UEFI PXE Boot Performance Analysis

February 2014

Li Ruth, Brian Richardson
Intel Corporation

Executive Summary

Network boot using the Preboot Execution Environment (PXE) is widely supported by current Unified Extensible Firmware Interface (UEFI) implementations. In large network deployments, boot performance is critical. This document analyzes firmware and operating system (OS) elements that impact UEFI network boot performance, and offers suggestions for reducing boot time and removing network bottlenecks. This document focuses on an audience of firmware engineers, BIOS vendors, network engineers and ODM/OEM system engineers.

Contents

1 Using PXE with UEFI

Preboot Execution Environment (PXE) defines a method for booting computers using a network interface, independent of local storage devices or installed operating systems (OSs). On platforms with UEFI firmware, PXE is supported by a network stack in the client firmware. The network's DHCP provides a path to a boot server and network bootstrap program (NBP), downloads it into the computer's local memory using TFTP, verifies the image, and executes the NBP.

- In a Windows Deployment Services (WDS) environment, the NBP is provided by **`wdsmgfw.efi`**.
- In a Linux environment, the NBP is provided by UEFI-enabled boot loaders such as GRUB, GRUB2 or ELILO.

This document focuses on boot performance for the PXE Client in UEFI firmware, and how network topology impacts PXE boot performance. Because the DHCP proxy process contributes minimal overhead to the network boot process, analysis focuses on the OS boot images delivered by TFTP, which can be up to 200 MB of data.

1.1 UEFI Networking

The *UEFI Specification* describes an interface between the OS and platform firmware, and defines a general purpose network stack, including MNP, IP, TCP, UDP, and DHCP Protocols. To support boot from network devices, the firmware makes use of Universal Network Driver Interfaces (UNDIs) and protocols defined in the *UEFI Specification*:

- **`EFI_NETWORK_INTERFACE_IDENTIFIER_PROTOCOL`** (NII)
- **`EFI_SIMPLE_NETWORK_PROTOCOL`** (SNP)
- **`EFI_PXE_BASE_CODE_PROTOCOL`** (PXE)

The PXE protocol is used for network access and booting from a network image.

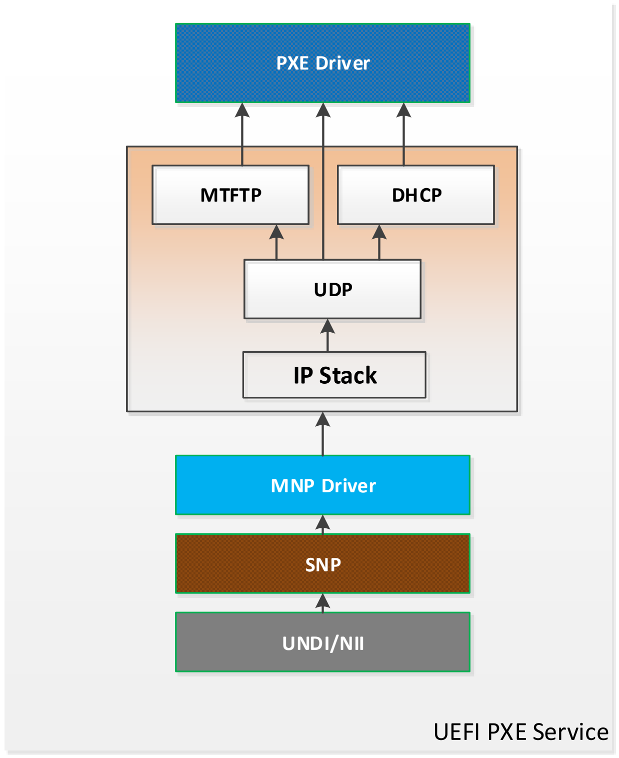

Figure 1: UEFI Network Stack Layer

Figure 1 illustrates the UEFI network layers related to PXE. The lower hardware level (UNDI/NII) is an abstracted interface provided by the network hardware vendor. For Intel platforms, the UNDI layer is provided by a UEFI Device Driver or by a PCI Express Option ROM. The SNP layer wraps around the UNDI interface and provides a packet level interface to the network adapter. The MNP layer provides a raw asynchronous network packet I/O service, allowing multiple drivers and applications to use the system network interfaces at the same time. The middle layer of the stack provides general network services (IP, UDP, DHCP and MTFTP).

The **`EFI_PXE_BASE_CODE_PROTOCOL`** is layered on top of a UEFI Network stack implementation. This protocol also

- Consumes EFI MTFTP4 and EFI MTFTP 6 protocols to provide the TFTP services
- Consumes EFI DHCP4 and EFI DHCP6 protocols to provide DHCP services
- Layers over EFI UDP4 and EFI UDP 6 protocols to provide UDP Read and Write services to the network interface.

1.2 UEFI PXE with Microsoft* WDS*

Microsoft WDS* enables PXE compliant boot environments, starting with Microsoft Windows Server 2003 R2.

Figure 2 is an overview of the Microsoft* WDS* PXE boot process.

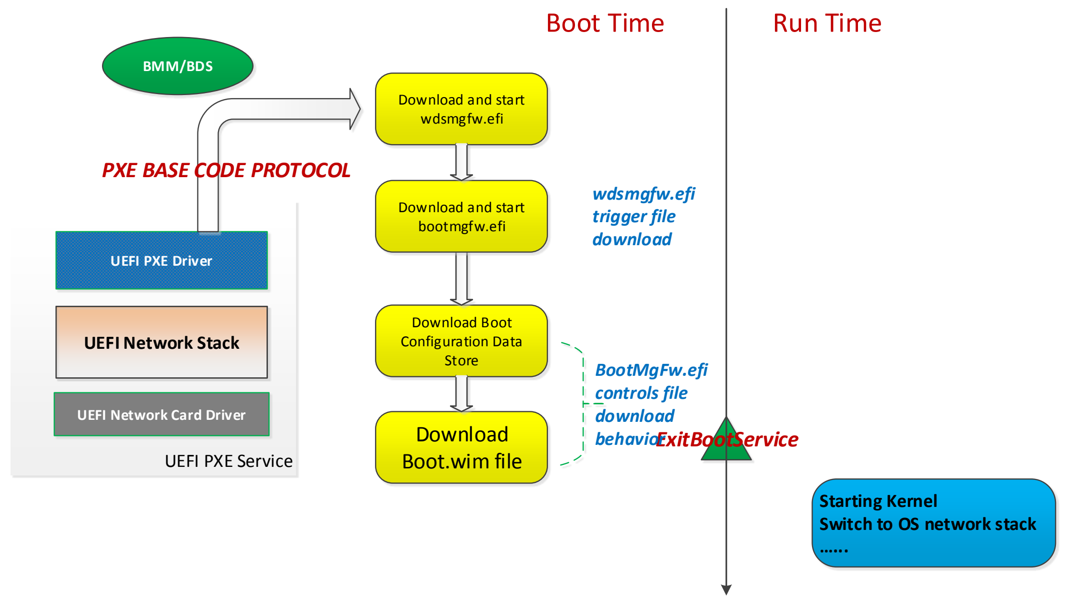
Boot Time
Run Time
BMM/BDS
Download and start wdsmgfw.efi
PXE BASE CODE PROTOCOL
Download and start bootmgfw.efi
wdsmgfw.efi trigger file download
UEFI PXE Driver
UEFI Network Stack
Download Boot Configuration Data Store
BootMgFw.efi controls file download behavior
UEFI Network Card Driver
Download Boot.wim file
ExitBootService
UEFI PXE Service
Starting Kernel
Switch to OS network stack
……

Figure 2: UEFI PXE with Microsoft* WDS*

After System Power on, the UEFI platform firmware initiates the PXE boot process. This requires firmware to enable the UEFI Network Stack and present a network boot option in the Boot Manager Menu (BMM) or Boot Device Selection (BDS) phase. The UEFI PXE Base Code protocol requests a network address via DHCP. Microsoft* WDS* assigns a network address to the client, allowing the client to locate the boot server and request the NBP (`wdsmgfw.efi`).

Once the NBP is downloaded, the client executes `wdsmgfw.efi` as the first stage of the network boot process. The `wdsmgfw.efi` executable trigger file download for boot loader (`bootmgfw.efi`), and boot loader downloads additional files, including boot configuration data store and a RAMDISK image (`boot.wim`). The `boot.wim` file is over 200 MB for a standard client system, so downloading this file consumes most of the PXE boot process.

After these files are downloaded to the client system, the WDS loader calls `ExitBootService()` to stop UEFI Boot Services and transition to Runtime. The Operating System (OS) kernel starts execution and takes over hardware management from UEFI, including network stack operations

1.3 UEFI PXE with Linux

Unlike Microsoft* Windows*, Linux distributions have the option to support different boot loaders with PXE support. The boot process is similar to the Microsoft* WDS* implementation. Boot loaders used by various Linux distributions include ELILO, GRUB or GRUB2. The next section describes using ELILO for UEFI PXE boot with Linux.

1.3.1 ELILO

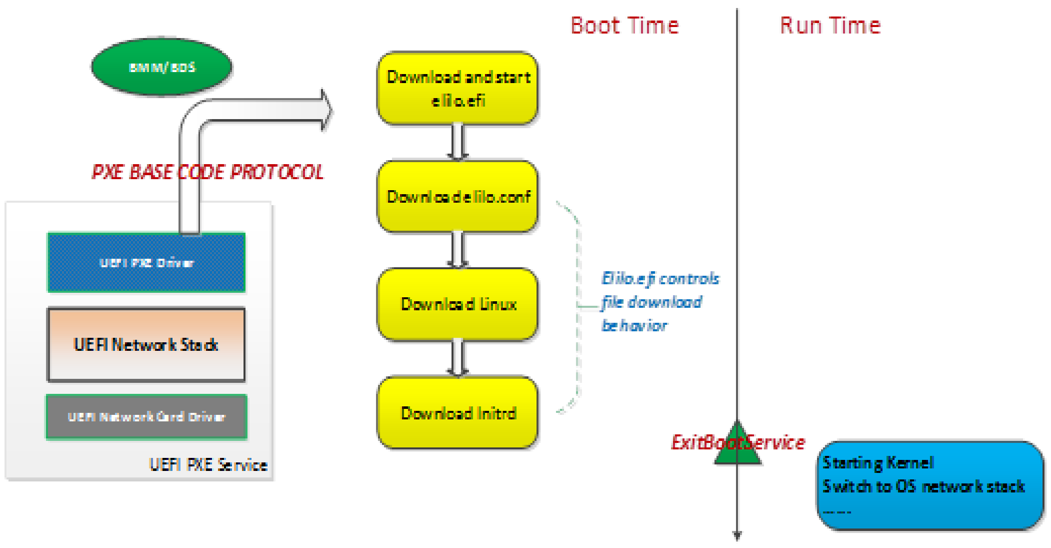

Figure 3 UEFI PXE with Linux PXE Service

Figure 3 shows an overview of the ELILO PXE boot process using UEFI. It's very similar to the Microsoft* WDS* procedure, but with the Linux boot loader (`elilo.efi`), the configuration file (`elilo.conf`), and the Linux kernel image and initial ramdisk taking the place of the downloaded files. The kernel and ramdisk sizes are variable, depending on configuration. Configuring SUSE SLE11 SP3 with a boot image for the Intel Desktop Board DQ77MK UEFI 2.3.1 Development Kit used a 4 MB kernel and 34 MB initial ramdisk.

2 Factors Affecting UEFI PXE Performance

Performance for UEFI PXE deployments is measured using client boot times. This document focuses on two aspects of PXE boot performance:

- Eliminating network infrastructure problems
- Tuning network boot performance

PXE relies heavily on TFTP for file downloads, and large files can cause delays if the network is not configured properly. Tuning the TFTP configuration may have a significant impact on network boot performance. This paper also describes a methodology for measuring boot performance, and how to configure an appropriate test environment.

2.1 Methods for Measuring Network Boot Performance

Measuring boot performance for UEFI PXE is similar to measuring boot performance from a local disk. There are two boot time measurements to examine:

- **Time1**: Time between system reset vector and `ReadyToBoot` event (platform initialization and boot environment setup)
- **Time2**: Time between `ReadyToBoot` and `ExitBootService` event (hand-off to OS from the UEFI Boot Loader)

Systems compliant with the *Advanced Configuration and Power Interface (ACPI) Specification Revision 5.0* can use the Firmware Performance Data Table (FPDT) to record this data. ACPI FPDT data is refreshed on every system reset, so it is important to collect this data prior to system reset or shutdown.

Network download performance impacts the Time2 measurement. The platform firmware uses the UEFI Network Stack for downloading boot files, based on the TFTP protocol. The Microsoft* WDS* process relies more on the UEFI Network Stack. Because it downloads larger files via TFTP, it is the primary example for our network performance analysis. The default configuration for TFTP stream depth is dynamically adjusted, but using `BcdEdit` you can adjust the TFTP stream depth for better network performance.

The largest file downloaded is `boot.wim`, so the download time for this file is used as an indicator for network performance. Network sniffer applications provide more accurate time measurements.

2.2 Network Configuration

Note that network boot performance data is only meaningful in a specific network. .configuration, one with the following variables:

Network bandwidth: Today's Intel client platforms include wired Gigabit Ethernet Controllers (1000 Mbps). The best network performance requires cabling and switch infrastructure supporting Gigabit Ethernet (GbE).

Network switch: The performance of the network switch itself has an impact on boot performance. Switches used for boot performance testing should be rated for GbE bandwidth. Certification by an interoperability agency is highly recommended (example: UNH-IOL testing for IPv6 and Gigabit Ethernet).

Server configuration: Even though this document focuses primarily on firmware performance, the boot server configuration can severely impact PXE boot performance. The TFTP protocol requires interaction between the client and server, and often the client must wait for a server response. The server's transfer rate can be improved through optimizing TFTP parameters and sizing hardware requirements to properly serve network clients.

PXE client configuration: The client's system hardware configuration can affect boot performance, but the platform BIOS also has a significant impact. Client systems should use the latest UEFI UNDI driver for the Ethernet adapter and platform firmware with an updated UEFI Network Stack.

Network background traffic: Naturally, a crowded network environment will decrease the overall network boot performance. Excluding the impact of extra traffic for optimal PXE boot performance requires a clean network environment.

3 Tuning Network Boot Performance

This section provides guidance to BIOS Vendors and OEM and ODM system designers to optimize PXE boot performance.

BIOS Customization: Eliminate Unused Network Modules

The UEFI Network Stack provided by EDK II uses a modular design. Refer to the *UEFI Specification* (sections 2.6.2 & 2.6.3) for information on protocols required for different scenarios.

Driver Profile for PXE Boot via IPv4: PXE boot over IPv4 requires the following modules for a complete network stack:

- SNP
- MNP
- ARP
- IP4
- IP4CONFIG
- UDP4
- DHCP4
- MTFTP4
- PXE

These modules are provided by EDK II MdeModulePkg. This also requires the proper UNDI driver for the networking hardware, which is provided by the adapter's PCIe Option ROM or UEFI Driver. If you are using an Intel Ethernet Adapter, the latest drivers can be found at intel.com.

Driver Profile for PXE Boot via IPv4 and IPv6: If both IPv4 and IPv6 are required, the following IPv6 network stack modules are required from the EDK II NetworkPkg:

- IP6
- UDP6
- DHCP6
- MTFTP6.

Also required is the PXE driver from `NetworkPkg` with support for both IPv4 and IPv6 (`NetworkPkg/UefiPxeBcDxe`), but the older PXE driver in `MdeModulePkg` must be removed.

3.1 Boot Policy: Eliminate Network Impact on System Boot Performance

A well-designed Boot Device Selection (BDS) policy can also improve PXE boot performance:

- Implement "fast boot" to optimize the boot path. Do not connect unnecessary I/O devices in BDS.
- If the client boots first from PXE, put the network device's boot option first in the UEFI boot order. This avoids BDS connections to other boot options. If there are multiple network device boot options, place the preferred device first to avoid connecting and configuring other network devices.
- Network devices may publish two UEFI boot entries per Ethernet port: one for IPv4 and one for IPv6. Put the correct entry at the top of the UEFI boot order, based on the network topology.

3.2 Code Revision: Use Updated UEFI Drivers

The entire UEFI PXE stack contains the Ethernet adapter UNDI driver and firmware UEFI Network Stack implementation. Updating these components can address known performance issues from older revisions:

- Download the latest device drivers from intel.com when using Intel Ethernet Adapters.
- The EDK II project has an open source UEFI Network Stack implementation using components from the `NetworkPkg` and `MdeModulePkg` packages.

Intel offers UEFI Development Kit platforms for reference, based on Intel motherboards running the latest UEFI platform firmware. These platforms are used for UEFI testing and development, and contain updated versions of the Intel UNDI driver and open-source UEFI Network Stack. More information can be found at the Intel UEFI Community Resource Center.

3.3 BIOS Porting: Performance of System Services

The UEFI Network Stack relies on several firmware services, which may slow boot performance if ported incorrectly:

- **GetTime:** The `GetTime()` runtime service is triggered to add a timestamp to every TFTP DATA packet. This call interacts with the runtime clock (RTC), and, if not optimized, may drastically slow TFTP download speeds.
- **GetVariable** and **SetVariable:** The *UEFI Specification* requires network drivers to provide NVRAM variables for configuration information (see Section 24 and Section 26 for reference). This can trigger multiple calls to the `GetVariable()` and `SetVariable()` services during PXE boot. Most variable service implementations use System Management Mode (SMM) for security purposes, which can add overhead if not properly optimized. Verify the SMRAM cache is enabled (write-back mode) for optimal performance.

4 Performance Data from Sample Platform

This chapter details the performance data gathered from a sample environment configured using the Intel® Desktop Board DQ77MK UEFI 2.3.1 Development Kit. This is an isolated test network, using one UEFI PXE Client and one Microsoft WDS* server connected via a 1 GbE switch.

- **PXE Client:** Intel® Desktop Board DQ77MK UEFI 2.3.1 Development Kit (BIOS revision SDV.MK.B3), Intel® Core™ i5-3550 Processor, 16 GB DDR3 RAM.
- Connected to the network via an on-board GbE adapter
- **Microsoft* WDS Server*:** Microsoft* Windows* 2012, Intel® Pentium® D Processor 920, 3 GB DDR RAM.
- Connected to the network via an Intel PRO/1000 PM GbE adapter.
- **Network Switch:** HP Procurve Switch 6108 (GbE)

4.1 TFTP Performance with Microsoft* WDS*

Table 1 shows network download performance data using TFTP with Microsoft* WDS*. The time to download `boot.wim` file is recorded, counted from the point the client sends a TFTP_READ_REQUEST for the file to the client sending TFTP ACK, which confirms receiving the last block (averaged across three PXE boot sessions).

The SDV.MK.B3 BIOS was tested with both IPv4 and IPv6. The `boot.wim` file is used because it is the largest TFTP payload delivered to the UEFI PXE client. In our test case, the `boot.wim` file size was 201 MB.

Table 1: Time measurements for downloading boot.wim

PXE Boot Test	IWdsTransportTftpClient WindowSize	IWdsTransportTftpClient BlockSize	Average Time (sec)
IPv4 (default)	Dynamic adjusted	1456	23.7053
IPv6 (default)	Dynamic adjusted	1456	29.0567
IPv4 (optimized)	50	1456	9.5382
IPv6 (optimized)	50	1448	9.3385

In our test case, Windows 2012 WDS started with the following parameter values:

- `ramdisktftpblocksize = 1456`
- `ramdisktftpwindowsize = 4`
- `ramdisktftpvarwindow = Yes`

The `WindowsSize` and `BlockSize` parameters are part of the Boot Configuration Data (BCD) store on the Microsoft* WDS* server. The test was run using default values and optimized values to demonstrate how server configuration impacts the TFTP transfer rate. When using the default setting for dynamic configuration of WindowSize, the value started at 4 and finally increased to 64. We captured ~4K TFTP ACK packets during the process of downloading boot.wim file. The change in WindowSize value was controlled by interaction between the loader and WDS server.

Note that the `BlockSize` value for IPv6 is smaller than for IPv4, to avoid extra IP packet fragmentation.

Please refer to MSDN for more information on TFTP parameters.

4.2 Impact of GetTime() and SetVariable()

The performance cost for `GetTime()` and `SetVariable()` service can be measured by making multiple interface calls to these services and calculating the average number. The measurement is made from the UEFI Shell using a simple UEFI application. The application calls the `GetTime()` and `SetVariable()` functions 1,000,000 times, records the start and end time, then calculates and reports the average cost for each function.

Using the Intel® Desktop Board DQ77MK UEFI 2.3.1 Development Kit (BIOS revision SDV.MK.B3), the cost for GetTime() is 0.0367 microseconds and the cost to set a Non-NV Variable (1024 bytes) through SetVariable() is 0.04 microseconds). Using the same test method on the Intel® Server Board S2600CP4 UEFI 2.3.1 Development Kit (BIOS revision SDV.CP.B4), the cost for GetTime() is 0.0365 microseconds and SetVariable() is 0.168 microseconds.

4.3 UEFI PXE Boot Performance

Table 2 shows the overall UEFI boot performance using the ACPI FPDT method.

Table 2: PXE boot performance time

Performance Data	PXE boot over IPv4 (seconds)	Normal boot (seconds)
Reset End	0.021	0.021
OS Loader LoadImage Start	7.037	5.321
OS Loader StartImage Start	20.246	5.512
ExitBootServices Entry	40.363	8.861
ExitBootServices Exit	40.365	8.862

This table reports time for five checkpoints in the boot process:

- Reset End
- OS Loader LoadImage Start
- OS Loader StartImage Start
- ExitBootServices Entry
- ExitBootServices Exit

In our test case the PXE Boot process requires a system reset, which destroys current ACPI FPDT data. This makes it impossible to test using a UEFI Shell Application or an OS tool. To retrieve the FPDT data before it is lost on reset, we replaced the FPDT module in the RELEASE image with a DEBUG version, including a module to output FPDT data via serial console. The boot performance data was collected from the serial log information.

The following example shows how to add the FPDT DXE driver to the platform's DSC file:

```
MdeModulePkg/Universal/Acpi/FirmwarePerformanceDataTableDxe/FirmwarePerf
ormanceDxe.inf {
<LibraryClasses>

DebugLib|MdePkg/Library/BaseDebugLibSerialPort/BaseDebugLibSerialPort.in
f
}
```

The time between 'Reset End' and 'OS Loader LoadImage Start' covers platform initialization and boot environment setup (**Time1**, in section 2.1).. The time from 'OS Loader LoadImage Start' to 'ExitBootServices Exit' (**Time2**, in section 2.1) covers execution of the DHCP/PXE service and downloading multiple NBP images. Note the large time differences between the PXE and normal boot (local drive) scenarios.

For a more detailed description of the FPDT fields see the *ACPI Specification Revision 5.0a* ('Firmware Basic Boot Performance Data Record').

5 Summary

This document focused on PXE Boot performance and provides design guidance on how to obtain good PXE Boot performance for BIOS developers. To facilitate this topic, we described the general process of PXE boot, and described the method of measuring network performance. We listed possible components which impacts the performance data and provided a benchmark performance data using one reference platform and the suggested method.

Acknowledgement

Sincere thanks to Fu Siyuan, Zhang Chao and Wu Jiaxin for help setting up test environments, working through test issues and collecting data for this document.

Reference Documents

Document	**Location**
UEFI 2.3.1 Specification	http://www.uefi.org/specs
Microsoft* Windows Deployment Services	http://msdn.microsoft.com/en-us/library/windows/desktop/dd379586(v=vs.85).aspx
SUSE Linux Enterprise Server 11 SP2 for UEFI Clients Best Practices (White Paper)	http://www.novell.com/site/docrep/2012/12/SUSE_Linux_Enterprise_Server_11_SP2_for_UEFI_Clients_Best_Practices
Advanced Configuration and Power Interface (ACPI) Specification	http://www.acpi.info
Preboot Execution Environment (PXE) Specification	ftp://download.intel.com/ial/wfm/pxespec.pdf
[RFC2131] Dynamic Host Configuration Protocol, IETF, 1997	http://www.ietf.org/rfc/ rfc2131.txt
[RFC2132] DHCP Options and BOOTP Vendor Extensions, IETF, 1997	http://www.ietf.org/ rfc/rfc2132.txt
[RFC 3315] Dynamic Host Configuration Protocol for IPv6 (DHCPv6), July, 2003	http://www.ietf.org/rfc/rfc3315.txt
Trivial File Transfer Protocol – TFTP	http://www.ietf.org/rfc/rfc1350.txt http://www.ietf.org/rfc/rfc2347.txt http://www.ietf.org/rfc/rfc2348.txt http://www.ietf.org/ rfc/rfc2349.txt

Legal Disclaimer