Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 8 (9): 1726-1731, 2008
ISSN 1812-5654

A Study on Direct Memory Loading of an Operating System Through Preboot Execution in a Homogenous Computing Environment

[1]S.N. Remesh and [2]A. Abid
[1]Intel Technology Sdn. Bhd, Bayan Lepas Free Trade Zone, 11900 Penang, Malaysia
[2]Faculty of Information Technology, Multimedia University,
Jalan Multimedia, 63100 Cyberjaya, Malaysia

Abstract: In this study, the remote boot process of an operating system was looked into to determine how an optimized solution using the Preboot Execution Environment (PXE) process could be created. This study focuses on Windows CE remote boot implementation in an embedded homogeneous computing environment. Instead of a DOS shell environment and a bootloader, this research demonstrates the use of PXE pre-OS environment to introduce the OS bootloading process, which allows a linear address space Windows CE boot image to be downloaded directly to the remote client machine system memory by using PXE. In addition, the use of optimized PXE pre-OS stage to perform the typical bootloading process and the use of the PXE OS boot stage to download the linear address space Windows CE boot image is shown. The result is a quicker remote boot solution customized for Windows CE which illustrated how PXE and an operating system could be built to produce a quicker remote boot boot-up experience.

Key words: Operating system, preboot execution environment, remote boot

INTRODUCTION

Remote boot is an alternative solution to booting system. It enables manageability for wide and dispersed computing environment, while providing a lower total cost of ownership. An operating system's remote boot solution is typically different for the many available OS. There are many parameters which govern the way that the OS remote boot is developed including the boot process, bootloader, OS image file format, file system, file I/O access performance and OS foot print. In typical remote boot solutions, there are also extra steps that are taken to boot the OS such as acquiring the IP address, booting a shell environment, loading a network stack, setting up a shared drive, setting up a ramdisk on the remote client machine and executing the bootloader process. As these additional steps enable the remote boot environment requirements, they also increase the boot up time of the OS. In a remote boot solution and in term of performance, the boot up speed is often looked at as the major factor in a user's experience, where these bootloading environment requirements are causing a slower performance.

The OS bootloader is an important part of the OS start-up process and its functionality is similar in the various OS. The main purpose of the bootloader is to initialize the platform and its environment to prepare the OS to boot; including the CPU and memory initialization, loading the OS to the memory, parsing boot parameter and booting the OS image on the target device. The bootloader execution requires the use of a pre-OS execution environment or shell environment to perform this process. In remote boot solutions, boot images are downloaded to the target machine's ramdisk in order to setup a shell environment to execute the bootloader.

In a diskless remote boot solution, there is no local boot storage media and therefore, the OS boot image is downloaded over the network to the target machine system memory or ramdisk. To enable this, a network stack would have to be loaded first, to download the boot image to the system memory. With the shell environment and also the network stack, access to a server or a shared drive could be established. The bootloader can then load the OS to the diskless remote target machine's system memory and subsequently boot the OS (Stückelberg and Clerc, 1999).

Both the shell environment and the loading of the network stack in a remote boot solution consume time, which clearly affect the boot up performance. In this research, the problem of the bootloader's extra phases is addressed, in a remote boot environment as bootloading is an important step in all operating systems. This study was motivated because the current bootloading requirement in remote boot solution consists of too many steps. As a result, a simplified approach is required to

Corresponding Author: S.N. Remesh, Intel Technology Sdn. Bhd, Bayan Lepas Free Trade Zone, 11900 Penang, Malaysia

download the OS image to the target machine. This research introduces the integration of the bootloader with the remote boot building block, that is, the Preboot Execution Environment (Intel Corporation, 1999).

PREBOOT EXECUTION ENVIRONMENT

With the introduction of Wired for Management (Intel Corporation, 1998) and also BIOS Boot Specification (Intel Corporation *et al.*, 1996), which allows the Network Interface Card (NIC) to be part of the selectable boot media, they also enabled an exciting world of remote manageability. This specification permits the NIC to be used as a boot device; where the NIC executes the network option ROM codes. In this network boot stage, the PXE provides the ability to initialize the NIC, loads a set of network stack and runs a Dynamic Host Configuration Protocol (DHCP) client process to acquire an IP address from a DHCP server. Coupled with this, the PXE also provides the capability of downloading the OS images to remote boot target machine through the use of a PXE Server, typically located on the DHCP server. The PXE introduces the use of the DHCP options (Johnston and Venaas, 2006) to differentiate DHCP discover packets of the OS DHCP client process and the PXE client process in the network boot ROM; for the PXE Server to respond on top of the DHCP Server. The DHCP Server together with the PXE Server, through the use of the PXE identifier in the DHCP options sent by the PXE client, provides the PXE client with the IP address, boot server information and the index of the boot menu (Intel Corporation, 1999). After the selection of the boot menu, the PXE client discovers the boot server, receives the TFTP/MTFTP information and then further downloads the network bootstrap program. This network bootstrap program execution will then start the OS bootloader and remote boot the operating system on the remote client machine.

PXE is an industry standard server client solution which is widely deployed for various managed solutions including remote installation (Microsoft Corporation, 2007), disaster recovery (3COM Corporation, 2000), virus scanning (emBoot Corporation, 2000), PC BIOS updates (Argon Technology, 2001) and also remote booting. The PXE process (Intel Corporation, 1999) is shown in Fig. 1. The PXE process follows these specific RFCs, RFC 4578 (Johnston and Venaas, 2006), RFC 2131 (Droms, 1997) and RFC 2132 (Alexander and Droms, 1997).

BOOTLOADING ENVIRONMENT

In the PXE boot process, the pre-OS environment downloads the network bootstrap program to the remote client ramdisk; at address 0×7C00h. The pre-OS environment then re-hooks the floppy drive's interrupt service routine to the remote client ramdisk which subsequently enables address 0×7C00h to boot the shell environment. Depending on the type of network bootstrap program boot image, MS-DOS or Linux shell, the boot image executes the start-up process of the shell environment. The network bootstrap program is able to run a script, create a network stack, establish an access to a shared drive, create a larger ramdisk and also runs a bootloader to remote boot an OS.

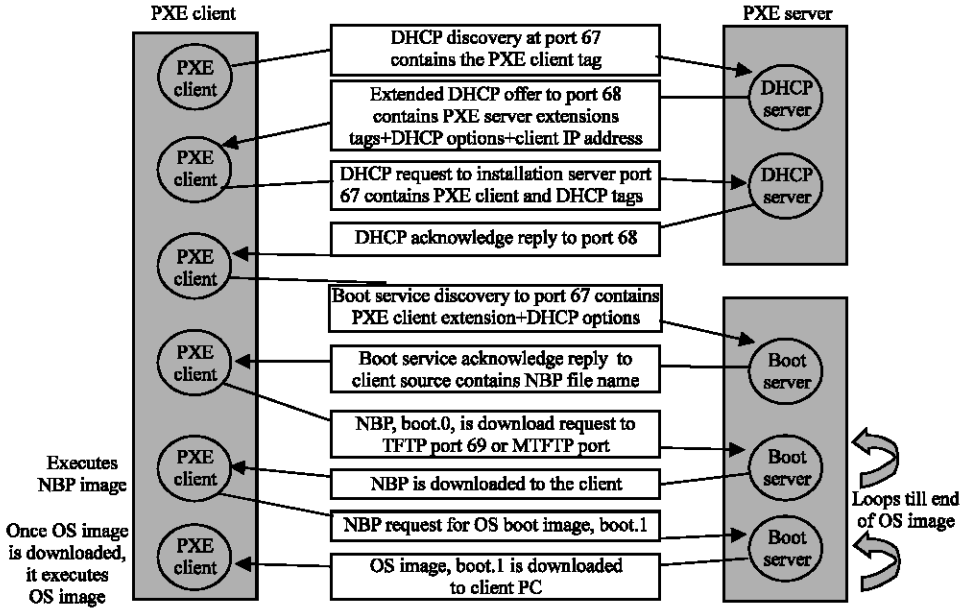

Fig. 1: PXE boot process

In an example of a remote boot solution to remote boot Windows CE, the network bootstrap program remote boots a DOS shell. Upon the start-up of the DOS shell, the autoexec.bat executes and loads a TCP stack. The TCP stack is then used to access a Windows CE NK.BIN boot image through a network shared drive. Next, the Windows CE loadcepc bootloader starts loading the Windows CE boot image to the remote client system memory before jumping to the start address location of Windows CE and booting the OS. In this solution, the network bootstrap program is used to load the Window CE boot image data to the system memory on the remote client machine. The use of the DOS shell environment and TCP stack allows the Windows CE bootloader to run and perform the task of loading the data to the system memory.

THE BOOTLOADER REQUIREMENTS

To load the OS files to the system memory and start the OS boot process, as a final goal, the different remote boot phases are analyzed, in this research, to reduce the number of steps in the remote boot process. The analysis looks into the pre-OS environment, DOS shell and also the OS boot requirement to understand the final output before starting the OS boot process. The method of accomplishing this has to be generic. It also has to be able to support a large boot image, configure the bootable address location and allow boot parameters to be passed; and initialize the hardware and have access to a large addressable memory range.

Memory access and size: The bootloader must have access to a large addressable memory range to download a large OS image to it. With a large memory access, we can boot a large operating system and not constrain the OS. Furthermore, it enables a minimized/custom OS to fit a fixed memory size.

Configurable address location: When downloading the OS boot image, the starting point of the address location has to be able to be configured according to the needs of the OS boot image.

Passing custom parameters: Allow boot parameters, which are specific to the platform, to be passed to the boot image. Some boot images require specific parameters to be passed in order to enable display resolution, display comments and RS232 debugging among others.

Reduce boot image processing: If a boot image requires additional processing, for instance decompressing the boot image, the boot time is then longer. With a linear OS boot image, we are able to download the OS image directly to the target client machine's memory. If an OS image is created with the OS kernel; device drivers, memory mapping configuration and other files are in the correct memory location, the boot process is then quicker and easier.

Jump to start address: After the OS image is downloaded to the memory, the execution must jump to the starting address location of the start function or process. This begins the process of booting the OS. Configurability of this jump address provides flexibility in the bootloader.

PXE BOOTLOADER

An approach of reducing the number of steps is by incorporating the bootloading process into PXE boot process. In other words, this is done by incorporating the objective of the DOS shell and bootloader process into the PXE boot process. Nevertheless, since PXE is a standard process; when incorporating the bootloader process into PXE, we have to ensure that we maintain backward compatibility to the existing PXE process.

The PXE process has a few important phases that should be maintained for the compatibility. The DHCP process is a standard process that acquires the IP address from the DHCP server. This process of acquiring the boot menu provides the selectivity of boot image. Discovery of the boot server allows the target client machine to obtain the information of the boot server on the network domain.

Without impacting any PXE implementation, any customization can be performed at the pre-OS phase which is the boot.0. During the boot.0 execution phase, we have access to the whole addressable memory region. In addition, the PXE network API stack is available, which gives the capability to download the subsequent boot image, i.e., boot.1. Furthermore, the boot.0 environment allows some execution where boot parameters can be passed and finally jumping to the start address location of the boot image.

The PXE bootloading implementation is evaluated using Windows CE. Windows CE is an embedded OS used by many commercial appliances. Windows CE supports many different boot media including remote boot. There are different methods of remote booting using Windows CE, ranging from shared drive to a server, ramdisk and also eboot. Windows CE is used in this analysis because it has a customizable boot image tool capability of memory configuration and it is capable of booting from compressed or linear ROM boot image, selectable of many different applications. In addition, its

bootloader is well-documented (Microsoft Corporation, 2006a, b) and its bootloading process is similar to many operating systems (Plagge, 2004). Windows CE OS memory layout is configured using config.bib, where the kernel, RAM memory location and other components selected in Platform Builder, should be located. Windows CE uses DOS loadcepc bootloader to boot a Windows CE boot image, NK.BIN. Platform Builder is the Windows CE utility used among others to build the boot image, customize desktop image and integrate device drivers.

INTEGRATING BOOTLOADING INTO PXE

To remain compatible to PXE, all of the bootloading steps must be integrated within the pre-OS boot.0 environment. Below are the taken steps:

Loading boot image directly to system memory: In the boot.0 execution, it contains the fixed address location to download the boot.1; the linear Windows CE ROM boot image. The fixed download address of boot.1 must be synchronized with the config.bib memory offset in Platform Builder. The TFTP/MTFTP will download the boot.1 linear ROM image to that fixed address location.

Linear boot image: The boot.1 boot image is a linear ROM image and not the compressed NK.BIN Windows CE image. By using the linear ROM image, the boot.1 file can be downloaded directly to the system memory instead of using loadcepc bootloader in the DOS shell environment; to decompress the NK.BIN image to the system memory.

Utilizing PXE network stack: The PXE MTFTP/TFTP network API is used to download the boot.1 linear ROM image from the PXE boot server to the client's system memory. This alternative is used instead of loading a DOS TCP network stack to get access to the shared network drive.

Figure 2 shows the boot.0 flow using the PXE bootloader process. The PXE bootloader solution demonstrates that it is able to provide a quicker remote boot boot-up experience on Windows CE. This improved performance is due to several factors. The removal of the need:

- For a shell environment, hence skipping an additional step
- To load another network stack to download the boot.1 boot image
- To access a shared network drive location to get the boot image
- To decompress the boot.1 image as it is a linear ROM image

Together with the improvement in performance, this solution also allows the configurability of the address location to download the boot image. In addition, it provides a pre-OS environment to run any bootloader execution, e.g., hardware initialization and passing boot parameters.

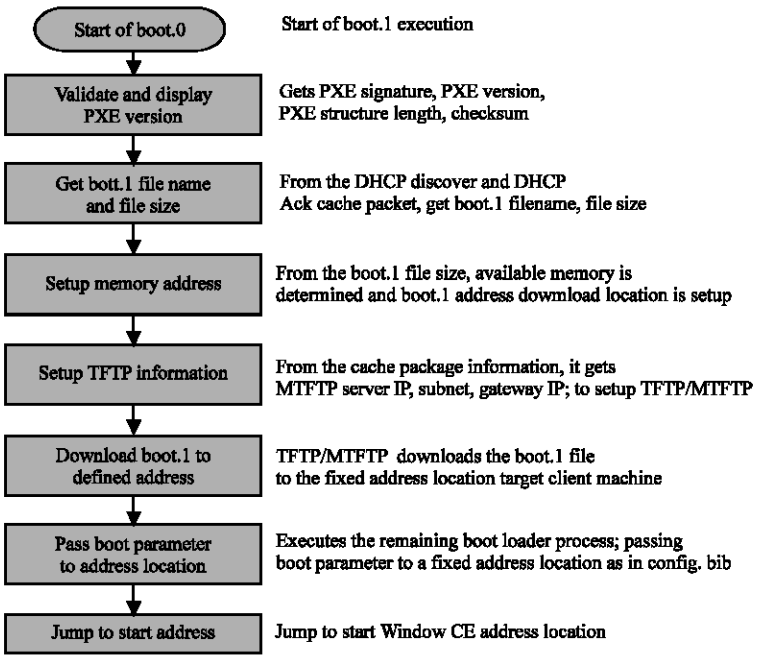

Fig. 2: Flow diagram of the bootloading integrated in the PXE boot.0 execution

Table 1: Bench marking results comparing PXE bootloader

Target client machine boot phase (from the power-up time)	Network shared drive (sec)	PXE Bootloader (sec)
Start-up to completion of PXE DHCP process	13.0	13.0
To the display of the Boot Server boot list	14.0	14.0
To start of Boot.0	25.0	25.0
To the complete download of Boot.1	25.5	29.0
To start of DOS shell environment	26.2	NA
To start of loading TCP network stack	26.6	NA
To start of loadcepc to load Windows CE image	35.0	NA
To the jump to Windows CE start boot address	40.5	31.2
Windows CE booted	42.3	33.5

BENCHMARKING

For benchmarking, the PXE bootloader solution is compared with the Windows CE remote boot. Here the boot-up performance of a DOS shared drive Windows CE loadcepc solution and the performance of PXE bootloader solution are compared. The same Window CE boot image size of 12.4 MB for an x86 target machine is used. For the PXE bootloader, the nk.bin image is converted to a linear memory boot image size of 12.6 MB; a slight increase of the image size due to address padding.

In Table 1, the Network shared drive column shows the time for a solution where Windows CE is remote booted through a network shared drive environment from a server. The shared drive access is created through the use of the DOS shell using the TCP NDIS stack and loadcepc bootloader.

To the complete download of Boot.1 (row 4) the time (29.0 sec) is longer for PXE Bootloader due to the PXE process of downloading the nk.nb0 (linear memory Windows CE boot image) file which is 12.6 MB in size. On the other hand, for the shared drive solution, the nk.bin (12.4 MB) is downloaded in benchmark while its boot.1 is only 1.4 MB.

From the benchmark results, it is found that the PXE bootloader solution reduces the boot- up time by about 9 seconds compared to a Windows CE shared drive remote boot solution. This is about 20% boot-up performance improvement compared to a shared network drive remote boot solution. The new PXE bootloader takes about 4 sec to download the Boot.1 (Windows CE linear boot image) before jumping to the start address of the Window CE. During the boot-up, the BIOS took 13 sec while the PXE DHCP process took 12 sec.

CONCLUSION

This PXE Bootloader solution was able to demonstrate a faster boot up time for Windows CE. This was largely because it managed to reduce the number of steps in the remote boot process and it optimized the boot image type.

The focus of PXE was to create an industry standard and generic solution for remote manageability and all customization left for solution providers. As a standard, PXE is needed to demonstrate compatibility, but not customization across a wide application. Such bootloader integration solutions can be done in the boot.0 pre-OS environment. In remote boot solution, especially in homogeneous embedded application, the PXE bootloader solution could be introduced to improve the boot performance which is an important user experience factor.

Future improvement would include adding the capability of another DHCP or PXE option to be able to pass the memory address where the boot image is required to be downloaded. This would avoid the need to fix the address location and therefore, provide more flexibility. Secondly, not many OS are capable of producing linear memory boot image. If a tool is available to create boot images for various operating systems, this PXE bootloader solution can be used by many different operating systems. To further reduce the platform boot-up time, a BIOS vendor might be able to integrate and optimize the PXE integration into the BIOS however it might have to compromise the standard PXE implementation; which might be possible in custom embedded solutions using a homogeneous environment.

REFERENCES

Alexander, S. and R. Droms, 1997. DHCP Options and BOOTP Vendor Extensions, IETF RFC 2132.

Argon Technology, 2001. Using RIS Menu Editor to Perform a Remote BIOS Update.

3COM Corporation, 2000. Boot services and disaster recovery. Technical Paper Disaster Recovery.

Droms, R., 1997. Dynamic Host Configuration Protocol, IETF RFC 2131.

emBoot Corporation, 2000. Network Boot Tool and Virus Scanning.

Intel Corporation, Compaq and Phoenix, 1996. BIOS Boot Specification (BBS) Specification Version 1.01.

Intel Corporation, 1998. Intel Wire for Management Specification Version 2.0.

Intel Corporation, 1999. Preboot Execution Environment (PXE) Specification Version 2.1.

Johnston, M. and S. Venaas, 2006. Dynamic Host Configuration Protocol (DHCP) Options for the Intel Preboot eXecution Environment (PXE), IETF RFC 4578.

Microsoft Corporation, 2006a. How to Develop a Boot Loader for Windows CE: MSDN: Windows Embedded Developer Center.

Microsoft Corporation, 2006b. Platform Builder for Microsoft Windows CE 5.0: Boot Loader Design. MSDN: Windows Embedded Developer Center.

Microsoft Corporation, 2007. Remote Operating System Installation.

Plagge, M., 2004. Microsoft Windows CE 5.0 Board Support Package, Boot Loader and Kernel Startup Sequence; Microsoft Corporation.

Stückelberg, M.V. and D. Clerc, 1999. Linux Remote-Boot Mini-HOWTO.