Cameron Stewart

# Network-Based Deployment
## Using Preboot Execution Environment

Bachelor's Thesis

Bachelor of Engineering

Information Technology

2021


South-Eastern Finland
University of Applied Sciences

South-Eastern Finland University of Applied Sciences

| Author (authors) | Degree title | Time |
|---|---|---|
| Cameron Stewart | Bachelor of Engineering | December 2021 |
| **Thesis title**<br><br>Network-Based Deployment<br>Using Preboot Execution Environment | | 30 pages |
| **Commissioned by**<br>South-Eastern Finland University of Applied Sciences | | |
| **Supervisor**<br><br>Matti Juutilainen | | |


**Abstract**

The objective of this thesis was to build a server that would provide bootable media over a network, which could then be deployed to computers using the Preboot Execution Environment standard, automating the installation of an operating system to a computer. The aims from this project were to learn how the standard works and is implemented. The basis of the results aimed to provide conclusions to whether implementing a PXE server in certain working environments may be an effective method for mass operating system deployment to computers.

The research was conducted by creating a PXE and iPXE server and using a virtual machine to install an operating system via PXE. Quantitative data was collected through comparing the time taken to install an operating system between the different PXE implementations. This data was then compared with the time taken for installing an operating system with a USB and DVD.

The results from this study showed that PXE could reduce installation times when deploying an operating system to a computer against other methods of installing an operating system. The server could also potentially streamline mass deployments of operating systems in a business environment. Using HTTP rather than just TFTP made operating system deployments to computers slightly quicker.



**Keywords**

PXE, deployment, operating system

**CONTENTS**

ABBREVIATIONS ..... 4

1 INTRODUCTION ..... 5

2 THEORETICAL BACKGROUND ..... 6

2.1 Operating System ..... 6

2.2 Removable Media ..... 7

2.3 Network Boot (PXE) ..... 8

2.4 DHCP ..... 9

2.5 TFTP ..... 11

2.6 PXE Boot Process ..... 12

3 METHOD ..... 16

4 IMPLEMENTATION ..... 17

5 DISCUSSION ..... 24

6 CONCLUSION ..... 26

LIST OF FIGURES AND TABLES ..... 28

REFERENCES ..... 29

**ABBREVIATIONS**

AoE – ATA over Ethernet

BIOS – Basic Input Output System

CD – Compact Disc

DHCP – Dynamic Host Configuration Protocol

FCoE – Fibre Channel over Ethernet

GB – Gigabyte

HTTP – Hypertext Transfer Protocol

IETF – Internet Engineering Task Force

IP – Internet Protocol

iSCSI – Internet Small Computer Systems Interface

MB - Megabyte

MTU – Maximum Transmission Unit

NBP – Network Boot Program

NIC – Network Interface Card

OEM – Original Equipment Manufacturer

OS – Operating System

POST – Power-On Self Test

PXE – Preboot Execution Environment

RAM – Random-access memory

ROM – Read-Only Memory

TFTP – Trivial File Transfer Protocol

UDP – User Datagram Protocol

UEFI – Universal Extensible Firmware Interface

USB – Universal Serial Bus

VM – Virtual Machine

VLAN – Virtual Local Area Network

# 1 INTRODUCTION

An operating system manages any hardware and software on a computer. It undertakes many basic tasks such as file, memory and process management, handling input and output, and controlling peripheral devices. The operating system coordinates which of the computer's resources are needed for software running on a machine. (University of Wollongong 2021.)

Within private households, personal computers are typically bought with a pre-installed operating system. New or different operating systems however can be installed with removable bootable media, and with the increasing availability of high-speed networking infrastructure (Traficom 2019), downloading and installing the latest operating systems has become a quick and easy solution.

The situation on the other hand for businesses or schools may differ with there being a substantially larger number of computers to manage. Vendors or OEM partners may sell their hardware without an OS license; licensing may be typically more cost effective when using a corporate style license but leaves the issue of installing operating systems to a large amount of computers. Businesses may also use custom operating systems whereby the bootable image is not readily available from the internet and is installed locally on premise. The process of installing an operating system in this manner on a large number of computers can be time-consuming and costly.

This thesis will investigate the PXE standard, commonly used in deploying and the automated provisioning of operating systems to servers and workstations over a network. It will explore how the standard works and an alternative implementation of the standard, understanding the use case of PXE. Research material will be gathered by deploying a PXE server to deliver bootable media to client computers, in the form of virtual machines for which an operating system will be installed. This will follow on with comparisons showing the effectiveness of

other implementations of the standard, namely the open-source variant iPXE using HTTP in the process and then comparisons made against more traditional forms of installation media.

# 2 THEORETICAL BACKGROUND

Theoretical background studies were undertaken to understand the components to the Preboot Execution Environment standard and how they worked.

## 2.1 Operating System

An operating system is a piece of software that manages a computer's hardware and software resources. Thus, an operating system acts as an interface between the computer hardware and the user, helping them to communicate with the computer without the need to know the computer's language or machine code.

Booting an operating system works by loading the kernel into main memory and starting its execution. The CPU is instructed to reset, and the instruction register is loaded with a predefined memory location for where execution is started. Prior to this, the POST process occurs initially when a computer is booted. The initial bootstrap program is loaded from the BIOS read-only memory which runs a set of diagnostics, making sure all the components in the machine are in good standing. After completing the diagnostics, the process of initializing the physical components of the computer occurs. This then leads to starting the Operating System loader, which in effect starts the operating system. This process involves setting up the needed data structures in memory, setting the registers in the CPU and finally creating and starting the first user-level program in most cases being the Desktop environment where a user can interact with the operating system. The operating system from this point runs only in response to interruptions. (Kansas State University 2015.)

Some of the ways for installing an operating system include using removable media or network booting to an installation. Cloud computing providers also offer solutions whereby bare metal servers can be automatically provisioned with an

operating system. The installation process of an operating system is like that of the boot process, though the software, once loaded, copies operating system files to a more permanent storage solution such as a hard drive which can then be booted from at a later stage.

## 2.2 Removable Media

With the advent of personal computers, one of the issues that arose was how to store data. Different physical forms of removable media allowed software (operating systems) to run on some of the first personal computers. With the introduction of low-cost read-write storage hard disks as boot media by the 1980s, removable media saw a transition from storing an operating system to becoming a medium that operating systems could be distributed on for installation to personal computers. This was due to the first compact discs and floppy disks being too slow to run complex operating systems and not being designed to boot from removable media.

Examples of removable media for installing an operating system typically come in the form of optical discs, memory cards, USB flash and external hard disk drives. Forms of removable media that are less used today include floppy disks and magnetic tapes. Installation of these types of media works by having an operating system with installation files stored on a form of removable media. The installation files are loaded first into the memory and then carry out the process of either booting the operating system, or more commonly by copying the operating system to the selected boot drive to install the operating system.

Removable media provides a small and portable solution for installing operating systems. With this, data can be read or transferred at a relatively quick speed. Removable media allow operating systems to be easily distributed. This allows for a greater reach and install base; with the infancy of the internet, getting access to or downloading an operating system over the internet was not a common, easy, or cost-effective method due to the internet infrastructure at the time when compared to distributing via the use of removable media. Many of the types of removable media used for operating system installations such as

compact discs and USB flash storage are already commonly used for others means, making them easy to use for end users.

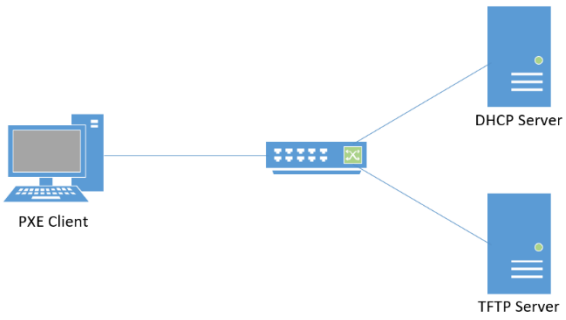
Removable media can also be used for live operating systems, whereby a complete bootable operating system runs directly from removable optical or flash media into a computer's memory. This allows a user to run an operating system without installing it. Additional uses could also include but are not limited to testing hardware and software, computer forensics, system repairs and restoration, providing a secure platform for servers where files cannot be altered.

### 2.3 Network Boot (PXE)

PXE, abbreviated for Preboot Execution Environment, is a standardized way to boot files, most commonly an operating system over a network. PXE is most commonly used within large-scale data centers to automatically deploy boot images to new servers for installation or, in some cases, run diskless servers/ nodes whereby the boot image is loaded into memory rather than installed to storage. (TechsavvyProduction 2019.)

Preboot Execution Environment was fore fronted by Intel in the early 1990s as part of the Wire for Management initiative with the goal of allowing a newly built computer to be controlled by a master computer over a network by gaining access to its hard drive and installing software.

The PXE framework is implemented on the client side either within a Network Interface Card BIOS extension or in UEFI code. This firmware includes a minimal UDP/IP stack, a Preboot client module using DHCP and a TFTP client module which together form the application programming interfaces used by the network boot programs; This allows network boot programs to interact and work with services offered by a PXE server. (ibid.)  A high-level overview of PXE implementation can be seen in below. (Figure 1.).




The PXE boot process relies on different internet protocols working with each other. These protocols include DHCP, to provide network connectivity and the location of the initial bootstrap program, and TFTP, used for requesting and downloading the initial bootstrap program. (RackN & Digital Rebar 2021.)

### 2.4 DHCP

DHCP or Dynamic Host Configuration Protocol is a network management protocol used on Internet Protocol networks to automatically assign IP addresses and other communication parameters to devices on a network. This is done by utilizing a client-server model whereby a DHCP server assigns IP addresses to devices on a network. The technology removes the need for configuring network settings manually on devices that are in a network. A DHCP server manages a pool of IP addresses and thus any client configuration parameters such as the default gateway, domain name and domain name servers, and time servers.

When a device is connected to a DHCP-enabled network, the DHCP client software sends out a DHCP broadcast query that will request information from a DHCP server on how to join the network. When the DHCP server has received the request, it can then respond accordingly with information for each client, either being configured by an administrator or with a specific address. This information also includes the allotted time for which the address lease is valid. DHCP clients usually query this information after booting to join a network, though also periodically thereafter to prevent the expiration of this information, and the address being available in the pool again for other clients to use.

DHCP uses a connectionless service model using UDP. The operation of DHCP is broken into four phases:

- Discovery – A DHCP client broadcasts a DHCPDISCOVER message on a network, stating their presence and asking for an IP address lease to join the network.
- Offer – The DHCP server receives the DHCPDISCOVER message from the client and reserves an IP address for the client. The DHCP server sends a DHCPOFFER message to the client, offering the leased IP address that has been reserved.
- Request – The client receives the offer from the DHCP server with an IP address. It first checks with an ARP request across the network to confirm that the offered address is not in use. If the address is available, the client replies with a DHCPREQUEST accepting the terms of the leased address from the DHCP server.
- Acknowledgement – The DHCP server receives the DHCPREQUEST from the client and sends back a DHCPACK packet, of which includes the lease duration and other configuration information that may have been requested priorly. The client now can use the assigned IP address.

The PXE boot process is based off the DHCP protocol. The Bootstrap protocol of which DHCP was built upon, allows systems to obtain an IP address and remote boot from a network, while DHCP develops upon this by adding features such as dynamic IP addressing and option fields for passing system information. The PXE boot process extends from DHCP by adding the necessary information to network boot a computer. This includes the client vendor and class that enables the PXE server to select an image based on the client accessing the server. (Intel 2021.)

PXE was designed to be compatible with networks that had already had a DHCP server integrated to them. This allows for PXE and DHCP to be provided by separate servers without interfering with each other. A proxyDHCP server provides strictly PXE functionality to a non-PXE DHCP environment and does not assign IP addresses. (TechsavvyProduction 2019.)

## 2.5 TFTP

TFTP is a protocol that provides basic file transfer function with no user authentication (IBM 2021). It allows a client to download or receive a file from a remote host. Being a simple protocol, it can be implemented with a small amount of memory and therefore mainly used to transfer firmware and configuration files to network appliances such as routers, thin clients and firewalls which do not have data storage devices. Due of the lack of authentication, encryption mechanisms and general security, TFTP is commonly used on local area networks only. (University of Aberdeen 2011.)

TFTP has a defined transfer file size limit of 4GB calculated by 65,535 bytes/block x 65,535 blocks. However, if an IP packet's size exceeds the minimum MTU within a network at any point, IP fragmentation and reassembly will ensue, adding more overhead and, in the case of some PXE boot ROMs, leading to the transfer failing when the minimalist IP stack does not implement IP fragmentation and reassembly. (IETF 2009.) The transfer file size limit of a TFTP packet when kept with the Ethernet MTU is roughly 92 MB though most modern clients and servers today support block number roll-over, which sets the block counter back to 0 or 1 after exceeding the maximum number of blocks, thus enabling an unlimited transfer file size (CompuPhase 2019).

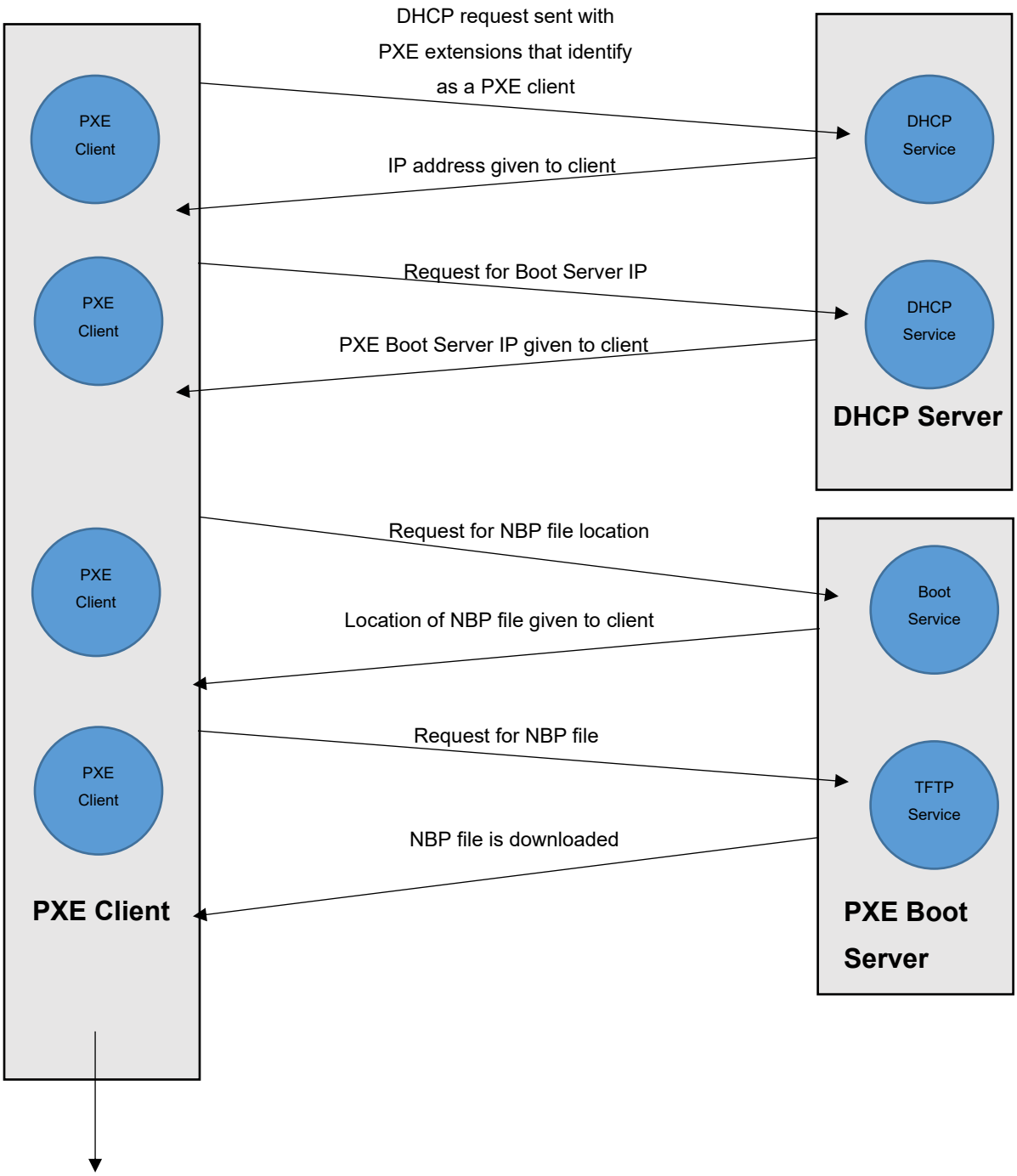
TFTP is implemented on top of UDP and must supply its own transfer and session support, with each file transferred constituting an independent exchange. Originally, the transfer of data is performed in a lockstep, thus meaning only one packet of either a block of data or 'acknowledgement' is ever in flight on the network at any time. Due to the lack of windowing, whereby a large amount of uninterrupted data blocks is sent before the transfer is paused to wait for the resultant acknowledge, TFTP has a low throughput over high latency links. The latest standard published by the IETF for TFTP allows for a "client and server to negotiate a window size of consecutive blocks to send as an alternative for replacing the single-block lockstep schema" (IETF 2015), which significantly improves performance for transfers.

## 2.6 PXE Boot Process

For a device to a PXE boot, relevant instructions are needed, with the most common method involving the use of a DHCP server to store and serve this information. The boot method is as follows:

- The client first initiates PXE boot by being selected as a boot option in the BIOS settings or a fallback option when there is no available boot media. A DHCP broadcast is then sent across a network stating the need for a PXE boot.
- The DHCP server, having received the broadcast, replies with an available IP address the client can use. The DHCP server may have knowledge of how the client can PXE boot, which is also included within the information that is sent to the client.
- The client/device replies to the DHCP server with an acknowledgement that it will use the provided IP address.
- The client then sends a request to the PXE boot server, requesting a boot file (NBP) that can be used. This information may be provided by the prior DHCP server for where it can be located.
- The PXE boot server replies with the IP address of the TFTP server and the location of the NBP file which can be downloaded. Once downloaded, the NBP can be launched by the client.

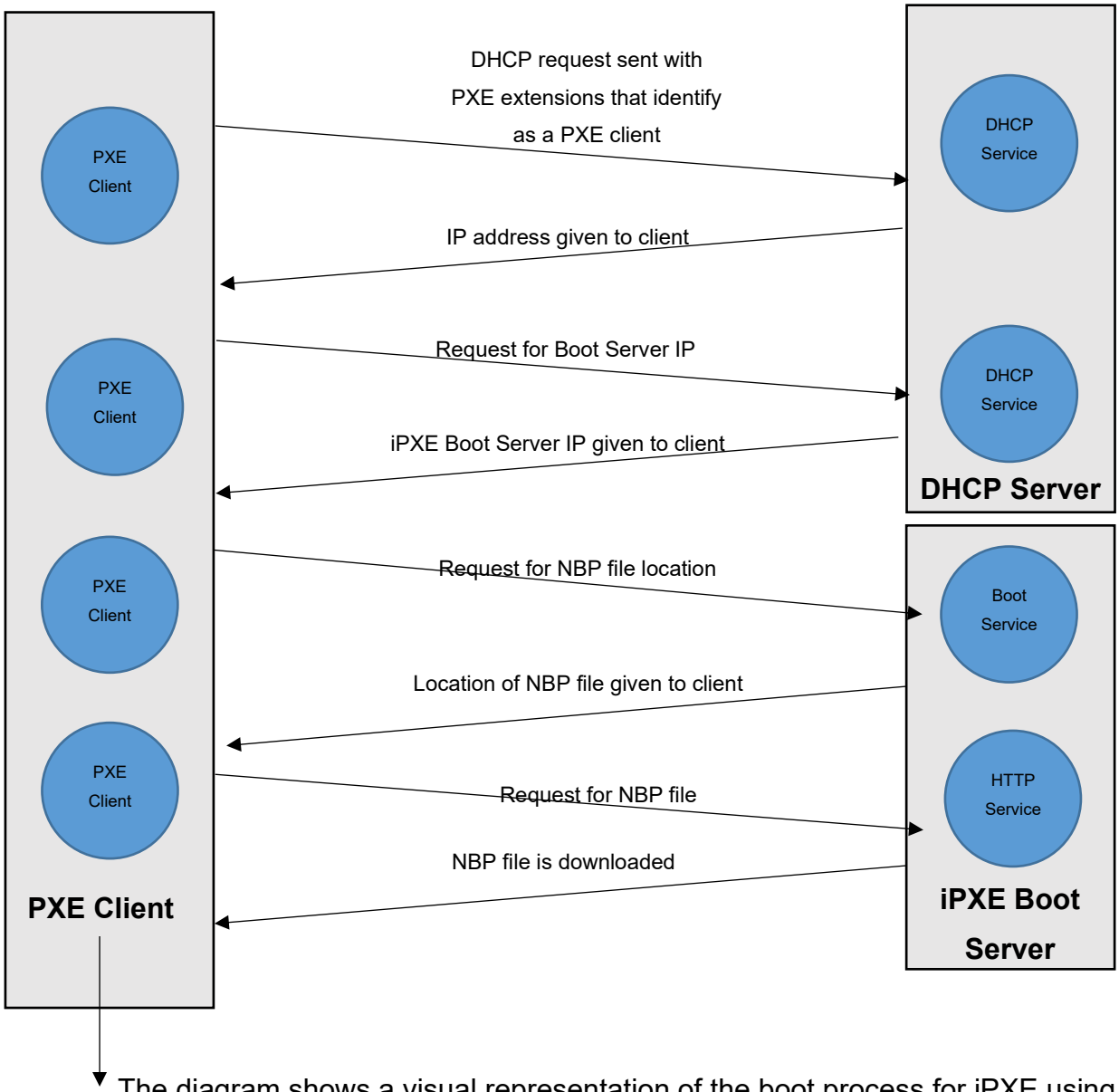
IP helper addresses are used when a PXE client, DHCP server, and PXE server are located on different subnets or VLANs, allowing PXE network requests to be routed to the appropriate server. A helper address is also required to be able to support both BIOS and UEFI firmware configurations. (TechsavvyProduction 2019.)



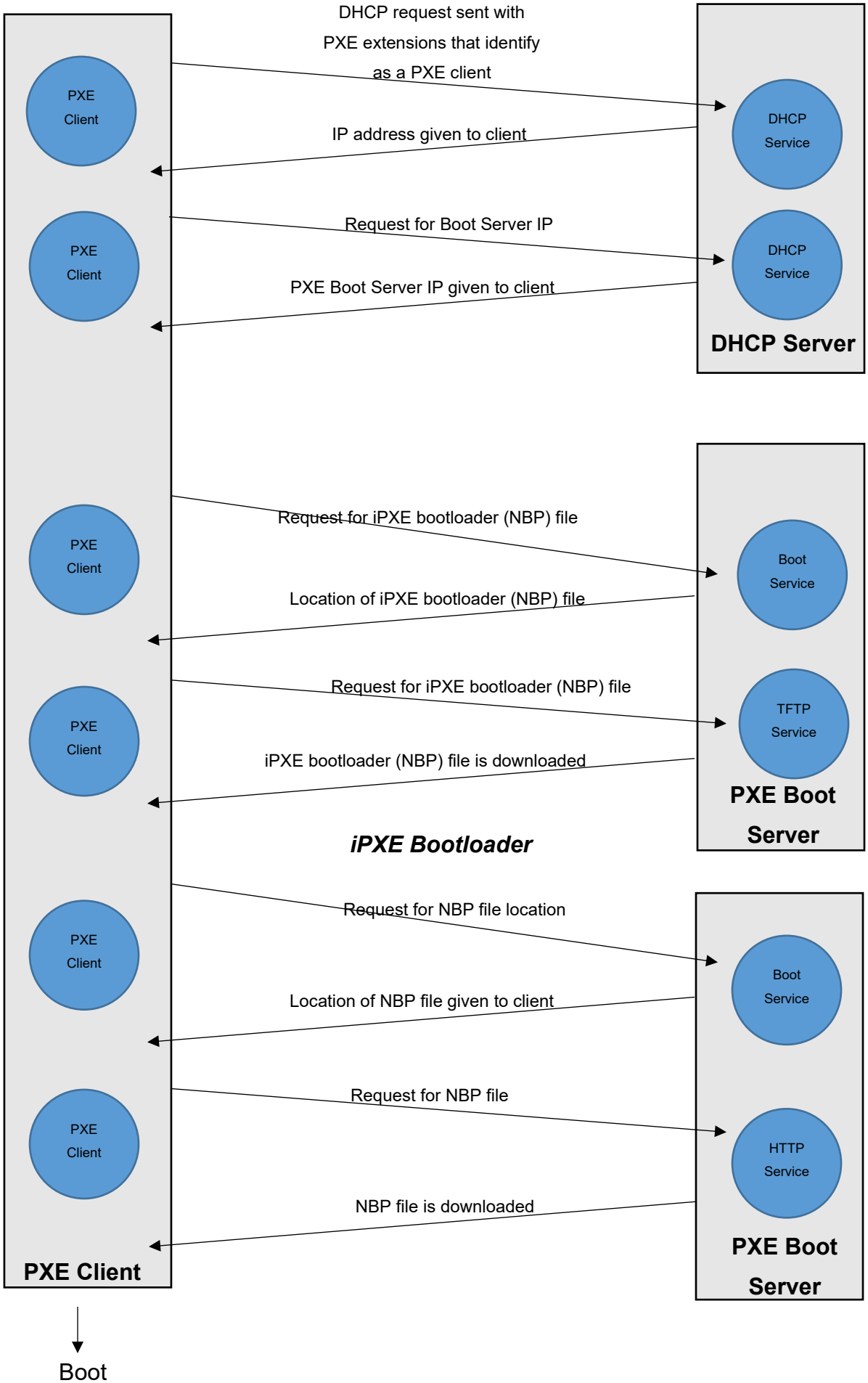

The diagram shows a visual representation of the boot process for PXE. (Figure 2.)

Alternative PXE projects expand upon the initial foundations of the boot process, by adding usability for additional protocols such as HTTP, iSCSI, AoE and FCoE. An open-source implementation of PXE, iPXE can be booted by re-flashing the

existing PXE ROM on a supported network interface card (iPXE 2021). Alternatively, iPXE can also be booted through chainloading into the iPXE binary using a standard PXE boot; this feature is aimed at scenarios where there may be a large number of computers that already have some form of PXE implementation, thus avoiding the need to reflash the PXE ROM on each machines network card. When using a ROM, iPXE network stack can be implemented allowing for removal of TFTP. (iPXE 2018.) The boot method is as follows:




The diagram shows a visual representation of the boot process for iPXE using a ROM. (Figure 3.)


DHCP request sent with
PXE extensions that identify
as a PXE client
IP address given to client
Request for Boot Server IP
PXE Boot Server IP given to client
PXE
Client
PXE
Client
DHCP
Service
DHCP
Service
DHCP Server
Request for iPXE bootloader (NBP) file
Location of iPXE bootloader (NBP) file
Request for iPXE bootloader (NBP) file
iPXE bootloader (NBP) file is downloaded
PXE
Client
PXE
Client
Boot
Service
TFTP
Service
PXE Boot
Server
iPXE Bootloader
Request for NBP file location
Location of NBP file given to client
Request for NBP file
NBP file is downloaded
PXE
Client
PXE
Client
Boot
Service
HTTP
Service
PXE Boot
Server
PXE Client
Boot

The diagram shows a visual representation of the boot process for chainloading iPXE. (Figure 4.)

## 3 METHOD

Research was initially conducted to understand in a practical sense how PXE works and is implemented as a standard. A comparison was made against other methods of installing an operating system to a computer. The basis of the results aimed to provide conclusions to whether implementing a PXE server in certain working environments may be an effective method for mass operating system deployment to computers.

Data collected from the research was quantitative, with an emphasis on collecting data on the time taken for an operating system to install to a computer, using different methods to form a comparison; initial experiments were made to compare the installation times of an operating system using a standard PXE implementation against using an open-source implementation of PXE, iPXE, using HTTP as a transfer protocol rather than TFTP. These installation methods were then compared against other common methods for installing an operating system such as DVD and USB. Collecting quantitative data on the time taken for an operating system to be installed was used as it was the most suitable measurement for this project due to its accuracy and ease of measuring, which would either support or oppose the hypothesis of the research.

A Windows-based computer with VMware Workstation Pro 16 was used throughout the research to create a set of virtual machines with the same specifications for where the experiments were carried out. A PXE server was created using Ubuntu 20.04, hosting a TFTP and DHCP server. The iPXE server was likewise created using Ubuntu, hosting a TFTP, DHCP and HTTP server. A standard virtual machine was then created with the recommended specifications needed for installation of CentOS 8 of which was installed. Each method of installation was then carried out and repeated a further two times in hopes to increase the accuracy of the collected data. The time taken for the operating system to install was measured from the moment the virtual machine was

initiated to displaying the desktop environment. A macro was used to sync a timer recording the time taken to install the operating system with the initialization of the virtual machine. A screen recording was taken of the installation process to capture the moment of when the operating system had completed its installation and booted to the desktop environment. When reviewing the screen recording, the footage was paused on the first frame to appear on the desktop environment appearing, providing an accurate completion time of the operating system.

# 4 IMPLEMENTATION

A host computer with more than the required specification for the project was located. VMware Workstation Pro was then used on the computer to configure and create three virtual machines of identical specification, two of which would be used to create the PXE and iPXE servers. The remaining virtual machine was used for testing the different installation methods. The virtual machines used a bridge network connection in VMware, allowing the virtual machines to communicate with each. The DHCP server from the host computer was disabled while doing the study to prevent disturbance from the host computer's network. Main specifications for the virtual machine included:

- 4 GB Memory
- 1 Processor with 2 Cores
- 40 GB Hard Disk connected using SCSI
- CD/ DVD connected
- Network Adapter connected and using a Bridged Connection
- USB controller present and using USB 3.1 compatibility

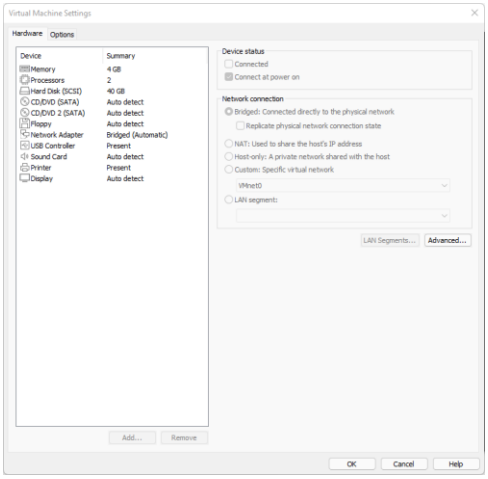


Figure 5.

Prior to installing an operating system on the virtual machine, the preferred boot method was selected in the BIOS for each of the different installation methods. The virtual machine was then restarted with the changes saved and correct media installation method attached to the virtual machine. The time taken for the operating system installation was then measured.

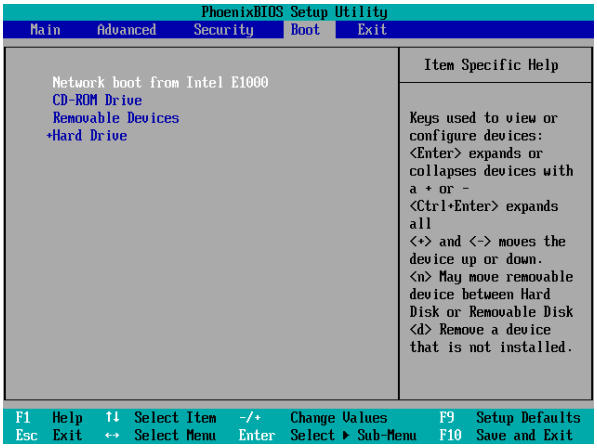



Figure 6.

Data was collected by timing the installation process of the CentOS 8 operating system using different installation methods. A screen recording was recorded using OBS Studio, and the footage was reviewed in Adobe Premiere Pro. The screen recording and initialization of the virtual machine were triggered with a keyboard macro to synchronize the starting point of the footage with the virtual machine being booted. The footage was scrubbed through, identifying the first frame for which the desktop environment appeared. The time of the frame was then recorded giving an approximate time taken to install the operating system.

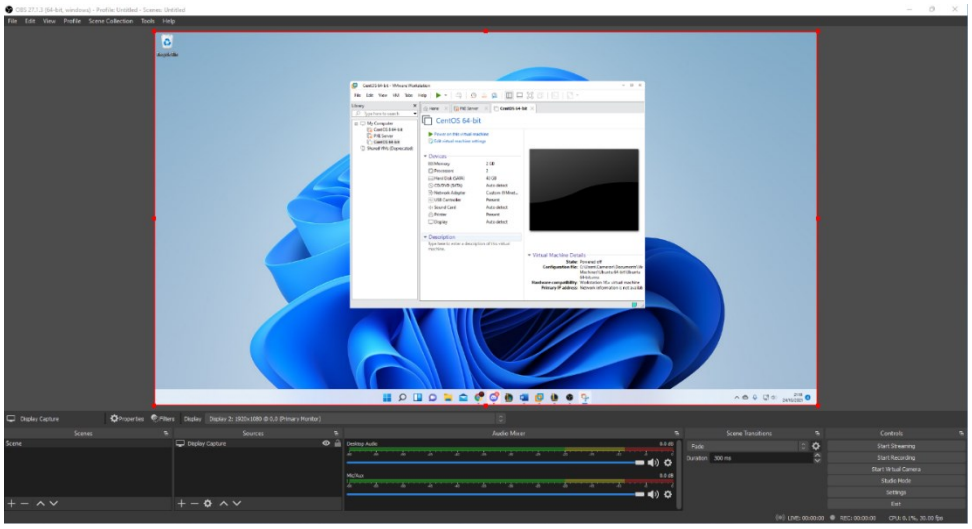


Figure 7.

The PXE server consisted of a DHCP and TFTP server. iPXE also included the addition of a HTTP server within its setup. These packages were installed using the command: `sudo apt install syslinux-common syslinux-efi isc-dhcp-server tftpd-hpa`. (ALSETEMA 2020.)

A directory was then created to store the necessary TFTP server configuration files (ibid.). The following commands were used to create and copy these files:

```
sudo mkdir /tftpboot
sudo mkdir pxelinux.cfg
cd /tftpboot
cd /usr/lib/syslinux/modules/efi64/
sudo cp ldlinux.e64 /tftpboot
sudo cp {libutil.c32,menu.c32} /tftpboot
cd ..
cd SYSLINUX.EFI/
cd efi64/
sudo cp syslinux.efi /tftpboot/
```

The TFTP server settings can be seen in Figure 8. The TFTP directory specified was the root directory of the TFTP server, containing the files previously moved

(ibid.). A firewall rule was created to allow the TFTP transfer over the network using the commands:

```
sudo ufw enable
sudo ufw allow 69/udp
```


Figure 8.

The settings for the DHCP server are shown in Figure 9, which included the subnet and range of available address, lease timing, the address of the DHCP server, router address and the location of the boot file. The DHCP server was also set to authoritative to prefer this DHCP server in the case of another DHCP server being on the network. (ibid.)


Figure 9.

The network interface was set for which the DHCP server would serve DHCP requests (ibid.). This is seen in Figure 10.

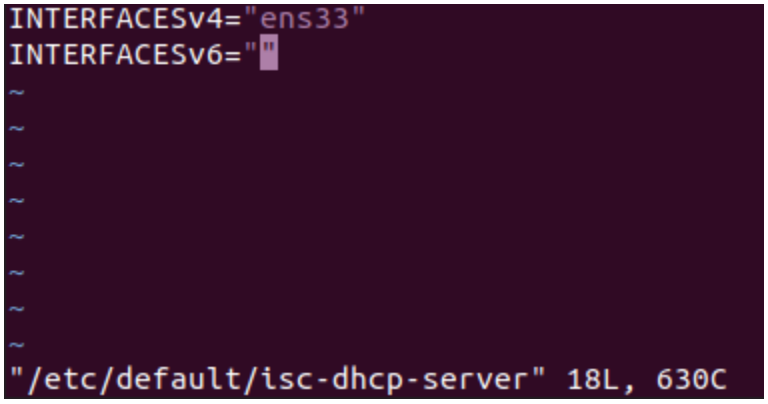



Figure 10.

For the networking booting process to work for CentOS 8 (and other Linux distributions), the PXE server needs to have all the necessary files inside the Linux kernel and the initial RAM disk to be able to boot. A pre-configured CentOS 8 image file was then mounted to the system, and its contents were copied to a new folder within the tftpboot directory (ibid.). This was done using the commands:

```
sudo mkdir /tftpboot/centos
sudo mount CentOS-8.iso /mnt
sudo cp -r /mnt/* /tftpboot/centos/
```

A default PXE configuration file was created. This pointed to the location of the needed files to boot the operating system. This would also include the location of the kickstart file, which would automatically start the installation process for the operating system (ibid.). This is seen in Figure 11.

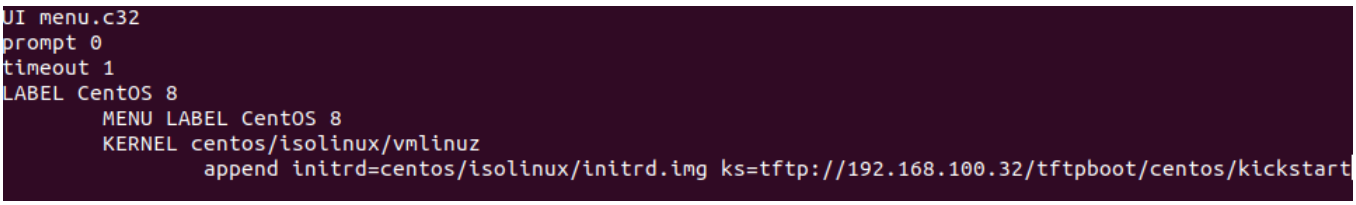



Figure 11.

Figure 12 shows a breakdown of the PXE boot process using Wireshark.

Upon booting from the network, the client virtual machine goes through the DHCP process to join the network. As the client virtual machine has a PXE-enabled network card, it can be seen by the DHCP server in the initial Discover packet. The DHCP server understands from this to offer the location to where the boot files are located on the network. This is shown in Figure 13.

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 55 | 5.483189 | 0.0.0.0 | 255.255.255.255 | DHCP | 389 | DHCP Discover - Transaction ID 0xbe2845ec |
| 56 | 5.483200 | 0.0.0.0 | 255.255.255.255 | DHCP | 389 | DHCP Discover - Transaction ID 0xbe2845ec |
| 86 | 6.483847 | 192.168.100.32 | 255.255.255.255 | DHCP | 347 | DHCP Offer - Transaction ID 0xbe2845ec |
| 87 | 6.483856 | 192.168.100.32 | 255.255.255.255 | DHCP | 347 | DHCP Offer - Transaction ID 0xbe2845ec |
| 110 | 9.475039 | 0.0.0.0 | 255.255.255.255 | DHCP | 401 | DHCP Request - Transaction ID 0xbe2845ec |
| 111 | 9.475050 | 0.0.0.0 | 255.255.255.255 | DHCP | 401 | DHCP Request - Transaction ID 0xbe2845ec |
| 112 | 9.476363 | 192.168.100.32 | 255.255.255.255 | DHCP | 347 | DHCP ACK - Transaction ID 0xbe2845ec |
| 113 | 9.476367 | 192.168.100.32 | 255.255.255.255 | DHCP | 347 | DHCP ACK - Transaction ID 0xbe2845ec |

Figure 12.

```
▾ Dynamic Host Configuration Protocol (Discover)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0xcad289c6
    Seconds elapsed: 0
  › Bootp flags: 0x8000, Broadcast flag (Broadcast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 0.0.0.0
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: VMware_7e:3c:d2 (00:0c:29:7e:3c:d2)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  › Option: (53) DHCP Message Type (Discover)
  › Option: (57) Maximum DHCP Message Size
  › Option: (55) Parameter Request List
  › Option: (97) UUID/GUID-based Client Identifier
  › Option: (94) Client Network Device Interface
  › Option: (93) Client System Architecture
  ▾ Option: (60) Vendor class identifier
      Length: 32
      Vendor class identifier: PXEClient:Arch:00007:UNDI:003016
  › Option: (255) End
```

Figure 13.

The DHCP server then sends an Offer packet to the client virtual machine, giving all the necessary information to join the network, including the IP address that will be used (with subnet mask), the gateway address and domain name server address. With the client device being PXE-enabled, DHCP Option 67 is included in the packet which includes where the boot file name and directory are on the TFTP or HTTP server. This is shown in Figure 14.

```
Dynamic Host Configuration Protocol (Offer)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0xcad289c6
    Seconds elapsed: 0
  > Bootp flags: 0x8000, Broadcast flag (Broadcast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 192.168.100.106
    Next server IP address: 192.168.100.32
    Relay agent IP address: 0.0.0.0
    Client MAC address: VMware_7e:3c:d2 (00:0c:29:7e:3c:d2)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  > Option: (53) DHCP Message Type (Offer)
  > Option: (54) DHCP Server Identifier (192.168.100.32)
  > Option: (51) IP Address Lease Time
  > Option: (1) Subnet Mask (255.255.255.0)
  > Option: (3) Router
  > Option: (6) Domain Name Server
  > Option: (15) Domain Name
  v Option: (67) Bootfile name
      Length: 12
      Bootfile name: syslinux.efi
  > Option: (255) End
```

Figure 14.

The Request and ACK DHCP packets confirm that the client virtual machine will start using the IP address given from the Offer packet. Once this process is done, the client virtual machine now finds the boot file from the location that was provided by the boot server and proceeds to download and load the software, in this case being the installation of the operating system.

The iPXE server setup utilized the chainloading method to obtain the features of iPXE and was specifically chosen to replicate a real-world scenario whereby installing the iPXE rom to a network card may not be an available option. Once the PXE process had completed, the operating system would proceed to install, as seen in Figure 15.

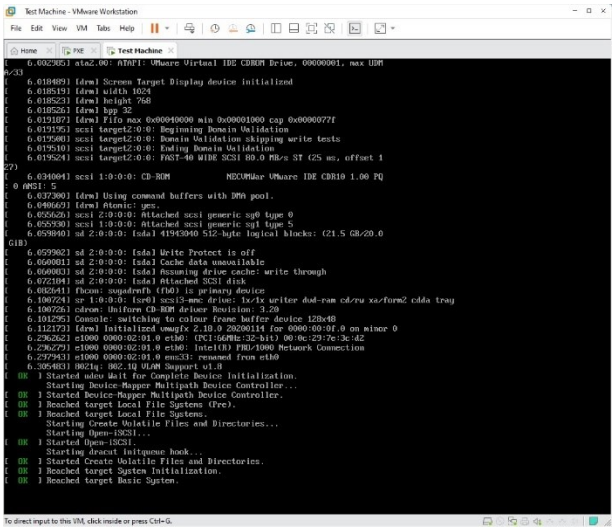


Figure 15.

The installation times were recorded into a table. The results from the study are shown in the table below.

Recorded Time of Operating System Installation (minutes:seconds)

| | OS Install 1 | OS Install 2 | OS Install 3 | OS Install Average |
|---|---|---|---|---|
| PXE | 6:06 | 6:22 | 6:19 | 6:16 |
| iPXE | 5:10 | 5:01 | 5:52 | 5:21 |
| USB | 5:15 | 6:40 | 5:36 | 5:50 |
| DVD | 12:04 | 14:11 | 12:33 | 12:56 |

This concluded the practical research and discussion from the study was then made.

## 5 DISCUSSION

Creating the testing environment and the PXE server itself was a quick process. Using Wireshark helped for a better understand of breaking down each of the components of PXE, seeing each of the protocols used. Documentation for setting up a PXE boot server solution was abundant, though documentation of iPXE was not. One could assume this was because of PXE being a standard used and implemented by manufacturers, whereas iPXE relies on the support of the open-source community.

The results from the initial part of the project showed that on average the iPXE server implementation completed the installations of the operating system quicker than that of PXE. When comparing operating system installation through PXE/ iPXE against a DVD or USB installation, the results showed that PXE and USB were similar in terms of installation speed to that of a DVD which was marginally slower.

A possible explanation for why the DVD was much slower could be the idea of how the operating system is being read from the physical optical disc and the physical limitations of the media. Although being read sequentially, the laser reader needs to keep repositioning itself to the next piece of data once a sector has been read with a theoretical limit read speed of 33.2 MB/s under perfect conditions which in effect could cause read errors leading to a faulty installation of the operating system. Whereas with flash-based storage there are no moving mechanical parts and a much higher read speed. Deeper analysis could have been made with specific types of installation media and their physical limitations such as USB transfer speeds, limitations of network infrastructure (using copper cable against using fiber optic, virtual networking). This could highlight and show a more precise set of data and present any bottlenecks in the implementation, as well as explain the idea of whether the physical hardware or transfer medium becomes the bottleneck when installing an operating system.

Kickstart files were included in the CentOS installation files, allowing for the automatic deployment of the operating system. This method was used to help prevent any anomalies in the results caused by user input. Although the kickstart files allowed the operating system to automatically deploy, the physical process of inserting the removable media could not, which adds to the argument of PXE allowing for fully automated operating system deployment from boot.

Furthermore, the results could be seen to only apply for scenarios created using a similar virtual environment. Different types of hardware and configurations may perform differently under the testing scenario. Creating the set specification for the virtual machines used was put in place to prevent distortion of the research results in the form of researcher bias.

A problem that occurred early in the practical implementation was that a lack of DHCP addresses on a network causes PXE to fundamentally break. If PXE cannot get an available address on a network, it cannot contact the boot server for the files to install the operating system. For the initial case in the

implementation, this would cause the PXE boot process to effectively time-out and default to the second preferred boot option in the BIOS/UEFI.

However, extrapolating from these results it can be assumed that implementing a PXE or iPXE server could reduce installation times when doing mass installations/ deployments of an operating system to computers. The physical cost for removable media needed to do mass deployments of the operating systems along with the time taken to setup each piece of removable media to be ready to install the operating system on a computer, the PXE server could be seen as a more cost- and time-effective solution.

An issue that arose when investigating different hardware types was the availability or usage of a legacy BIOS. Many hardware manufacturers are pushing for the use of UEFI by making it difficult to access the legacy BIOS or providing features exclusively for UEFI. Many businesses may have security policies in place such as using Secure Boot to prevent a computer from being tampered with, which for some operating systems such as Windows 7 is not supported officially, causing compatibility issues when implementing a PXE server.

A future project could investigate an implementation of the PXE server within a containerized environment. Using Kubernetes, this could allow for the potential of a portable implementation while offering high availability for when the server is under a significant amount of load, helping to eliminate bottlenecks relating to transferring files using TFTP or HTTP to multiple clients at once.

## 6 CONCLUSION

The intention of this project was to develop an understanding of the PXE standard, used for network booting operating systems. Conducting a theoretical study beforehand increased understanding of PXE and how the standard was implemented. A practical implementation of a PXE server was deployed using a virtual machine, and comparisons were made with different communication protocols and installation medium methods.

The resulting comparisons had shown that using iPXE in combination with utilizing HTTP had provided quicker results that that of the standard PXE implementation using primarily TFTP for transfer. Comparisons against other installation methods showed that in some cases a USB could be a faster method for installing an operating system. However, it can be extrapolated from these results that installation through network booting (PXE or iPXE) would be a more efficient and cost-effective method for mass installation of an operating system to computers than using USB or DVD. The physical cost and setting up each USB or DVD to be ready to install an operating system would likely be higher than that of a network booting solution like PXE, which could perform the installation procedure concurrently with other computers while only needing a PXE server to communicate with.

When looking at further developments of this project, deeper analysis could be made with specific types of installation media and their physical limitations such as USB transfer speeds and the limitations of using copper cable against using fiber optic. This could enable highlighting and showing a more precise set of data and presenting any bottlenecks in the implementation.

Furthermore, an implementation of the PXE server within a containerized environment would also be a next stage for this project. Using Kubernetes, this could allow for the potential of a portable implementation while offering high availability for when the server is under a significant amount of load, helping to eliminate bottlenecks relating to network traffic congestion while transferring files to multiple clients at once using TFTP or HTTP.

## LIST OF FIGURES AND TABLES

1. Diagram of a high-level overview of PXE.
2. Diagram of the PXE boot process.
3. Diagram of the iPXE boot process with ROM.
4. Diagram of the iPXE boot process using chainloading.
5. Screenshot of the settings used in VMware for each virtual machine.
6. Screenshot of the boot order used in the BIOS.
7. Screenshot of the screen recording method used to collect data.
8. Screenshot of the TFTP server settings.
9. Screenshot of the DHCP server settings.
10. Screenshot of network interface settings used on the DHCP server.
11. Screenshot of the PXE configuration file.
12. Screenshot of PXE boot process in Wireshark
13. Screenshot of DHCP Discover packet in Wireshark
14. Screenshot of DHCP Offer packet with PXE information in Wireshark
15. Screenshot of operating system initializing installation

Table 1. Recorded Time of Operating System Installation (minutes:seconds)

**REFERENCES**

University of Aberdeen. 2011. Trivial file transfer protocol. WWW document. Available at: https://erg.abdn.ac.uk/users/gorry/course/inet-pages/tftp.html [Accessed 1 December 2021]

Setting up an UEFI PXE server on Linux (Part 1). 2020. ALSETEMA. Video clip. Available at: https://www.youtube.com/watch?v=U3RC20ANomk [Accessed 1 December 2021]

UEFI PXE | HTTP and NFS | Live Booting Ubuntu Over LAN (Part 2). 2021. ALSETEMA. Video clip Available at: https://www.youtube.com/watch?v=Sa_7AA9w06o [Accessed 1 December 2021]

CompuPhase. 2019. Extending TFTP. WWW document. Available at: https://www.compuphase.com/tftp.htm [Accessed 1 December 2021]

IBM. 2021. Trivial file transfer protocol. WWW document. Available at: https://www.ibm.com/docs/en/i/7.3?topic=services-trivial-file-transfer-protocol [Accessed 1 December 2021]

IETF. 2009. Rfc5505. WWW document. Available at: https://datatracker.ietf.org/doc/html/rfc5505 [Accessed 1 December 2021]

IETF. 2015. Rfc7440. WWW document. Available at: https://datatracker.ietf.org/doc/html/rfc7440 [Accessed 1 December 2021]

Intel. 2021. Troubleshooting PXE boot with network protocol analyzer. WWW document. Available at: https://www.intel.com/content/www/us/en/support/articles/000006544/network-and-i-o/ethernet-products.html [Accessed 1 December 2021]

iPXE. 2018. Chainloading iPXE. WWW document. Available at: https://ipxe.org/howto/chainloading [Accessed 1 December 2021]

iPXE. 2021. WWW document. Available at: https://ipxe.org/start [Accessed 1 December 2021]

Kansas State University. 2015. 1.11. System Boot – Operating Systems Study Guide. WWW document. Available at: http://faculty.salina.k-state.edu/tim/ossg/Introduction/boot.html [Accessed 1 December 2021]

PXE Boot Provision Explained – Advanced Topics. 2021. RackN & Digital Rebar. Video clip. Available at: https://www.youtube.com/watch?v=QGwIgyYwcFk [Accessed 1 December 2021]

PXE Boot Provisioning Explained – the Basics. 2021. RackN & Digital Rebar. Video clip. Available at: https://www.youtube.com/watch?v=w_ZGlxihlEI [Accessed 1 December 2021]

PXE Explained: How to deploy an operating system. 2019. TechsavvyProductions. Video clip. Available at: https://www.youtube.com/watch?v=5qyTLfJrQM4 [Accessed 1 December 2021]

Traficom. 2019. 100Mb broadband available in nearly 60% of households | Traficom. WWW document. Available at: https://www.traficom.fi/en/news/100mb-broadband-available-nearly-60-households [Accessed 1 December 2021]

University of Wollongong. 2021. Understanding operating systems. WWW document. Available at: https://www.uow.edu.au/student/learning-co-op/technology-and-software/operating-systems/ [Accessed 1 December 2021]