Universidad Autónoma de Yucatán

Facultad de Matemáticas

Gestión de dispositivos de TI

Profesora: María del Carmen Zozaya Ayuso

Proyecto integrador: Implementación de Servidor Linux con Clientes sin Disco Duro

Equipo 1 - Integrantes

Adrián Jesús Baas Noh

Víctor Hugo Martínez Salazar

Maricarmen Buenfil Pérez

Licenciatura en Ingeniería en Computación

Licenciatura en Ciencias de la Computación

1 CONTENIDO

2 Introducción 3
3 Objetivo del proyecto 3
4 Descripción de la actividad 3
5 Fase 1 – Investigación Teórica 4
5.1 ¿Qué es un servidor Linux? 4
5.1.1 Funciones típicas de un servidor Linux: 4
5.2 ¿Qué es un cliente sin disco duro? 5
5.2.1 ¿Por qué existen los clientes diskless? 5
5.2.2 Tecnologías fundamentales para un cliente sin disco: 5
5.3 ¿Qué es PXE? 6
5.4 ¿Qué es DHCP? 7
5.5 ¿Qué es TFTP? 7
5.6 ¿Qué es NFS? 8
5.7 ¿Qué es una instalación basada en red (Network-based installation)? 8
5.8 ¿Por qué DEBIAN es adecuado? 9
5.9 Comparación con LTSP, FOG Project, Clonezilla 10
6 Fase 2 – Documentación Técnica 10
6.1 Instalación de DEBIAN 13 10
6.1.1 Selección e instalación del sistema operativo 10
6.1.2 Instalacion de herramientas necesarias para el proyecto 12
6.2 Configuración de la red 13
6.2.1 Verificación de la interfaz de red 13
6.2.2 Problemas iniciales de DNS y “fallo temporal en la resolución del nombre” 14
6.3 Configuración de servicios 16
6.3.1 DHCP (dhcpd.conf) 16
6.3.2 TFTP (tftp-server) 16
6.3.3 PXE (pxelinux.0) 18
6.3.4 NFS (/etc/exports) y creación de roofts 20
6.4 Arquitectura de red 22
6.4.1 Flujo de arranque resumido 22
7 Fase 3 – Pruebas Virtuales 23
7.1 Consideraciones sobre pruebas virtuales 23
7.2 Pruebas conceptuales realizadas sin virtualización 23
8 Fase 4 – Implementación Física 24

8.1 Preparación del cliente físico 24
8.2 Arranque inicial del cliente 25
8.3 Carga de pxelinux.0 y menú PXE 26
8.4 Carga del kernel e initrd 26
8.5 Montaje del rootfs mediante nfs 26
8.6 Inicio del sistema Debian en el cliente 27
9 Conclusiones 28

2 INTRODUCCIÓN

En esta actividad, los alumnos realizarán una investigación exhaustiva sobre la implementación de un servidor Linux que permita a computadoras cliente sin disco duro arrancar, ejecutar aplicaciones y conectarse a Internet simultáneamente a través de una red LAN.

3 OBJETIVO DEL PROYECTO

Investigar, planificar e implementar una solución en red donde una computadora funcione como servidor con un sistema operativo basado en alguna distribución de Linux (o proyecto equivalente), y permita la conexión de dos computadoras cliente sin disco duro a través de una red LAN.

4 DESCRIPCIÓN DE LA ACTIVIDAD

Los estudiantes deberán:

• Investigar cómo instalar y configurar un sistema operativo Linux en una computadora para que funcione como servidor.

• Configurar el servidor para que permita el arranque remoto (boot por red) de dos computadoras cliente sin disco duro, utilizando tecnologías como PXE (Preboot Execution Environment).

• Asegurar que ambos clientes puedan ejecutar simultáneamente programas desde el servidor, como juegos, procesadores de texto u otras aplicaciones funcionales.

• Verificar que las computadoras cliente tengan acceso a internet a través del servidor.

• Documentar el proceso, incluyendo los pasos de instalación, configuración de red, servicios utilizados (DHCP, TFTP, NFS, etc.), pruebas realizadas y resultados obtenidos.

5 FASE 1 – INVESTIGACIÓN TEÓRICA

En esta fase se investigaron los conceptos fundamentales relacionados con la implementación de un servidor Linux capaz de proporcionar arranque por red a clientes sin disco duro.

5.1 ¿QUÉ ES UN SERVIDOR LINUX?

Un servidor Linux es un sistema informático que ejecuta una distribución del sistema operativo Linux, diseñado para ofrecer servicios de red a múltiples clientes de manera robusta, segura y eficiente.

5.1.1 Funciones típicas de un servidor Linux:

SERVICIO	PROPÓSITO
DHCP	Asignación de direcciones IP en la red
DNS	Resolución de nombres de dominio
TFTP	Envío de archivos de arranque PXE
HTTP/FTP/NFS	Distribución de archivos o sistemas completos
SSH	Administración remota segura
PXE	Arranque por red para clientes

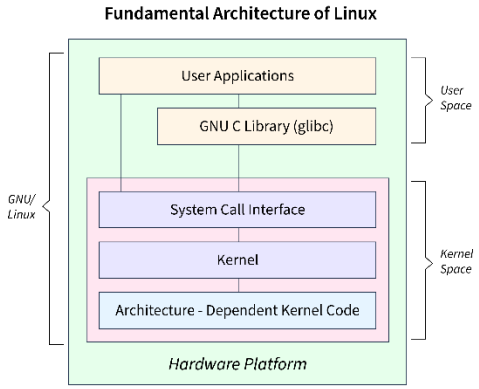
En el contexto de este proyecto, el servidor Linux cumple cuatro roles simultáneos:

Servidor ProxyDHCP (solo envía parámetros PXE, no direcciones IP).

Servidor TFTP (envía pxelinux.0, kernel e initrd).

Servidor NFS (proporciona el sistema operativo completo para montar desde red).

Servidor PXE (coordina todo el flujo de arranque de los clientes).



5.2 ¿QUÉ ES UN CLIENTE SIN DISCO DURO?

Un cliente sin disco duro (diskless client) es una estación de trabajo que no cuenta con almacenamiento local y depende totalmente de un servidor remoto para:

- Cargar e inicializar su kernel
- Obtener su sistema de archivos
- Ejecutar procesos
- Administrar usuarios y configuraciones

5.2.1 ¿Por qué existen los clientes diskless?

Reducción de costos (no hay necesidad de SSD/HDD).

Seguridad: ningún dato queda almacenado localmente.

Control centralizado: una sola actualización afecta a todos los clientes.

Mantenimiento mínimo: el hardware del cliente es casi stateless.

5.2.2 Tecnologías fundamentales para un cliente sin disco:

PXE → arranca desde la tarjeta de red.

TFTP → descarga bootloader y kernel.

NFS-root → monta el sistema de archivos desde el servidor.

DHCP/ProxyDHCP → obtiene parámetros de red.

Con estas tecnologías, el cliente ejecuta un sistema Linux totalmente remoto.

5.3 ¿QUÉ ES PXE?

PXE (Preboot Execution Environment) es un estándar de Intel que permite a una computadora arrancar sin disco desde un servidor remoto utilizando exclusivamente su tarjeta de red.

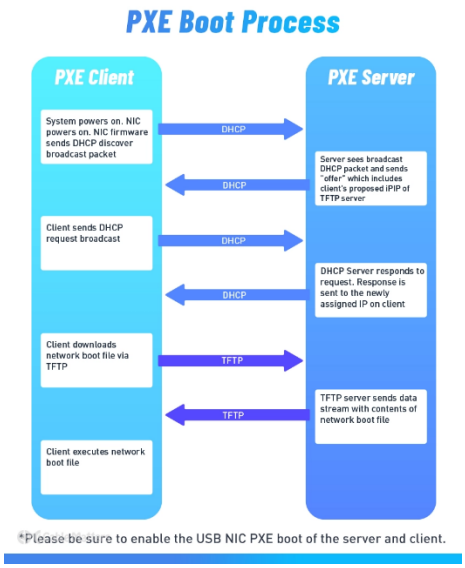
El proceso PXE ocurre antes de que el sistema operativo cargue.

El flujo PXE completo:

1. NIC envía DHCPDISCOVER con bandera PXE
    a. La tarjeta de red solicita:
    b. dirección IP
    c. ruta del archivo de arranque (bootfilename)
    d. dirección del servidor TFTP
2. DHCP institucional responde con una IP
    a. Esto ocurre automáticamente en redes administradas.
3. ProxyDHCP responde con las opciones PXE faltantes
    a. pxelinux.0
    b. dirección del TFTP
    c. configuración PXE
4. Cliente descarga pxelinux.0 desde TFTP
   Este archivo es el bootloader (Syslinux).
5. pxelinux.0 descarga el menú PXE y los archivos necesarios
    a. kernel → vmlinuz
    b. initrd → initrd.img
    c. Configuración de arranque (pxelinux.cfg/default)
6. Kernel se inicializa y ejecuta el initramfs
   El initrd contiene scripts para montar el root filesystem.
7. Montaje del sistema operativo por NFS-root
   El cliente ejecuta Linux desde /srv/nfs/rootfs.

Ventajas de PXE

- Arranque estandarizado reconocido por todas las BIOS/UEFI.
- No requiere modificaciones en hardware.
- Permite despliegues masivos y administración centralizada.



5.4 ¿Qué es DHCP?

DHCP es el protocolo encargado de asignar parámetros de red:

- IP
- Máscara de subred
- Gateway
- DNS
- Opciones PXE

Sin embargo, en redes institucionales ya existe un servidor DHCP oficial. Si instaláramos otro servidor DHCP:

- Generaríamos conflictos de red
- Causaríamos pérdida de conectividad en equipos ajenos
- Violaríamos políticas de red

Por eso usamos ProxyDHCP, no DHCP completo, el ProxyDHCP solo envía opciones PXE, pero NO asigna IP, por lo que permite coexistir con el DHCP institucional sin interferencias.

5.5 ¿Qué es TFTP?

TFTP (Trivial File Transfer Protocol) es un protocolo extremadamente simple usado en procesos de arranque. Carece de:

- Autenticación
- Cifrado
- Complejidad

Precisamente por eso es perfecto para PXE, ya que:

- Se implementa fácilmente en firmware
- Usa UDP (rápido y ligero)
- Permite transferir archivos pequeños (pxelinux, kernel, initrd)

El servidor TFTP almacena:

```
/var/lib/tftpboot/pxelinux.0

/var/lib/tftpboot/vmlinuz

/var/lib/tftpboot/initrd.img

/var/lib/tftpboot/pxelinux.cfg/default
```

El cliente los descarga durante el arranque.

5.6 ¿Qué es NFS?

NFS (Network File System) es un protocolo que permite que un cliente monte un directorio remoto como si estuviera en su propio disco.

En este proyecto se usa para montar el sistema operativo completo del cliente desde el servidor.

Ventajas de NFS-root:

- Sin instalación local
- Muy rápido en redes LAN
- Fácil de actualizar (solo se modifica el rootfs en el servidor)
- Permite múltiples clientes con el mismo sistema base

5.7 ¿Qué es una instalación basada en red (Network-based installation)?

Una instalación basada en red permite que un sistema operativo sea instalado o ejecutado utilizando:

- HTTP
- FTP
- NFS
- PXE

Existen dos modalidades:

1. **Instalación remota tradicional**

El cliente instala un sistema operativo desde la red y lo guarda en disco.

2. **Network Booting / Diskless (el de este proyecto)**

El cliente NO instala nada.
El sistema se ejecuta completamente desde el servidor.

Ventaja:
Un solo rootfs sirve para muchos clientes.

5.8 ¿POR QUÉ DEBIAN ES ADECUADO?

Debian es una de las distribuciones más apropiadas para este tipo de proyectos:

Herramientas nativas ideales

- debootstrap — crea sistemas base minimalistas
- nfs-kernel-server — NFS estable
- syslinux/pxelinux — bootloader integrado
- tftpd-hpa — servidor TFTP líder en PXE

Estabilidad y compatibilidad: al ser un sistema operativo que busca siempre coexistir con paquetes y recursos sin romper nada del sistema es que utiliza versiones completamente probadas y estables para cualquier equipo.

Debian funciona bien en hardware antiguo y moderno: debido a su gran comunidad en el proyecto, muchos participantes se han involucrado en toda la programación de controladores para tener una amplia compatibilidad con equipos antiguos y modernos.

Gran documentación: con la alta participación de usuarios alrededor de todo el mundo es posible encontrar manuales adaptados al idioma de cada país, así como su largo trayecto a través del tiempo es que ya se han corregido numerosas cantidades de errores que son comunes y fáciles de arreglar.

Ideal para proyectos académicos: al ser una distribución de Linux que busca ser estable y lo más armoniosa posible con configuraciones, controladores y versiones de programas es lo que la convierte en un sistema operativo excelente para realizar actividades o proyectos escolares.

Comunidad experta en servidores: sumado a la gran cantidad de voluntarios y su vasta documentación en diversos idiomas es posible implementar servidores y sistemas complejos brindando una seguridad robusta en su implementación, además de generar confianza en que no se romperá el sistema fácilmente.

Las tecnologías de red suelen estar mejor soportadas en Debian que en Ubuntu Desktop o Fedora Workstation.

5.9 COMPARACIÓN CON LTSP, FOG PROJECT, CLONEZILLA

LTSP (Linux Terminal Server Project)

- Proporciona entornos thin-client.
- Requiere configuraciones adicionales y es más complejo.
- Adecuado para aulas enteras.
- Mucho más grande que lo necesario para un proyecto académico.

FOG Project

- Enfocado en clonación masiva de equipos.
- No está diseñado para diskless root tradicionales.
- Usa iPXE y servidores web.

Clonezilla

- Herramienta de clonación de particiones.
- No permite arranque diario sin disco.
- Se usa para despliegues únicos, no para ejecución continua.

6 FASE 2 – DOCUMENTACIÓN TÉCNICA

En esta fase se describe el proceso real de implementación del servidor Debian 13, iniciando con su instalación y continuando con la preparación de los servicios necesarios para permitir el arranque de los clientes por red.

6.1 INSTALACIÓN DE DEBIAN 13

6.1.1 Selección e instalación del sistema operativo

El servidor se instaló utilizando Debian 13 con un entorno mínimo (“minimal install”) orientado a servidor, sin entorno gráfico completo, para tener un sistema ligero y fácil de administrar.

Durante la instalación se realizaron las siguientes acciones conceptuales:

- Selección de idioma, zona horaria y distribución de teclado.
- Configuración básica de red usando DHCP (IP asignada por la red de la universidad).
- Creación del usuario administrador y contraseña de root (si se habilitó).
- Particionado automático del disco local del servidor.
- Instalación de herramientas básicas del sistema.

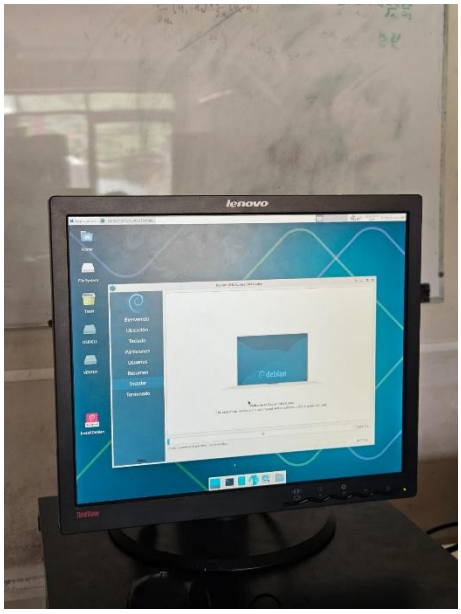


Figura 1. Instalación de Debian 13

Una vez que el sistema arrancó por primera vez, se inició sesión en la terminal y se realizó la actualización básica de paquetes:

```
bash

sudo apt update
```

6.1.2 Instalacion de herramientas necesarias para el proyecto

Desde el servidor Debian se instalaron los paquetes que soportan toda la arquitectura PXE + NFS-root. En algún punto del proceso se utilizó un comando integrado para instalar varias herramientas clave:

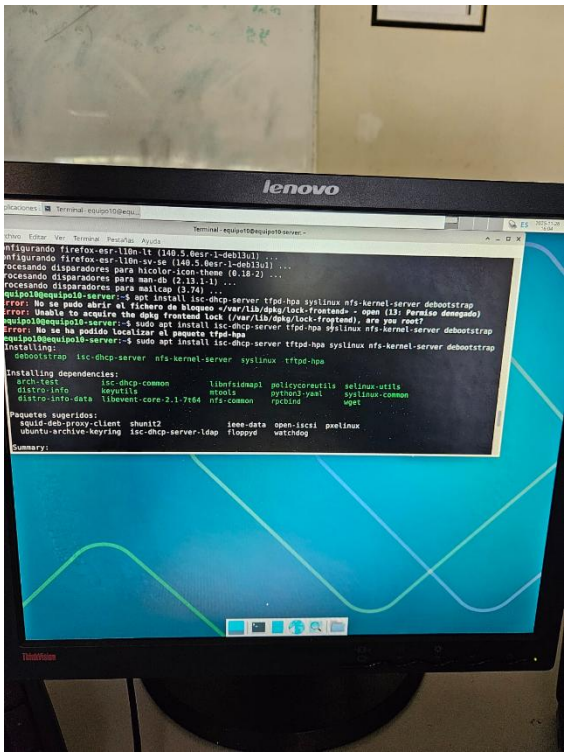


Figura 1.1. Instalación de paquetes

```
sudo apt install isc-dhcp-server tftpd-hpa syslinux nfs-kernel-server debootstrap
```

- tftpd-hpa → servidor TFTP
- syslinux → proporciona pxelinux.0
- nfs-kernel-server → servidor NFS
- debootstrap → creación del root filesystem para los clientes

En fases posteriores también se instaló:

```
sudo apt install dnsmasq
```

para usar dnsmasq como ProxyDHCP.

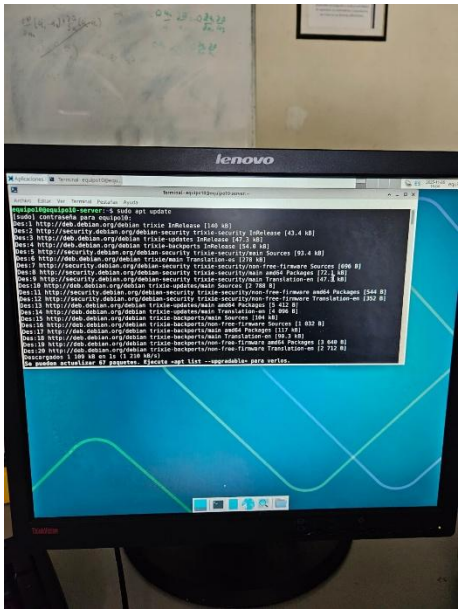


Figura 2.1. Actualización de paquetes

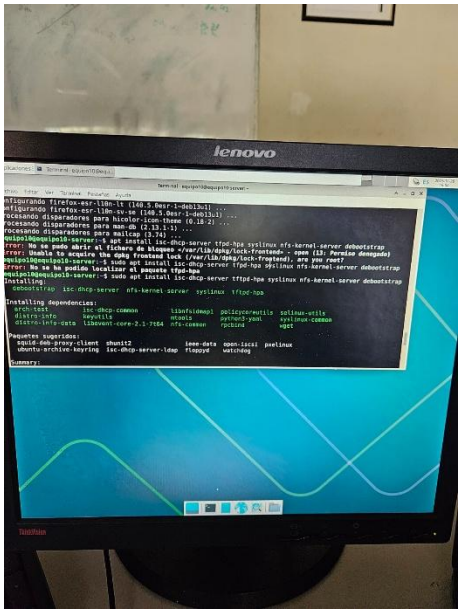


Figura 2.2. Instalación de paquetes necesarios

6.2 CONFIGURACIÓN DE LA RED

Una vez que el servidor estaba instalado, el siguiente paso fue garantizar que:

- Tuviera conectividad de red estable.
- Pudiera resolver nombres de dominio (DNS).
- No interfiriera con la infraestructura de la universidad.

6.2.1 Verificación de la interfaz de red

Primero se verificaron las interfaces disponibles y el estado de la conexión al switch del laboratorio:

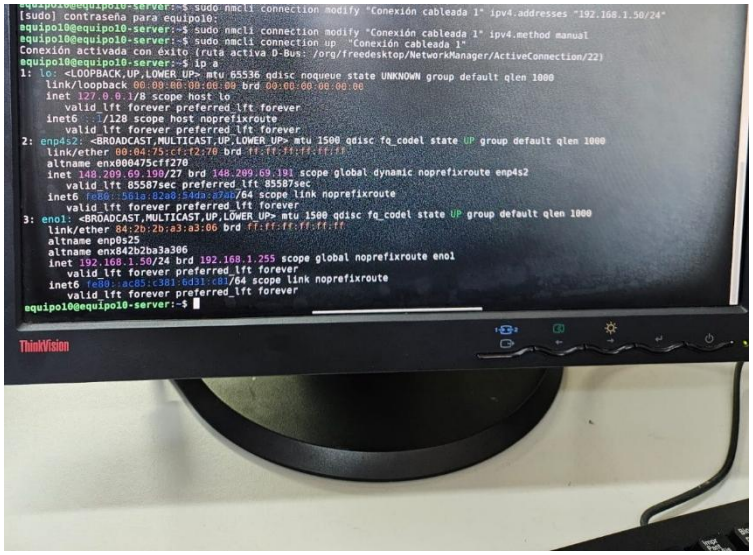


Figura 2.3. Verificación de las interfaces de red

```
ip a
```

Se identificó la interfaz principal conectada al switch como algo similar a:

`eno1`

En el entorno de laboratorio, el servidor recibe una IP por DHCP desde la red institucional.

6.2.2 Problemas iniciales de DNS y "fallo temporal en la resolución del nombre"

En algún momento, al intentar actualizar paquetes o usar debootstrap, aparecieron errores como:

Fallo temporal en la resolución del nombre

Esto indicaba que, aunque había conectividad IP, la resolución DNS no estaba funcionando correctamente.

Para controlarlo, se decidió lo siguiente:

- Evitar que NetworkManager sobreescribiera `/etc/resolv.conf`.
- Definir manualmente los servidores DNS.

a) Configuración de NetworkManager

Se editó el archivo de la siguiente ruta:

```
sudo nano /etc/NetworkManager/NetworkManager.conf
```

Y se dejó de la siguiente forma:

```
[main]

dns=none
```

Esto indica que NetworkManager no debe gestionar el DNS del sistema.

Después se reinició NetworkManager:

```
sudo systemctl restart NetworkManager
```

b) Configuración explícita de DNS

Se editó el archivo de resolución de nombres:

```
sudo nano /etc/resolv.conf
```

Y se agregaron DNS públicos:

```
nameserver 8.8.8.8

nameserver 1.1.1.1
```

Para evitar que este archivo fuera modificado por otros servicios, se llegó a usar:

```
sudo chattr +i /etc/resolv.conf
```

Con esto, `/etc/resolv.conf` queda protegido contra cambios indeseados.

c) Verificación de conectividad y DNS

Se probaron pings hacia IPs públicas y dominios:

```
ping -c 4 8.8.8.8

ping -c 4 deb.debian.org
```

Una vez que ambos funcionaron, el servidor quedó listo para descargar paquetes y realizar el `debootstrap` del rootfs para los clientes.

6.3 CONFIGURACIÓN DE SERVICIOS

En esta etapa se configuraron, en este orden lógico:

1. DHCP / ProxyDHCP (realmente ProxyDHCP con dnsmasq)
2. TFTP (tftpd-hpa)
3. PXE (pxelinux.0 + configuración)
4. NFS (exportación del rootfs)
5. (Opcional) Firewall y NAT

6.3.1 DHCP (dhcpd.conf)

Inicialmente se instaló el paquete:

```
sudo apt install isc-dhcp-server
```

El plan original era editar:

```
sudo nano /etc/dhcp/dhcpd.conf
```

para definir un "dhcpd.conf" clásico con un rango propio.

Sin embargo, debido a la política de la red de la universidad, se identificó que:

- Ya existe un servidor DHCP oficial.
- Levantar otro servidor DHCP en la misma red puede causar conflictos graves.

Por esa razón, se descartó el uso de `isc-dhcp-server` y se tomó la decisión técnica de:

Usar dnsmasq como ProxyDHCP, de forma que no entregue IPs y solo proporcione opciones PXE a los clientes.

6.3.2 TFTP (tftp-server)

El servidor TFTP se implementó con el servicio `tftpd-hpa`.

a) Instalación de TFTP

Esto se logró con:

```
sudo nano /etc/dhcp/dhcpd.conf
```

b) Configuración básica de TFTP

Se editó:

```
bash
```

```
sudo nano /etc/default/tftpd-hpa
```

Con un contenido típico (ejemplo):

```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/var/lib/tftpboot"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="--secure"
```

Se creó el directorio raíz:


```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/var/lib/tftpboot"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="--secure"
```


Se creó el directorio raíz:

```
sudo mkdir -p /var/lib/tftpboot
```

```
sudo chown -R tftp:tftp /var/lib/tftpboot
```

6.3.3 PXE (pxelinux.0)

Una vez operativo TFTP, se colocaron los archivos de PXE en `/var/lib/tftpboot`.

a) Copia de pxelinux.0

`pxelinux.0` viene del paquete `syslinux`.

Tras la instalación:

```
sudo apt install syslinux
```

Se copió `pxelinux.0` al directorio TFTP:

```
sudo cp /usr/lib/PXELINUX/pxelinux.0 /var/lib/tftpboot/
```

b) Creación del directorio de configuración PXE

```
sudo mkdir -p /var/lib/tftpboot/pxelinux.cfg
```

c) Primera versión de configuración: instalador de Debian (netboot)

En la primera etapa, el cliente PXE llegaba a una pantalla de instalador de Debian "netinstall" y mostraba mensajes como:

- Solicitud de dominio
- Selección de espejo/mirror
- Error de "Réplica de Debian inválida"

Esta fue una evidencia importante de que:

- ✓ PXE funcionaba

- ✓ TFTP entregaba pxelinux.0
- ✓ El cliente se comunicaba correctamente con el servidor

Sin embargo, este flujo correspondía a instalación por red, no al modo diskless NFS-root deseado.

d) Versión final de configuración para diskless (NFS-root)

Tras crear el rootfs y copiar el kernel e initrd, se modificó:

```
sudo nano /var/lib/tftpboot/pxelinux.cfg/default
```

Con contenido similar a:

```
DEFAULT linux

LABEL linux

    KERNEL vmlinuz

    INITRD initrd.img

    APPEND   root=/dev/nfs   nfsroot=SERVIDOR_IP:/srv/nfs/rootfs   rw
ip=dhcp
```

Ejemplo:

```
 text

APPEND   root=/dev/nfs   nfsroot=148.209.69.168:/srv/nfs/rootfs   rw
ip=dhcp
```

Esta línea indica al kernel que:

La raíz del sistema (`root=/dev/nfs`) se montará por NFS desde

`148.209.69.168:/srv/nfs/rootfs`.

El cliente obtendrá sus parámetros IP por DHCP (`ip=dhcp`).

6.3.4 NFS (/etc/exports) y creación de roofts

6.3.4.1 Creación del directorio rootfs

Primero se definió el directorio donde viviría el sistema de archivos de los clientes:

```
bash

sudo mkdir -p /srv/nfs/rootfs
```

6.3.4.2 Creación del Sistema Debian mínimo con debootstrap

Se ejecutó:

```
bash

sudo debootstrap --arch=amd64 stable /srv/nfs/rootfs http://deb.debian.org/debian
```

Este comando descarga e instala un sistema Debian base dentro de `/srv/nfs/rootfs`.

6.3.4.3 Preparación del roofts desde chroot

Se ingresó en el entorno chroot:

```
sudo chroot /srv/nfs/rootfs
```

Dentro del chroot se ejecutaron comandos como:

```
apt update

apt install linux-image-amd64 nfs-common

echo "cliente-diskless" > /etc/hostname

printf "auto eth0\niface eth0 inet dhcp\n" > /etc/network/interfaces
```

Se verificó que en /boot se hubieran generado archivos como:

```
ls -l /boot

# vmlinuz-...

# initrd.img-...
```

Luego se salió del chroot:

```
exit
```

6.3.4.4 Copiar kernel e initrd al TFTP

Una vez confirmada su existencia dentro del rootfs:

```
sudo cp /srv/nfs/rootfs/boot/vmlinuz-* /var/lib/tftpboot/vmlinuz

sudo cp /srv/nfs/rootfs/boot/initrd.img-*
/var/lib/tftpboot/initrd.img
```

6.3.4.5 Configuración de NFS en /etc/exports

Se editó:

```
sudo nano /etc/exports
```

Y se añadió:

```
/srv/nfs/rootfs *(rw,no_root_squash,no_subtree_check)
```

Se aplicó la configuración:

```
sudo exportfs -ra

sudo systemctl restart nfs-kernel-server
```

Y NFS quedó funcionando.

6.4 ARQUITECTURA DE RED

La arquitectura final puede describirse así:

1. Servidor DHCP institucional
    - Asigna IP a todos los equipos del laboratorio, incluyendo el servidor Debian y los clientes diskless.
2. Servidor Debian 13 (PXE/NFS)
    - Conectado al mismo switch.
    - Ejecuta:
        - `dnsmasq` como ProxyDHCP (solo opciones PXE).
        - `tftpd-hpa` como servidor TFTP.
        - `nfs-kernel-server` con export `/srv/nfs/rootfs`.
3. Switch del laboratorio
    - Punto de interconexión de servidor, clientes y resto de la red universitaria.
4. Clientes sin disco (diskless)
    - Configurados en BIOS/UEFI para Network Boot (PXE).
    - No requieren sistema operativo local.

6.4.1 Flujo de arranque resumido

1. Cliente enciende y hace PXE → solicita IP (`DHCPDISCOVER`).
2. DHCP de la universidad le da una IP.
3. `dnsmasq` (ProxyDHCP) responde con:
    - Nombre de archivo de arranque: `pxelinux.0`
    - IP del servidor TFTP (Debian).
4. Cliente pide `pxelinux.0` por TFTP a `tftpd-hpa`.
5. `pxelinux.0` carga el menú definido en `/var/lib/tftpboot/pxelinux.cfg/default`.
6. Se descargan `vmlinuz` e `initrd.img` desde TFTP.
7. El kernel se inicia, el initramfs monta `/srv/nfs/rootfs` por NFS.
8. El cliente finalmente muestra un prompt de login de Debian, **sin disco local**.

7 FASE 3 – PRUEBAS VIRTUALES

Aunque el proyecto estaba enfocado en hardware real dentro del laboratorio, se contempló la posibilidad de realizar pruebas virtuales previas utilizando entornos como VirtualBox o QEMU.

7.1 CONSIDERACIONES SOBRE PRUEBAS VIRTUALES

Un entorno virtual hubiera permitido:

- Validar la configuración de PXE sin depender de hardware físico.
- Simular múltiples clientes diskless.
- Probar el flujo TFTP → PXE → kernel → NFS-root.

Sin embargo, se determinó que:

- La red institucional tenía restricciones.
- Los switches del laboratorio no soportaban VLANs propias para aislamiento virtual.
- VirtualBox en las máquinas disponibles no permitía un modo puente adecuado sin interferir con la red real.

Por ello, se decidió trabajar directamente con clientes físicos, lo cual fortaleció la comprensión del proceso al enfrentar problemas reales de red, DNS, servicios concurrentes y configuración del servidor.

7.2 PRUEBAS CONCEPTUALES REALIZADAS SIN VIRTUALIZACIÓN

Estas pruebas se realizaron en el servidor antes de proceder al cliente físico:

a) Verificación del servicio TFTP

```
sudo systemctl status tftpd-hpa
```

Y desde el propio servidor:

```
tftp localhost

get pxelinux.0
```

Un archivo descargado correctamente confirmó que TFTP estaba operativo.

b) Verificación de ProxyDHCP (dnsmasq)

```
sudo dnsmasq --test
```

El resultado:

dnsmasq: syntax check OK

confirmó que no había errores en la configuración.

c) Verificación de la exportación NFS

```
sudo exportfs -v
```

Y:

```
showmount -e
```

Esto confirmó que `/srv/nfs/rootfs` estaba exportado correctamente.

8 FASE 4 – IMPLEMENTACIÓN FÍSICA

Esta es la fase más importante del proyecto, donde se emplearon los equipos reales del laboratorio:

- Un servidor con Debian 13
- Clientes sin disco (o arrancando sin usar su disco)
- Un switch institucional compartido con otros alumnos
- Una red DHCP institucional con restricciones

8.1 PREPARACIÓN DEL CLIENTE FÍSICO

El cliente se configuró en BIOS/UEFI:

1. Activar Network Boot (PXE).
2. Subir PXE al primer lugar en el boot order.
3. Seleccionar modo Legacy PXE o UEFI PXE, según soporte.

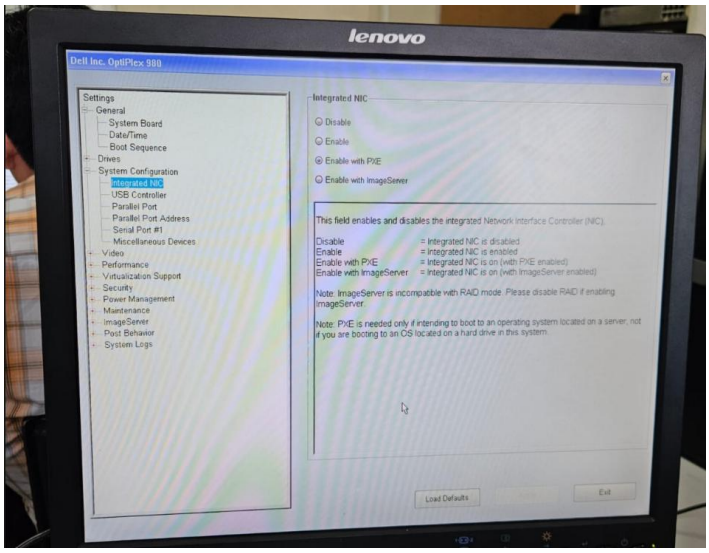


Figura 3.1. Habilitación del arranque PXE

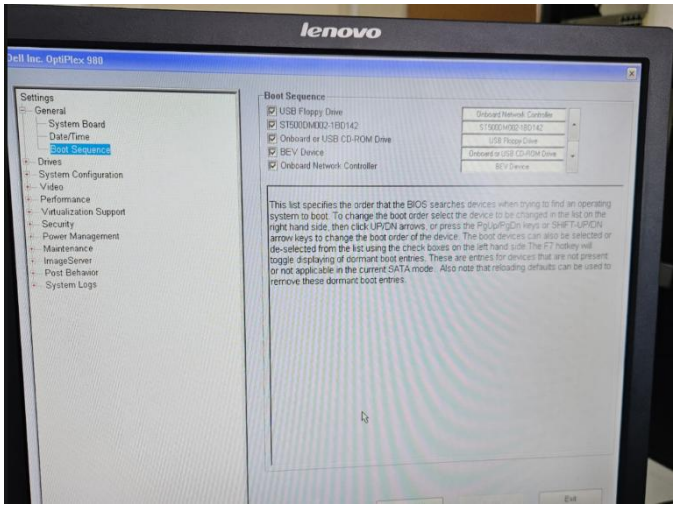


Figura 3.2. Ajuste en el orden de arranque

8.2 ARRANQUE INICIAL DEL CLIENTE

Al encender el cliente, se observaron estas etapas:

1. PXE ROM inicializa la tarjeta de red.
2. Envía un paquete DHCPDISCOVER con identificador PXE.
3. El servidor DHCP institucional responde con una IP.
4. El servidor Debian responde como ProxyDHCP, indicando:
    - Archivo: `pxelinux.0`
    - IP del servidor TFTP

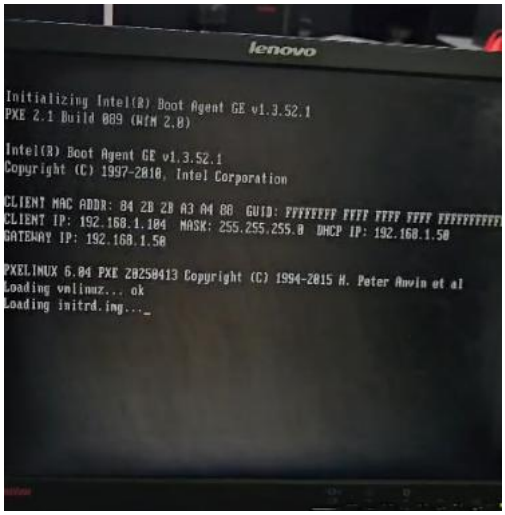


Figura 3.3. Arranque del cliente

8.3 CARGA DE PXELINUX.0 Y MENÚ PXE

El cliente mostró mensajes como:

Downloading pxelinux.0 via TFTP...

Esto confirmó:

- ✓ TFTP funcional
- ✓ dnsmasq entregando opciones PXE correctamente

Si `pxelinux.0` faltaba, TFTP fallaba; si `dhcp-range` estaba mal escrito, PXE no funcionaba. Todos estos errores se resolvieron en fases previas.

8.4 CARGA DEL KERNEL E INITRD

Tras el menú PXE, el cliente descargó:

- `vmlinuz`
- `initrd.img`

Ambos provenientes del rootfs generado por `debootstrap`.

Si el kernel no existía dentro del rootfs, aparecía el error:

8.5 MONTAJE DEL ROOTFS MEDIANTE NFS

Luego, el cliente mostró:

```
IP-Config: Got DHCP answer

IP-Config: eth0 hardware address XX:XX:XX:XX:XX:XX

Root-NFS: Mounting 148.209.69.168:/srv/nfs/rootfs
```

Esto es el indicio más claro de éxito, porque significa:

- ✓ NetworkManager y resolv.conf del servidor estaban correctos
- ✓ NFS estaba exportado correctamente
- ✓ El kernel tenía los módulos nfsroot disponibles

8.6 INICIO DEL SISTEMA DEBIAN EN EL CLIENTE

Finalmente, el cliente arrancó el sistema root desde el servidor y presentó:

```
Debian GNU/Linux ...

login:
```

Posteriormente se instaló una interfaz gráfica que permitiera explorar con mayor facilidad el sistema operativo, sin tener que estar usando consta ntemente la línea de comandos

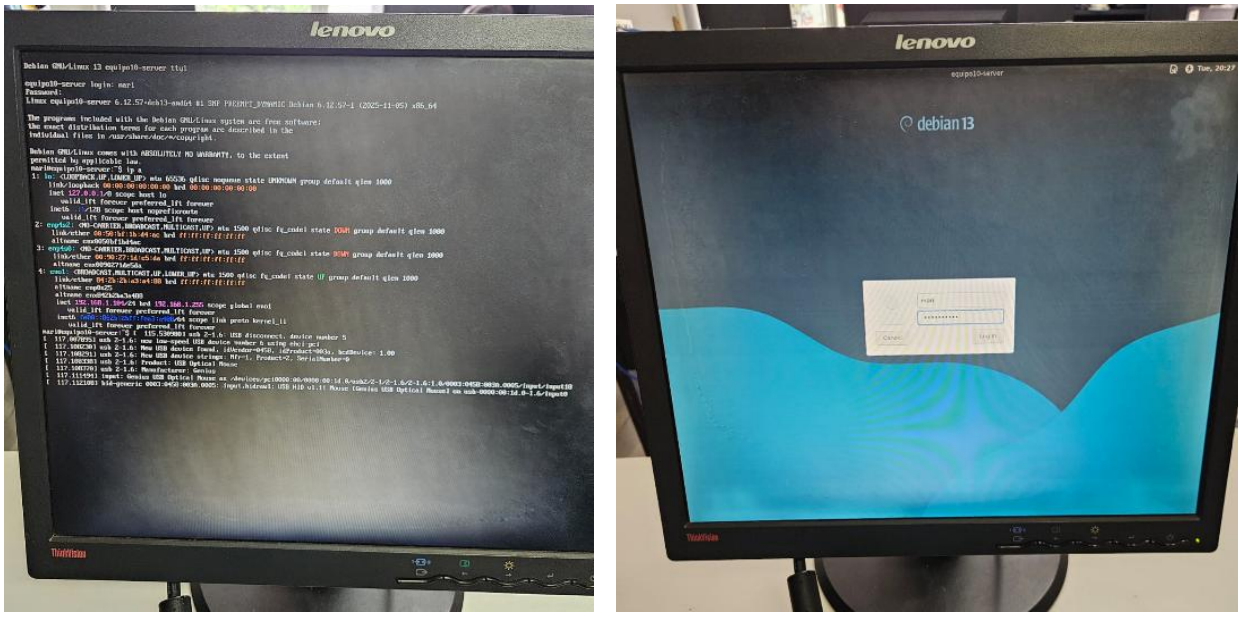


Figura 3.4. Cliente sin interfaz gráfica (izquierda) y con interfaz gráfica (derecha)

9 CONCLUSIONES

La implementación de un entorno de arranque por red para clientes sin disco mediante PXE y NFS en Debian 13 permitió comprender de manera integral el funcionamiento de servicios esenciales en infraestructuras, tales como ProxyDHCP, TFTP, PXELinux, NFS y la creación de sistemas mínimos con debootstrap. A lo largo del proyecto se enfrentaron y resolvieron problemas reales relacionados con DNS, conflictos de puertos, configuraciones incorrectas y dependencias del sistema, fortaleciendo las competencias técnicas necesarias para diagnosticar y corregir fallas en entornos de red. El resultado final, en el que un cliente físico logró arrancar completamente sin disco duro utilizando únicamente los servicios del servidor Debian, demuestra la correcta integración de todos los componentes y valida la arquitectura configurada. Este proyecto no solo cumplió con los objetivos planteados, sino que proporcionó una experiencia práctica valiosa en diseño, despliegue y administración de servicios de red avanzados, consolidando conocimientos fundamentales para el desarrollo profesional en sistemas y redes.

Enlaces a las evidencias funcionales

Evidencias proyecto gdti