distributed with the help of virtualization technologies. OpenSLX clients might be deployed in a wide range of setups beside the standard diskless desktop system:

- General Linux application service providing: Flexible platform to offer services to the average end user on a standard Linux platform without the need to install software locally. To reduce the licence costs of your client setups, this could also be used to implement something like a licence pool, where a certain number of instances of the same software are served to every machine in the network until the number of available licences is exhausted.
- Set-top box operating system: No need to squeeze a complete system into a small firmware flash device (better use this for users data). Enough bandwidth to boot over the net, fast updating without user intervention, several different image types could be offered in parallel (to be selected in boot time)
- Secure Internet box: ISP could offer a secure general purpose Internet surf box booted on the standard desktop Windows PC (as an alternative to the locally installed system). The local system is not altered at all, but the set of online applications are kept in updated state by professional service providers (as a small fee service or an add on like telephony flatrate). This could be secured with the help of TPM.
- Automated special purpose systems: jobs like seeking the local installed system for viruses, making backups or other maintainment tasks could be run by a small Linux system automatically booted outside business hours via wake on lan on every client computer in the network.
- Live-demos of distributions and their development states: No need to install a system for evaluation (no need of CD/image to distribute, fast update, only the newest official version is available, comparable to the well known Knoppix System)

An OpenSLX client is just a Linux system as you would expect, if you had installed just any distribution onto the local disk. The average user will not see any difference. The typical applications are cluster computing (booting just exactly the same system on a large number of machines with a wide range of different hardware configurations) and PC pools in education (on a range of identical machines; robust base system without any local installed OS; flexible booting of various Linux distributions with different configurations providing a basic office software setup).

Why not use terminal server products?

For the administrator things change significantly: All software installation is done on a central server (or servers for failover) and provided via network in a way similar to the well known LTSP or X2GO. So the project uses the same infrastructure of PXE/etherboot/gPXE, DHCP and TFTP – or general DHCP environment with PreBoot – for initial booting and then NFS, NBD, DST ... for the root filesystem of the clients. Different to LTSP or alike the users logs in to the machine locally and not to a remote server (this can still be done, but it's not the main focus). So the local resources of the machine are directly useable and the user can easily access to all devices of his machine (USB, IEEE1394, CD/DVD, floppy,...). Furthermore he is able to play any 3D game, watch videos and may connect audio devices like headsets, microphones.



OpenSLX vs. LTSP

The switch-over between different operation modes occurs at boot time, since there is no expensive Image distribution (imagecopy) before or during the machine boot. Plenty of good Linux distributions are available: For this reason OpenSLX doesn't choose the approach to offer its own package collection. Instead a script collection is available which is responsible for preparing many standard distributions for the stateless operation. The OpenSLX package prepares a middleware for you that stays hidden from the user of the machine. So the appropriate "Look & Feel" of a distribution is maintained, the customization and changes occur in the background.

How everything works together

In the following we will describe the requirements of the system, what steps are done to setup a OpenSLX server environment and how the preparation of a customized Linux distribution for network booting is solved.

The OpenSLX Server

Depending on the range of the installed system, a system needs 2 - 8 Gbytes for its reference copy on the server. Determined by the type of export you choose to distribute the system it requires additionally between 90 and 98% of this amount for a NFS share; for a NBD/SquashFS-Export between 35 - 50%.

SLX-Clients operate stateless and have no local operating system installed. They obtain their joint root filesystem from a central server, which should be configured with sufficient capacity. The largest load is carried by the root filesystem export service, which generates the largest filesystem output. For a setup consisting of 100 clients, you should have one to two gigabit interfaces and fast hard drives bound by RAID installed on your server. The requirement of DHCP and TFTP do not even burden an average server with more than 1000 clients. Depending on the type of operation, in one day between 0.3 and 3 Gigabytes of data are transported per client. With the installation of Network block devices with SquashFS the amount of data can be reduced up to 50% in comparison to NFS.

An increase of the client connections to 1 Gigabit should be accompanied with improved IO of the server: A very fast RAID or placing the exports in RAM connected with gigabit-trunking accelerates the start and operation of the clients considerably. Then they are noticeably faster as in classic operation from a local hard disk.

For demonstration purposes within smaller networks and only few clients an average equipped fileserver or a normal desktop PC should suffice.

On the software side every common Linux distribution is applicable as a basis for our OpenSLX server. The only limitation is that you will need to use a 64bit distribution on the server if you want to serve 64bit client systems.

From the master image to the first boot

To run the OpenSLX suite it is not necessary to dissolve complicated dependencies. You will just need an off-the-shelf Perl with standard modules which are offered within every typical Linux distribution, alternatively via CPAN. The database backend requires an installed SQLite or MySQL. For the installation of the toolkit packages are provided in DPKG, RPM or TGZ format.

The additional requirements are determined by the type of export for the client root filesystem over the network. You might choose between the classical NFS method and different kinds of Network Block Devices using a SquashFS on top.

To boot your clients using PXE or Etherboot your server or any other server in your network should provide DHCP and TFTP services which should be accessible for the OpenSLX scripts. If you deploy PreBoot systems only, you would need a standard DHCP and additionally a webserver to provide Kernel, InitRamFS and configuration packages.

There are two possible ways for the user to prepare a linux distribution for the boot process / usage with OpenSLX. The preferable method is the setup of the distribution on a reference machine - either on real hardware or within a virtual machine/appliance. Once the reference system is fulfilling the needs of the customer/user it's going to be synced to the openSLX server with a call of the "slxos-setup" utility. This method is called "clone". The transfer of the data to the server system is handled by the "rsync" utility - this has the big advantage that it dramatically reduces the effort (amount of data transferred) of a re-clone after an update or other modifications on the reference system.

Alternatively to the cloning from a reference system OpenSLX offers the option to install a Linux system directly from the repositories of the vendor. This is done by using a chroot environment directly on the server (bootstrapping). In contrast to the cloning you would get only a base system with text console access. Components for graphical desktops are not installed by default because of complex dependencies. But you could extend the base setup by hand within a change-rooted shell on the server. At the moment stage of development the install option is not offered for all distributions.

The image created of the installed Linux located in a subdirectory on the server is named stage1. For the distribution of the root filesystem for the network boot of the clients some modifications are required. For this reason - even for the price of nearly doubled amount of disk space - the stage1 system is copied again via rsync. At this point all components which are not needed for netboot are skipped. This selection contains the data of the package manager, log files, caches and autostarter files for graphical desktops which don't make any sense in stateless operation like the NetworkManager applet.

The stages in detail

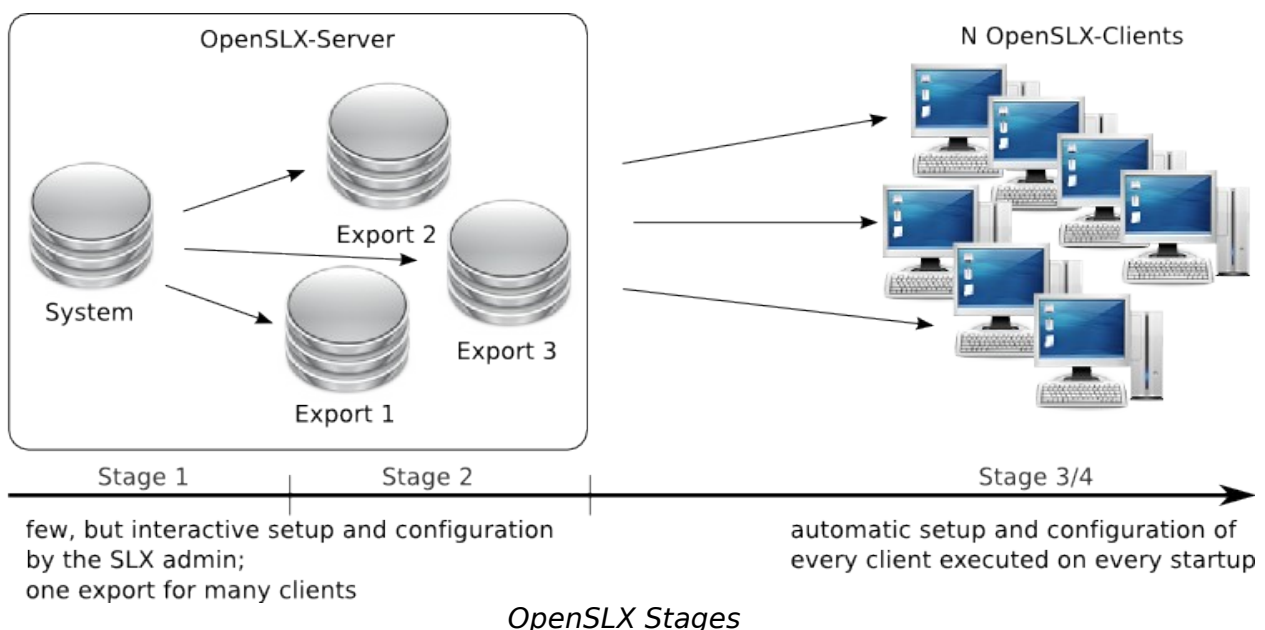
The task of the OpenSLX project is the setup of exportable systems - distributions in a certain version - that can be incorporated as root filesystems for stateless clients over the network. A SLX server can supply many of these systems.

For a clear view the setup procedure of your stateless is separated into couple of defined stages:

- **Stage0** identifies a running reference system on real or virtual hardware, of which a copy through Rsync is generated on the server. This stage simply plays a role for the clone mode.
- **Stage1** identifies an executable client system, which is placed beneath `/var/opt/openslx/stage1/<name>`. With consistent hardware architecture you can access this directory through "slxos-setup shell <name>". The system is strictly setup and extended with particular local extensions. To this you could count the installation of special external software packages, not offered from the distribution. There are two ways in which

you could setup stage1, through direct installation from the respective packet source or copy of an existing system (clone, stage0).

- **Stage2** - from stage1, different client root filesystems can be created. For this stage1 is duplicated, this means the omission of irrelevant files for the stateless operation, a new subdirectory in NFS-Export is created or for network block devices a compressed SquashFS is applied. Therefore you need an appropriate Kernel (usually taken directly from the distribution being used), at least one InitialRamFS fitting to the Kernel and the client configuration. The client configuration can contain an array of files that are extracted to the appropriate place in stage3.
- **Stage3** – The administrative procedures preparing the stages 0 to 2 have to be carried out on the SLX server only a few times. Stage3 describes the environment provided on every single client at every restart within the InitialRamFS. In this stage the individual machines are set up. This area is covered by the shell scripts `init` (SLX-Init), `hwautocfg` and `servconfig`. The latter places all the clients' important configuration files in `/mnt/etc`. Local extensions (for all clients that use a certain InitRamFS) are possible by using `preinit.local` at the very beginning of SLX-Init, `postinit.local` towards the end of SLX-Init. `Postinit.local` is created through `ConfTGZ`. This is a package of configuration files that are downloaded using TFTP e.g. and then extracted in stage3. The template for this can be found in `/var/opt/openslx/config/<system>`. `<system>` is the name of the vendor OS linked by two colons and the type of the export.
- In **stage4**, SLX view, only some smaller configuration tasks take place. The client now has a complete root filesystem (all programs and files); thereupon the keyboard configuration can be done. Finally the graphical login is started at the earliest possible time.
- **PreBoot** – is a special mini-Linux prepended to stage3 to allow non-PXE setups, e.g. in WLANs or over the Internet.



Depending on the type of the clients, you may want to prepare one or more different system exports on a server. This can mean that a stage1 system in many varieties, like NFS and at the same time NBD/SquashFS, should be delivered to the clients. On the other hand, very different stages 1 can be served with the same technology from different exports. This is, for example, useful for on-site tests, when you would like to offer the new system parallel with an existing system.

If you would like to export one and the same base system in different ranges, there are two options: Either from the start you deploy two different stage1 versions which are maintained separately or you move the differences into a certain area which will later be additionally mounted in the environment, e.g. by using a stacking file system.

Configuration of the Client Systems

The configuration of a client system is done by two tools. On one hand you have got the “slxconfig” tool to setup the truly basic parameters like the kernel parameters, on the other hand you have “slxos-plugin” to install OpenSLX plugins, a flexible modular infrastructure for all other complex configuration tasks as the setup of the X window system or virtualization.

Basic configuration with slxconfig

The basic configuration tasks which hardly differ on every system are done with “slxconfig”. This covers the setup of the kernel – modification of the kernel command line as well as the selection of modules included in the initial ramfs, the console setup and various toggle switches for default daemons, like cron hal, ntp, ssh, etc.

A list of all current available options and default settings can be retrieved with the execution of the following command:

```
slxconfig --verbose list-system default
```

The change of a parameter follows the pattern:

```
slxconfig change-system <vendor-os> attr2=... attr2=...  
slxconfig-demuxer
```

The execution of the demuxer is required to rebuild the initial ramfs. It is also possible to change most of the parameters for a specific client – identified by its mac address – only. Beside these parameters for the vendor-os there are also some parameters for each export, like the source address of the root filesystem. The usage of slxconfig is analogue to the already shown examples – only switch list-system with list-export and change-system with change-export.

Plugins – Extensions to the Base Client

The general software architecture of OpenSLX aims to extend the base functionality of the system by implementing special extensions (OS-Plugins). The previous steps were sufficient for a bootable base system, however not a comfortable graphical desktop. This task is taken over by the plug-ins "xserver" and "desktop". The first takes care of the automatic setup of Xorg and the integration of proprietary OpenGL drivers from NVidia or ATI/AMD. The second deals with the setup of display managers and the graphical default session.

OpenSLX-Plugins should keep the base infrastructure compact and clear. Therefore all primary extensions to the basic system are applied as plugins. Every OS plugin presents a particular extension for all of OpenSLX supported vendor OS. These modules can each be developed and tested independent of the kernel system and of other extensions.

By now there exists a fairly extensive list of plugins, which realize a number of different tasks:

- The "xserver" plugin serves for the setup and configuration of the X-Server (Xorg) for all graphical interfaces of desktop systems. As soon as the graphic hardware on a computer is used (no X-Terminal server, that only offers X remotely), this plugin is urgently needed. The plugin allows for many distributions that install the proprietary 3D drivers from ATI and Nvidia.
- The "desktop" plugin is needed for the graphical user interfaces of Xorg and can, for example, setup the display manager. It is dependent on an installed xserver plugin.
- The "vmchooser" plugin extends the list for selection of desktop sessions for virtualized environments. Alternative to a Linux Gnome or KDE session, a virtualization environment like VMware (QEMU, Virtual Box, ...) can be started, which provides a complete own desktop.
- The "vmware", "virtualbox" and "qemukvm" plugins provide the bases for virtualized environments.
- The "bootsplash" plugin offers a nice graphical start screen with a progress bar, so that the user doesn't need to look at the console output of the system booting.
- In addition to these plugins there exist many more, like the "syslog" plugin, which are concerned with the configuration of the machine logging about the absolute basics.

All plugins have their own attributes, which are entered into the database for each existing vendor OS entry at the first installation. At this point the attributes were set to their default values which later can be changed using the the command "slxconfig".

Stage1-attributes of a plugin are supposed to give the operator the possibility to control the installation of the plugin into the vendor OS, i.e. the values of these attributes determine which variant of the service provided by the plugin should be available in the respective vendor OS (and therefore in the systems and clients). The task of stage3-attributes is to perform the configuration of the plugin for particular systems/clients, i.e. the values of these attributes determine which of the possible variants of the plugins services should be used when the system boots for a particular system/client. Thereby the possible variants are determined through stage1-attributes; the concrete selection of a particular variant from these possibilities is done by stage3-attributes.

The practical integration of plugins follows the pattern:

```
slxos-plugin install <vendor-os> <plugin> stage1attr2=... stage1attr2=...
slxos-export export <vendor-os> <exportType>
slxconfig-demuxer
slxconfig change-system <vendor-os> xserver::prefnongpl=1
```

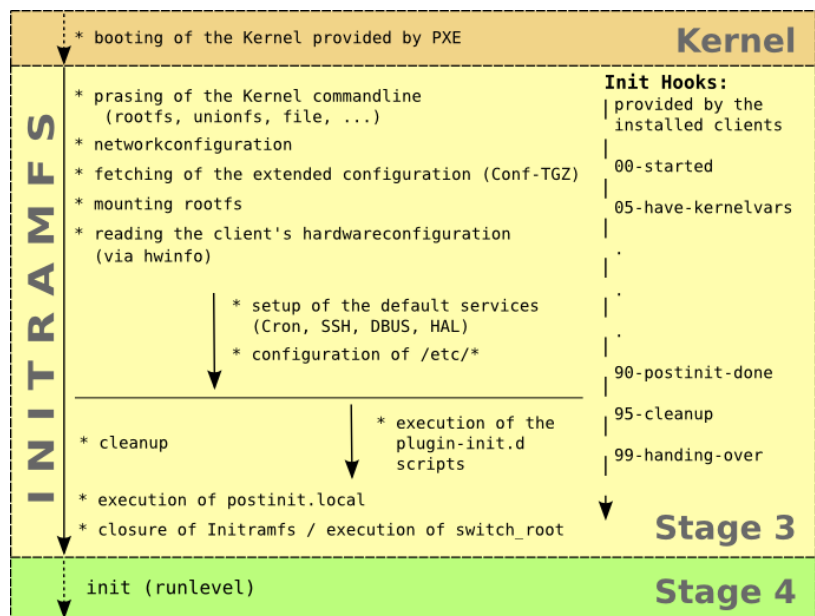
This means: Plugins are first added to the vendor OS. In order that later they can be started and configured in stage3 by the client, the export should be run followed by a run of the demuxer. If you would like to change some of the attributes, slxconfig will take care of this, however then demuxer must also be executed again.

The default boot process

The boot process starts with a “grub” like textual menu to choose the desired image to boot – this is provided by PXELinux (a subproject of H. P. Arvin's Syslinux).

After the users decision the image's corresponding Kernel and initramfs is fetched via TFPD and the Kernel is executed.

Afterwards the distribution specific init script out of the OpenSLX tools is executed inside a minimal uClibc based environment (stage3) – all together part of the initialramfs



generated by the demuxing process of OpenSLX. The init process is responsible for the live configuration of the client and the execution of additional scripts provided by plugins (plugin-init.d scripts as well as their init hooks) and scripts

coming out of the OpenSLX configuration package (preinit.local and postinit.local).

After all stage3 tasks have been completed the initramfs is left and the boot continues like on a local installed distribution: the system's own init process is started.

PreBoot – Replacing PXE for Special Scenarios

As we have seen in the introduction there are quite a lot of dependencies on the given infrastructure of the environment. To reduce these, a couple of preboot media were created that take over the place of a PXE based setup.

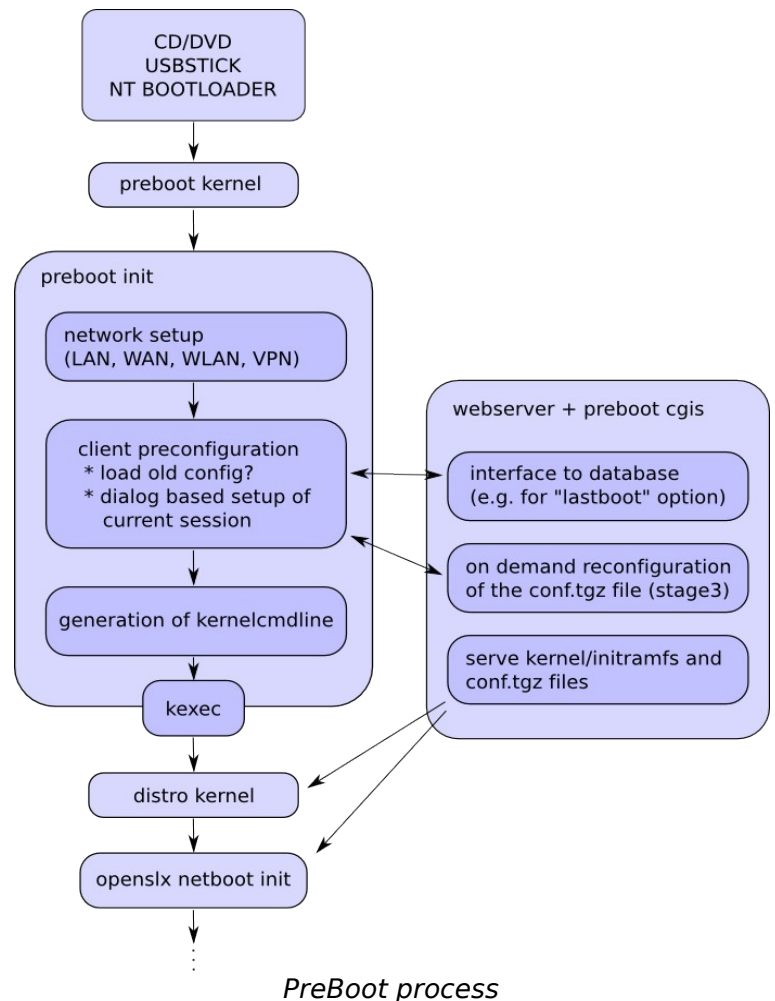
On these media a generic kernel and a small uClibc based Linux environment is placed to fulfil the network setup as well as the selection, configuration and start of the desired system to boot.

Due to the fact that we took http as replacement for tftp as desired method to fetch the required files for booting (Kernel, initramfs and OpenSLX configs), the way to extend this with the possibility to interactively manipulate the OpenSLX configuration of the system on boottime was pretty close.

Beside the selection of the desired image to boot – comparable with the boot menu we know from our pxe setup - the PreBoot media allows the user to choose between kiosk and normal boot, to set up users and passwords, configure plugins and much more.

The data is stored in files on the SLX server. The association with the client is done by naming the files after the client's unique MAC address, which later makes the allocation of the client to its configuration quite easy.

After the user has selected and set up the image to boot the corresponding Kernel, initramfs and config is fetched. Finally the normal boot process is triggered with a call of the “kexec” tool.



Future

The current development is following two main goals. On one hand we are working on using the OpenSLX system for grid computing in cooperation with the Black Forest Grid Group: one goal is to provide easily configurable Linux instances optimized for every job on the old grid setup, the other goal is to extend the grid on our existing pool pcs during night hours or while they are idle. On the other hand the focus is to build specialized installation for WAN boot on top of the work done with the PreBoot environment. For example a discless Android installation can be thought of.

Further reading & contact

For further information, questions or just to follow the development:

Website: <http://lab.openslx.org>

Mailinglists: <http://lists.openslx.org>

Contact: info@openslx.org

The nightly builds of OpenSLX can be found on <http://packages.openslx.org>.