University of Nevada, Reno

An Approach for Enhanced Management of Network-Attached Devices

A thesis submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Computer Science

By,
Michael J. McMahon Jr.
Dr. Sergiu M. Dascalu / Thesis Advisor

May, 2007

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

University of Nevada, Reno
Statewide • Worldwide

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

MICHAEL J. MCMAHON JR.

Entitled

An Approach for Enhanced Management of Network-Attached Devices

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Dr. Sergiu M. Dascalu, Advisor

Frederick C Harris Jr

Dr. Frederick C. Harris, Jr., Committee Member

Dr. Rahul Bhargava, Graduate School Representative

Marsha H. Read

Marsha H. Read, Ph. D., Associate Dean, Graduate School

May, 2007

Abstract

The recent proliferation of inexpensive, high-quality network-attached devices has made them extremely popular within organizations because of their efficient, simplified nature. In order to fully realize the advantages of these network-attached devices, however, certain management shortcomings must be addressed. The development of a management system designed specifically for the optimal incorporation of network-attached devices on a network, under their native network-attached paradigm, stands to improve network and system administration performance significantly.

This thesis analyzes the potential benefits of network-attached devices, researching the development of an administrative tool to make migration toward a network-attached device paradigm possible. Using this tool, the practical benefits of utilizing network-attached devices under their native paradigm versus a network-capable paradigm are determined. The features provided by this new management tool, in tandem with future developments, stand to improve network performance, device utilization, resource management, and user support.

Acknowledgments

I wish to thank my advisor, Dr. Sergiu Dascalu, for his support, knowledge, guidance, and timeless willingness to attempt to defeat me in tennis. Without him, this research and a good deal of my knowledge about the software process would have been tenuous, at best. Dr. Fred Harris, my other advisor, has provided guidance and thorough instruction over the years; though his courses were difficult, they were most instructive. Also, Dr. Rahul Bhargava, my final advisor, for his support on my committee.

I would also like to acknowledge Professor Nancy LaTourrette for her encouragement and assistance in courses, as she provided me with a critical, beneficial introduction to programming and computer science. My instructors for this semester were also crucial to the development of this thesis. Without the willingness of Dr. Aaron Covington and John Kenyon to give me latitude in my assignments, I would not have had the opportunity to develop the thesis to this level.

Finally, I with to thank my family for their support, especially their understanding when I sequestered myself for several weeks to complete this thesis. And of course, my girlfriend Tawnya for her support, warmth, and understanding of the occasional long night doing homework.

Table of Contents

List of Tables

List of Figures

Note: All figures in this document are original creations by the author using public-domain images provided by [1] and its contributors. Further, the figures in this document are available for public-domain use.

1 Introduction

Network-attached devices have become increasingly common as both high- and mid-range office supplies. These devices, in contrast to others that require a server to manage the sharing of their resources, are capable of autonomous management of incoming data and use requests. Despite the proliferation of these devices, they have not been deployed as network-attached devices, but rather as network-capable devices.

The reason for this deployment choice is a lack of management and client installation infrastructure for these devices. This forces administrators to configure the devices as network-capable devices, dependent on a server, which requires additional resources. This need not be the case; network-attached devices may be deployed across a network using the existing management tools, leveraging the capabilities of the device with the centralized management needs of administrators. The purpose of this thesis is to demonstrate a method by which administrators can retain such management capabilities while minimizing office resource and network bandwidth use by allowing the device to operate in its native mode in a Windows 2000/2003 network.

Traditionally, administrators have utilized logon scripts to achieve management tasks that were otherwise out of their direct reach. Most scripts, however, are resource-intensive to execute on systems or pose security risks. Therefore, an alternative that uses the built-in, efficient operating system functions has been developed.

Some existing products have attempted to offer the capability of network device management via Group Policy Extensions. The most notable of these products are PolicyMaker Standard Edition [2], IntelliPolicy [3], and Microsoft Print Management [4]. Currently, PolicyMaker offers support for the addition of TCP/IP-based printers as well as printer and drive mapping; IntelliPolicy offers support only for printer and drive mapping; Microsoft Print Management offers support only for printer mapping. The first two of these products offer an expanded set of policies that include these features as a subset – they do not focus on network-attached device management.

The key advantage of the prototype produced for this thesis (called the Network Device Management Group Policy Extension) is the ability to manage the client installation of all network-accessible printers, as well as the associated printer drivers. Further, the extension allows for the future addition of other network-attached devices, such as scanners and fax machines. This allows administrators to extend and upgrade their networks with minimal effort and as they need. Finally, the Network Device Management Group Policy Extension distributes this functionality as open source software, encouraging administrators to extend the framework and share it with others.

The extension developed for this thesis was designed with system administration in mind – it is fully deployable on Windows 2000 and 2003 Server networks as an MSI. This allows administrators to install the extension and manage it from within Group Policy on a server. Currently, the extension supports the installation of network-connected printers, network-attached printers and their drivers, and network storage devices as drive mount points. Using this tool, the benefits of deploying and managing network-attached devices under their native paradigm were researched in terms of deploying previously network-capable printers as network-attached printers. The resulting network performance and management improvements improvements make the paradigm extremely appealing.

Despite these features, the extension still has several development goals that have not yet been implemented. For example, closer integration between drivers and devices to improve performance will reduce client execution time and network bandwidth use significantly. The reimplementation of the extension as a C# extension that utilizes WMI system information is an ideal development, which would make the extension suitable for the management of all types of network-attached (and possible local) devices. The most important developments are, however, the support of additional network-attached devices, such as scanners, which would easily make this extension unarguably unique as a system administration tool.

The remaining chapters in this thesis provide background on the network-attached and network-capable devices, with special focus on the management of network printing devices. This is followed by the software design models used in the creation of the Network Device Management Group Policy Extension that the thesis used to both investigate and illustrate the advantages of centralized management of network-attached devices (in this case, IP-based printers are used to illustrate). The closing sections compare the results and features to those of commercially-available products, noting the future work that the extension will undergo. The thesis ends with a summary of the results, feasibility, and benefits of the implementation of network management tools around a network-attached device paradigm.

2 Network-Attached Devices

The term “network-attached device” can, and is, applied to a great number electronic devices. Strictly speaking, devices bearing this moniker need meet no requirement greater than a connection (via physical, electromagnetic, or other means) to an interconnected set of other electronic devices. It is therefore necessary to clarify the term in order to convey the precise meaning within the context of this thesis.

2.1 Definitions

In order to clarify the subtle differences between devices utilized on a network, the following subsections describe the classification scheme used throughout this thesis. Despite the subtlety of the differences between the devices, these quirks become quite significant in terms of the abilities of the devices and their management.

2.1.1 Network-Attached Devices

A network-attached device, in relation to modern computer networks, is an electronic entity that provides a necessary function either to an end-user or to another entity on the network. The network mentioned here is generally a mesh of hardware and interconnecting media that allows entities at the endpoints to communicate quickly and efficiently, with minimal (optimally no) errors. The hardware and media that constitute the network are referred to as the "network fabric." That said, it is important to note that network-attached devices are not those that constitute the network fabric – that is, they do not take an active role in the management of the network information. Rather, they participate in the utilization of the network fabric, providing services to other entities that send and receive information over the network fabric.

It is therefore reasonable to describe network-attached devices as entities that have a defined presence on a network, provide services to other entities, and consume/produce information on the network. Typically, the "presence" of a device on a network is marked by the assignment of a unique identification value (e.g. IP or MAC address) that distinguishes the device (and thus the capabilities it provides) from other entities on the network. By this definition, a network-attached device is no different from a computer attached to the same network. However, the distinguishing mark of a device that bears the designation "network-attached" is that it provides a specific set of defined functions, typically with a high degree of efficiency. This means that the device cannot simply be reprogrammed to perform another task, as a computer can; it provides well-defined functionality to other entities on the network and is able to perform only those functions.

One final point worth mentioning is that the scale of a network-attached device is relevant as well. The previous elucidation applies equally to the internal workings of a computer as it does to a computer itself, while present on a network. A transistor connected to other transistors via a set of wires is just as valid a network-attached device as a computer on a corporate network. It is necessary to clarify that, within the scope of this thesis, network-attached devices are macro-scale electronic devices. These entities are not particularly portable, though they can be moved, and they are typically measured most appropriately on a scale of kilograms for mass and decimeters or greater for dimension.

2.1.2 Network-Attached vs. Network-Capable

A network-attached device, within the context of this thesis, is different from a network-capable device. Though both are attached directly to a network in order that their resources be shared, they differ in one key manner: network-capable devices are dependent on a server to manage their data, whereas network-attached devices manage their own data. That is, the dependency on a server is the key difference. An illustration of this difference is given in Figure 1.

A network-capable device cannot support multiple use requests from different sources – they require a server to manage the data flow and resource management. Network-attached devices, on the other hand, are capable of managing all use requests and data transmissions autonomously. This is a key difference that has several important

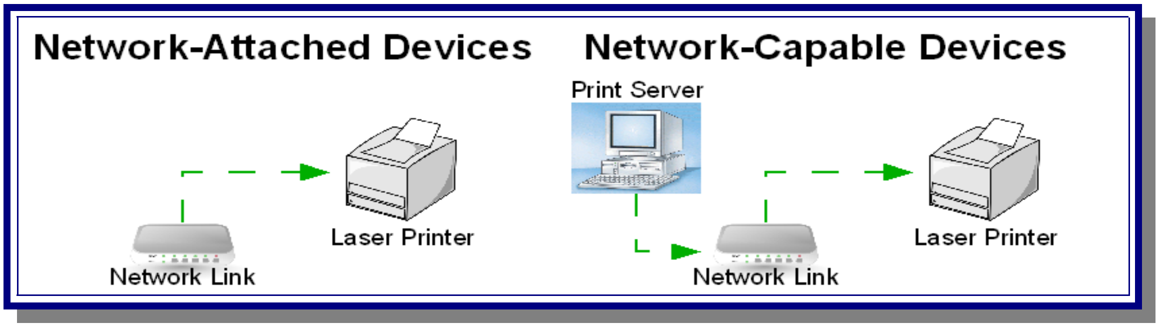


Figure 1: Illustration of the difference between network-capable and network-attached devices

consequences, as shall be discussed later.

2.2 History

The earliest network-attached devices were, technically, the devices that formed the networks – the routers, switches, cables, etc. that constitute the network fabric. When these devices were first constructed, their purpose was to allow communication between the first truly network-attached devices: computers. As computers grew in popularity, so did the availability of devices that were directly attached to them. This allowed a single computer to perform a variety of tasks using devices attached to them that were designed to perform a single, well-specified task.

The growth of computer networks to even a moderate size made clear the need to share resources directly attached to computers. Organizations with multiple employees found it expensive to supply each individual with the devices they required to do their jobs, and inconvenient to require employees to share resources in a manner akin to time-domain multiplexing of company resources attached to computers. For example, an office had the option of supplying each of its employees that produced reports with a printer, or requiring a group of employees to share access to a single computer with a printer attached. A graphical representation of this situation is given in Figure 2.

As neither of these solutions proved to be particularly efficient or cost-effective to enterprises, computers were endowed with the ability to share their resources with other computers connected to them. This allowed, for example, a single computer with a

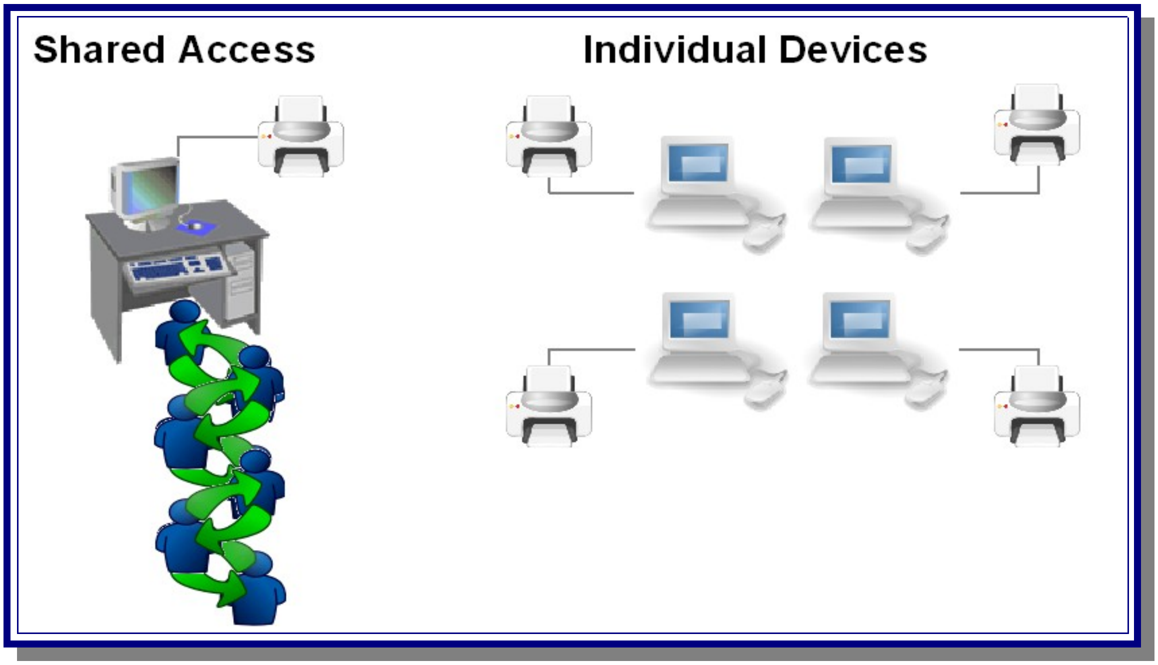


Figure 2: Depiction of organizational resource distribution options before network-attached devices

specialized device attached to allow other computers on the network to communicate with that device, without interrupting the user on the computer where the device was physically attached. This also allowed groups of computers to utilize one specialized device in a significantly more efficient and productive manner, without the need to provide duplicate devices to users. Resources have been shared in this manner for some time; however, the ability of the hosting computer to adequately manage the use of the attached device while serving the needs of a user declines quickly as the number of devices requiring access to the device increases.

Continued increase in the sizes of networks drove the development of servers – computers reserved specifically for the hosting and managing of access to specialized devices that must be made available to a number of other computers on the network. These servers were not directly utilized by users, but rather were left next to the device which they were managing. A representation of this management scheme is shown in Figure 3. In this graphic, the green lines depict the flow of device access information from client entities to the dedicated servers; red lines indicate the flow of information to the devices. In this case, the information must pass through the dedicated server.

This solution is quite effective in resolving performance decreases for meso- and macro-scale organizations. The drawback to this approach is that access to the device is dependent on a separate, dedicated computer system. This means that the specialized, shared resource is dependent not only on the functionality of its internal workings, but also on those of the server. Should the server fail or need replacement, the device is

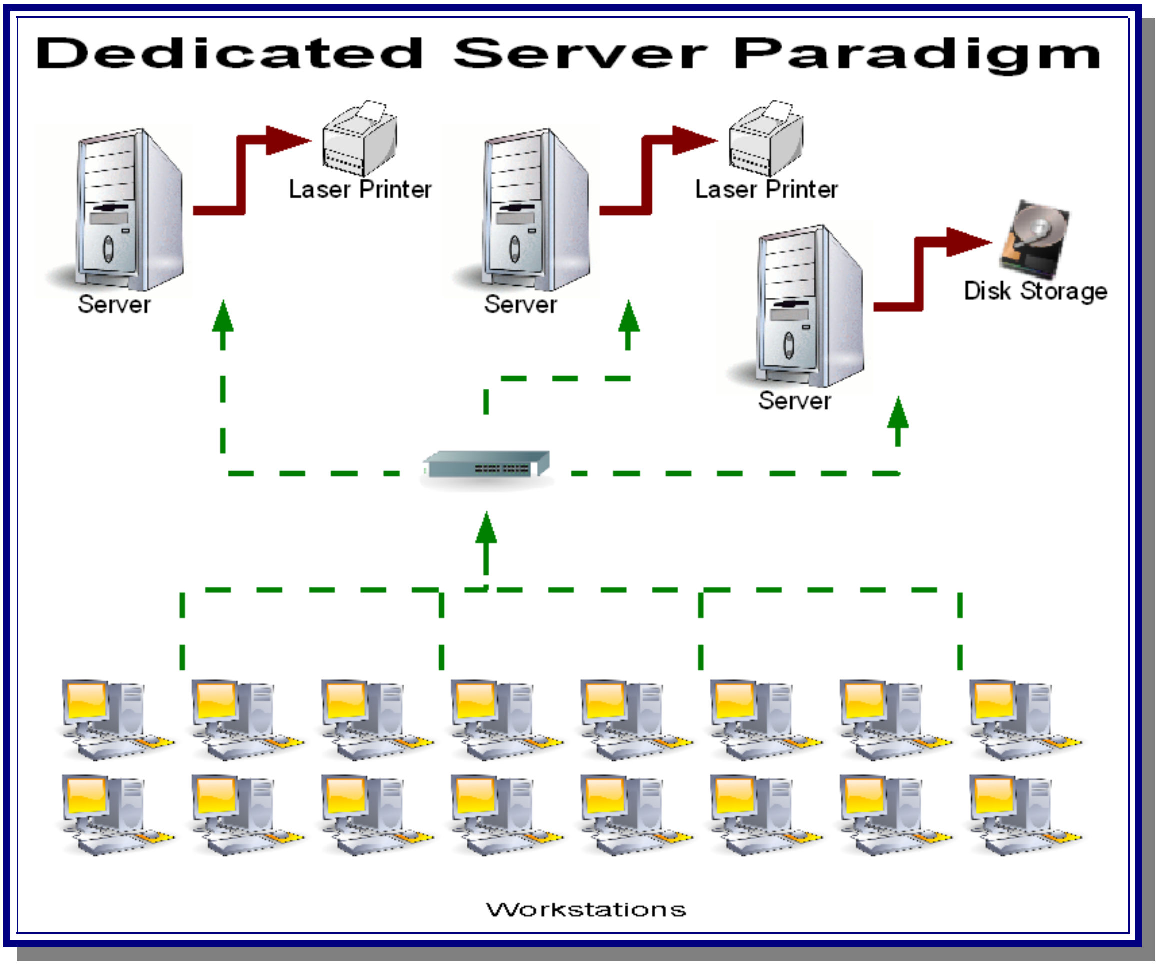


Figure 3: Representation of the dedicated server management paradigm

unusable during that period of time. Further, changes to the server often required that network computers utilizing the resource be reconfigured to access the device on the modified or replaced server. Again, while these modifications are trivial on smaller networks, they become expensive and time-consuming for larger organizations – in essence, inefficient to support.

The advent of network-capable devices was an intermediary step toward eliminating the dependence of devices upon dedicated servers. Network-capable devices are physically connected directly to a network, but require a configured server to manage the data they receive. This paradigm is represented in Figure 4. Here, the green lines indicate the flow of information from clients that is destined for the devices that the server controls. The red lines indicate the path of information from the server to the

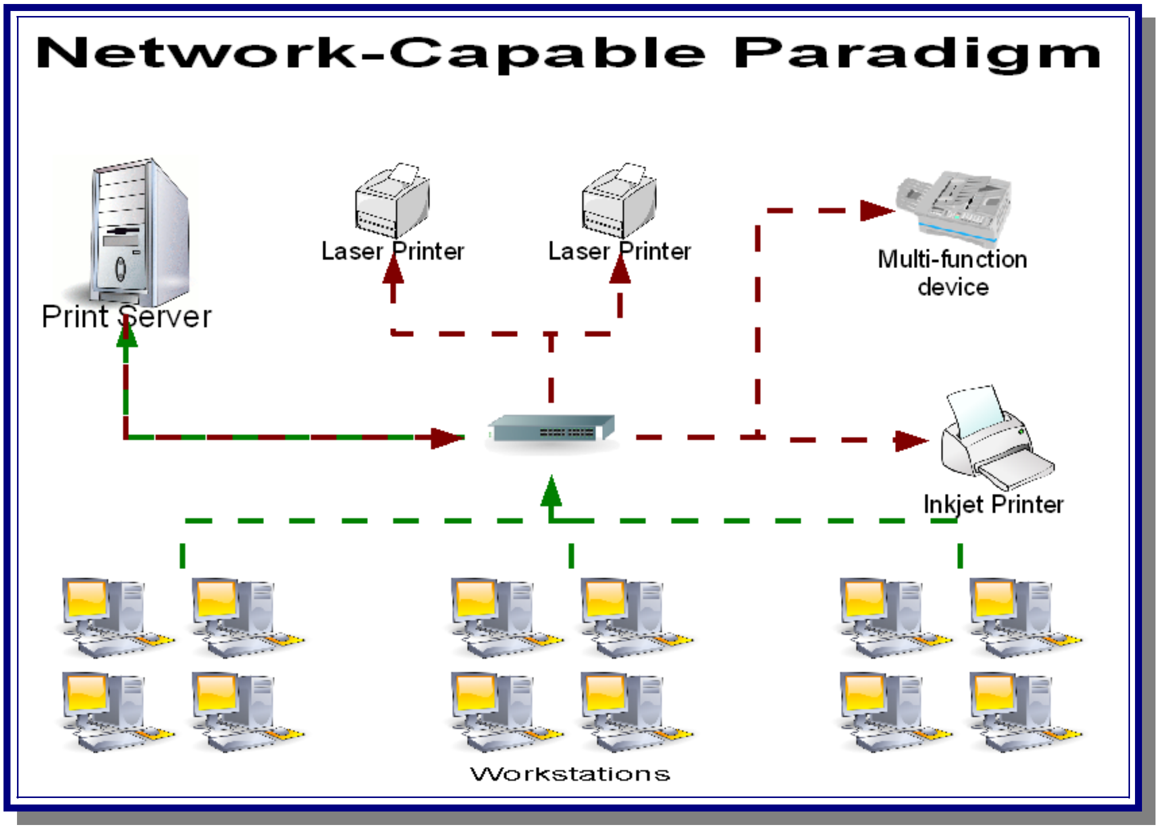


Figure 4: A representation of the network-capable device access paradigm

network-capable devices. Note how information from clients passes through the network fabric more than once before reaching the device for which it is destined.

The advantage of this development was that the device server did not necessarily have to be located next to the device, but could reside on any segment of the network. Further, it was then much easier to use a single server to manage multiple network-capable devices. An example of such a use would be the creation of a dedicated print server, which manages print data for all printers in an organization. Since a single server controls the data for multiple devices, the overhead and support for the devices is reduced to a single (or a small number of) machines.

It is at this point that engineers began to eliminate the device dependency upon a dedicated server for management and sharing of device features. By integrating network interfaces, as well as specialized microprocessors into the devices, entities that previously required a server for access could directly communicate with the device itself. These devices are those commonly referred to as “network-attached devices,” as they interface with other entities directly on a network, without an intermediary or server, as shown in Figure 5.

Here, again, the green lines (were they present) would represent device requests sent to the servers; red lines, as expected, indicate the flow of information to network-attached devices. Note that the information flows directly to the device through the network, and that the types of devices vary from multifunction devices to network-attached storage (NAS). This management paradigm has several benefits:

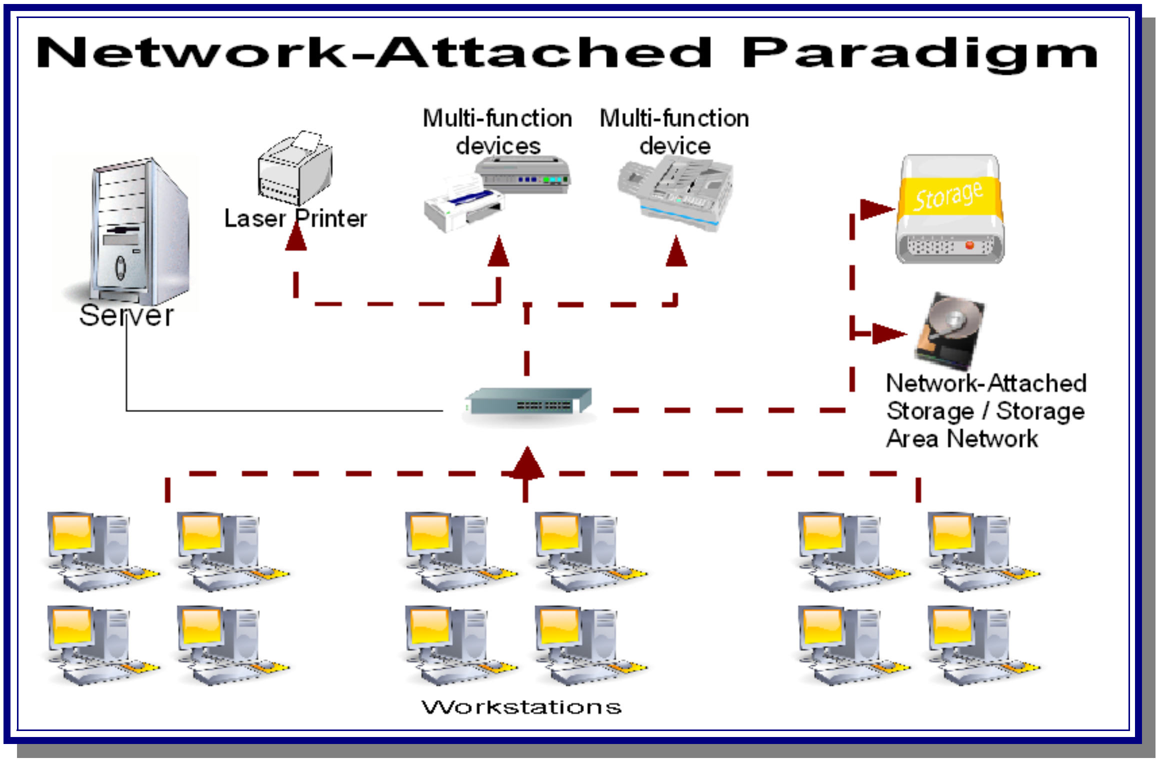


Figure 5: A representation of the network-attached device access paradigm

- *Network traffic is decreased*. By transmitting information destined for a device directly to the device, intermediary management transmissions are eliminated. A network-capable device relies on a server for management, requiring network bandwidth. For extremely large or intensive jobs, the bandwidth required may be quite large. This means that data is transferred over the network to the (possibly remote) server, then to the device. By sending data directly to the network-attached device, the retransmissions from the server to the device are eliminated.
- *Management is simplified*. In many cases, these new devices allow remote management and status updates.
- *Devices perform efficiently*. Since the device is aware of the features it possesses, formatting and other processing functions can be optimized for the hardware of

the device. This results in faster request processing by the device, as well as faster completion times for a task.

- *Hardware dependency is minimized*. Devices require only two things: power and a network connection. All management and interface requirements are internal to the device, reducing the probability of failure, and increasing productivity of end-users.
- *Standardization is enforced*. Because the devices access their host networks in well-defined ways, standards for access to the device must follow interface standards for the networks they utilize. Further, in order to create very flexible products, manufacturers developed standards for each specialized device to ensure consistent interoperability. This had the effect of creating uniformity in the way these devices were accessed.
- *Platform independence*. Network-attached devices are designed to accept data using standard protocols. As such, the operating system of the client machines is irrelevant, so long as they send the information using the protocol that the device requires. These devices can then be utilized by heterogeneous networks, consisting of entities with many different operating systems. Because they require no translation hardware or support, these devices are extremely well-suited to environments supporting multiple operating systems that do not wish to purchase devices for each platform.

2.3 Device Classes

The development of fully independent network-attached devices has been driven by the need to share resources of very specific types. Though more devices become network-accessible with each passing year, the majority of network-attached devices fall into one or more of the following categories:

- *Output*. These devices include printers, plotters, etching machines, routers (used for manufacturing), and other devices that produce tangible objects. These devices allow multiple entities on a network to send information for output in any of a variety of ways. These devices are optimized to provide features such as document management, buffering, statistics, and other useful abilities via network communication in an efficient manner. Many HP, Lexmark, and other enterprise-level printers are designed to operate directly as a part of the network.
- *Storage*. Network-attached storage (NAS) devices have become a popular [5] and convenient way for network entities to share storage [6]. These devices, though they can be costly in terms of network bandwidth, can be combined to form high-performance, high-efficiency storage area networks (SANs) [7]. As the need for organizations to store information continues to increase, so can the devices they employ to meet those needs. Network-attached storage devices are highly-efficient banks for storage devices that allow fast, redundant access to data, often ensuring that hardware failures do not interrupt data access or result in data loss. The performance of these systems is typically significantly better than that of a

dedicated server with physically-attached storage. Sun Microsystems, Buffalo Tech., and other manufacturers currently produce storage systems optimized for capacity, performance, and safety.

- *Input*. Input devices are relatively new as network-attached devices. These devices allow other entities to request images or other information regarding media that have been provided. At present, a few scanners and cameras fit into this category. While digital cameras typically access network resources via a wireless link, scanners often exist as components of multi-function network-attached devices. For example, the Brother MFC-xxxCN line of multi-function devices [8] is designed specifically to provide the features of all these categories to all entities on a network.

These categories represent a generalization of the prominent network-attached devices within an organization. Most devices shared between users within an organization will easily meet the criteria of one category, but many emerging devices will meet those of both.

Multifunction network-attached devices are cost-effective, compact, efficient combinations of multiple devices which are accessible directly from a network [9]. Just as a fax machine is, in essence, a combination of a scanner and a printer and, thus, meets the criteria of both categories, multifunction devices combine input and output devices into a central unit.

A generic example is shown in Figure 6, which depicts a device that combines a printer, copier, and fax machine. Another example is the Brother MFC-440CN network-

Figure 6: A generic example of a multifunction network-attached device

attached device [8], which combines an ink jet printer, scanner, fax machine, and copier into a small, efficient hardware package that provides full network access to all its functions. Such devices are inexpensive and ideal for small- and medium-sized organizations, which accounts for the increased appearance of these devices in the market.

Dedicated network-attached devices are those devices that provide a single function on a network, but that have been designed for optimal performance. Laser printers are an example of such devices, as they are designed to maximize paper output speed while preserving print quality. An example of such a network-attached device is the HP LaserJet [10] with a built-in network module. This device can be attached directly to a network, allowing users to access the high-performance device quickly and easily. The relatively high cost of these devices makes their use sparse in moderately-sized organizations, but standard in larger ones.

2.4 Life Cycle

Because network-attached devices are present and accessible directly on a network, the administration of these devices has been simplified. Using standard interfaces and protocols, network-attached devices benefit from simplified installation, configuration, and maintenance procedures. Although these benefits are notable, the installation of network-attached devices is not fully developed, suffering from several restrictions that hinder their full utilization. The development of software mechanisms by which this hurdle can be overcome is important to the development of network-attached devices and their efficient integration into networks. The ultimate benefit of such integration is contained within the general goal of system administrators: to keep the system running as efficiently as possible so as to maximize the productivity of users [11]. Optimizing network traffic and minimizing installation and update time is commensurate with this goal, and certainly attainable, once certain deficiencies have been overcome. In order to understand where the deficiencies lie, it is necessary to understand the typical processes involved in the life cycle of network-attached devices.

2.4.1 Installation

The installation of any device within an organization requires more than a simple physical connection. For a device to be "installed," it must be both ready to receive data and readily available for use by other entities on the network. This means that simply plugging a new device into a network port is insufficient – clients must be able to use the device without additional configuration on the part of the end users.

For this reason, the installation of a new network-attached device can be described in these steps:

- Physical connection of the device.
- Installation of the device dependencies (e.g. software and drivers) on client machines.
- Configuration of client machines to use device, or to allow them access to the device.

While the physical connection of a network-attached device is often trivial, the installation of the device in such a manner that it is readily available to all client entities that wish to utilize its resources is not. This process often involves the installation of drivers or specialized software for the device, as well as configuration of the operating system of each client machine in order for the device to be visible to the end user. Most operating systems do not support the installation of network-attached devices in an efficient manner; they do, however, often support the installation of network-capable devices that are hosted on a supported server [12] [13].

2.4.2 Configuration

The configuration of network-attached devices is considerably easier than that of devices shared using a dedicated or shared server. Most devices are configurable via a standard interface protocol (e.g. HTTP) that allows administrators easy access to device settings. This centralized configuration model allows administrators to set device options once, from any network location, and know that, since the device manages its own resources, any restrictions or configuration options they specify will be applied to all entities that utilize the device.

In the past, administrators were required to configure the device via the host system that facilitated network access to the device. The configuration options applied to the host system did not necessarily reflect the options that would be applied to users of the device. This posed both a management and maintenance issue, which was eventually mitigated by the inclusion of configuration screens built-in to the devices. These interfaces were limited in functionality and offered few configuration options; the advances in fully network-attached devices have significantly improved both the configuration options and the methods by which these options are accessed and applied.

2.4.3 Maintenance

The maintenance of network-attached and network-capable devices is a task that has become nearly trivial [14]. Whereas isolated devices required administrators and support personnel to monitor them for regular maintenance (e.g. cleaning cycles and ink levels), devices on a network can send status messages and error information to administrators as necessary. Such information allows support personnel to order supplies regularly, perform maintenance tasks, perform upgrades, and resolve errors quickly. This management feature results in decreased downtime, efficient allocation of support personnel, and extended device operational lifetime through regular maintenance.

It is worth noting that the maintenance of network-capable devices is not as trivial as that of network-attached devices. This is due to the dependency of the device upon the server that manages its data and network access. Maintenance issues that affect the device server will then affect the device itself, possibly rendering it a useless network vestige. Further, changes in the server configuration of network-capable devices may require the reconfiguration of client machines that access the device – a process which could be extremely time- and resource-intensive, depending on the organization.

The maintenance of devices made network-attached via a dedicated server are increasingly complex due to a lack of remote access to status information. For example, an administrator may notice that a device shared by a host may not be operating correctly. This means that the administrator must diagnose both the device and the host to determine which has caused the error. Further, this process requires that an administrator go to the physical location of the device to fix the error, which may be some distance, without knowing anything about the error beforehand. In organizations with various remote locations, it is significantly more cost-effective to be able to remotely tell a user that their device is, for example, out of ink, than to drive for 3 hours to diagnose the same problem. The advantages of a network-attached or network-capable device are, in this example, notable.

3 Modern Network-Attached Device Management

As was illustrated in the previous section, the management of network-attached devices is significantly easier and more efficient that that of devices requiring dedicated or shared servers. However, the modern use of network-attached devices is not as an autonomous entity, but rather as a network-capable device that requires a server. As shall become clear, this choice results from limitations within the operating systems of the network entities that wish to utilize the network-attached device [15]. Specifically, this deficiency is within the installation and maintenance mechanisms that the systems provide for centralized device management. As this section will illustrate, the implementation of new mechanisms that support the optimal management paradigm for network-attached devices is required in order to fully exploit the features that these devices provide.

3.1 Paradigms

When managing a network-attached device, administrators have two options: treat the device as a network-capable device, or as it is – an autonomous, network-attached entity. The choice of management paradigm determines the manner in which a network-attached device will access data and be accessed by clients on the network. If the device is installed with a server to manage its data access, it is installed as a network-capable device. Please reference Figure 4 for more information. This is the most common scenario in organizations, as the server can often broadcast device information and automate device installation to clients to some degree. If the device is installed with no supporting hardware as an autonomous device (which it is), it is installed using a network-attached paradigm (see Figure 5). This is the optimal paradigm in terms of network bandwidth use and device autonomy.

3.1.1 Network-Capable Devices

Focusing on the installation portion of the life cycle for network-attached devices as network-capable devices (the most common configuration), the basic procedure for the addition of a new network-attached device within an organization consists of the following steps:

- Physically connect the new network-attached device to the network.
- Install the device drivers and software on a particular server that will host the device.
- Install references to the new device on workstations. The workstation will then access the device through the server where the device software is installed.
- Allow users to customize device properties (to the allowed extent) as they wish.

These steps represent the general procedure that system administrators must undertake in order to fully integrate a new network-attached device into their network. Note that more is required than simply connecting the device; the device is completely installed when it is easily available to users. The meaning of “easily,”in this context, means that the user can choose to use the device without preparing it for use (with the exception of customizations).

Currently, most system administrators will complete the first two of the above steps easily and efficiently. The third step is the one that causes administrators the greatest problems and requires the greatest amount of time. This is due to the simple fact that most users on most systems lack either the expertise or the permissions to install a new device on their system. Thus, a system administrator is required to install the device reference on the workstations.

In order to completely understand the importance of efficiently installing a new network-attached device, consider the amount of time that a system administrator must spend to complete the first three steps of the installation (the fourth is not directly within their purview). The first step requires only minutes, often entailing the physical placement of the device and connection, which has already been prepared or is already available. The second step also requires only a few minutes, rarely exceeding the time required to install device software and drivers on the host system. The third step requires that a system administrator go to each workstation on a network and install the device reference, as well as any drivers or software required for the device to function properly. This process increases linearly with the number of systems on the network. Thus, for even a small network of 20 workstations, an average installation time of 5 minutes equates to 100 minutes of time and effort required to complete a simple task. While this initially seems reasonable, consider that any change in the network location or status of the device requires additional maintenance and additional travel. Further, this estimate does not consider the effect of such a change in a small organization with geographically disparate locations, which would then require additional travel time and resources.

3.1.2 Network-Attached Devices

The network-capable management paradigm is sub-optimal, however it is the most common. There is one simple reason for this: operating system support exists for network-capable devices, but not for network-attached devices. While operating systems may treat the installation of network-attached devices in different ways, by and large they offer far greater support for network-capable devices than for network-attached devices. This is due to the fact that, on most networks, a shared or dedicated server can broadcast device availability information to other entities that wish to use a device, thus simplifying the installation process.

The installation of an autonomous network-attached device under a network-attached paradigm differs from that of the network-capable paradigm in that device software must be installed on each client machine that wishes to use the device. In a network-capable paradigm, the installation of such software needs to occur only on the server system; here, it must occur on every system, although the installation of a driver is often the only requirement. Because operating systems do not easily support the utilization of network-attached devices under the paradigm for which they were designed, administrators have continued to use the sub-optimal network-capable management paradigm. In order to reap the full benefits of network-attached devices, they must be supported fully under their inherent paradigm.

3.2 Installation and Maintenance Mechanisms

As operating systems have evolved, so have their intrinsic capabilities [6]. Modern operating systems include mechanisms for the addition of both locally- and network-attached devices. However, most of these abilities are limited, requiring user intervention to add a new device.

As system administrators know, users are largely incapable of administering their own machines, for any of a multitude of reasons. The most efficient method of installing a new device is with the assistance of a system administrator. This system is adequate for a static network topology that requires rare updating or reconfiguration. However, for even small organizations with several workstations, the effort required to perform regular maintenance or incorporate new devices makes these tasks daunting and expensive at best. Hence, mechanisms for the automation of workstation maintenance – including device installation – have been developed since the time and resource availability of administrators for individual workstation updates is limited. Administrators utilize two main methods of automation: scripting and operating system / third-party solutions.

Within the context of these methods of automation, the only devices that can be reliably installed are printer-like devices. Printer-like devices include (obviously) printers and fax machines. The support for the fax machines is limited to sending faxes in many cases.

3.2.1 Scripting

All modern operating systems allow scripting in some manner. Further, they all include options for scripts to be executed when users log on to a system and, sometimes, when they log off. Although both the scripting syntax and flexibility of the various operating systems vary significantly, they all possess the capability to be used to install and, to a limited degree, manage network-attached devices.

3.2.1.1 *NIX

One of they key strengths of the *NIX operating system is scripting – it is replete with utilities that allow administrators and users to automate and simplify any task. The cost of such automation is an in-depth knowledge of both the operating system and the built-in functions that it provides for scripting. Thus, the use of scripting for device installation is usable by only an elite set of administrators that understand the syntax and capabilities of the system. Irresponsible or uncertain use of these scripting techniques may have undesirable consequences.

Within the scope of this thesis, the scripting functions that allow system administrators to control printer installation are of interest. Because *NIX is a largely file-oriented operating system, printer installation via scripting is a simple matter of copying or modifying the appropriate files on the workstations to include the new printer and driver (as necessary).

Using a bash (GNU Bourne again shell) script [13], administrators can specify printers using a log on script. Depending on the particular distribution involved, installing the printer entails modifying a few system files to include the new network device and installing a device driver. Modifying the system files can often be accomplished by replacing a local configuration file with a copy stored on a server that lists all printers available to that user or computer. Since bash scripts are very flexible, they can be tailored to a great degree, allowing specific users access to specific printers, group access to printers, or organization-wide access to resources. Further, maintenance of the printer options can be accomplished using built-in scheduling features of the operating system. Installation of the drivers can also be accomplished via execution of the manufacturer software or, more likely, copying required files to workstations from a server location.

The point of this scenario is to make clear the fact that printers can be installed using scripting on a *NIX system. In fact, installation can be accomplished with a great deal of efficiency and control. The drawback to this scenario is maintenance: uninstalling or modifying a printer may become difficult as script size and readability may become difficult. For example, the replacement of local configuration files may remove user customizations, or may require modification of a poorly-documented installation script. While inconvenient when infrequent, a network that undergoes constant updates would make customization on the part of the user an exercise in futility.

3.2.1.2 Mac OS X

The Mac OS X operating system is a variant on *NIX. Specifically, it is based on the BSD distribution (or flavor) of the *NIX operating system. As such, the system contains the built-in scripting functions of the generally-distributed *NIX operating systems, such as the bash interpreter. Thus, scripting printer installation, while it may involve very different files and procedures, is just as attainable in Mac OS X as it is in *NIX. However, the primary method of printer installation on a Mac OS X system is not scripting, but rather centralized administration.

3.2.1.3 Microsoft Windows

The Microsoft Windows operating systems (2000, XP, and Server) contains support for scripting at both a basic and a high level of sophistication. However, Windows is not designed around its scripting system as *NIX and Mac OS X are, but rather its graphical interface and dynamically-linked libraries. As such, scripting commands entail a higher degree of overhead when executed, and offer significantly less feedback.

The most basic scripting level in Windows is a command script, which contains a series of DOS-like commands to be executed by the system. Since Windows is not oriented toward such access mechanisms, these scripts almost immediately make use of system calls within dynamically-linked libraries that offer robust system access. For example, command scripts may make use of older utilities such as “net” or “net use.”

More robust scripts will make use of the Windows libraries such as “PrintUI.dll,” which offer a variety of options to administrators for installing and removing printers.

At a higher level, Windows workstations support scripting in Visual Basic. Certain aspects of this language inherently support the mapping or installation of network devices. The syntax and functions allow devices to be mapped to the workstation in a manner similar to the creation of a grocery list. The flexibility of these scripts is comparable to those of *NIX, but are still limited by the built-in functionality offered by the language.

3.2.2 Centralized, Robust Management

Ostensibly, operating systems do not rely on scripting for the majority of printer management tasks. Scripting the installation of devices is unreliable and often restricted by the scripting support of the system. For example, a system may support device driver installation, but not removal or update. Further, scripting utilities often provide minimal, if any, feedback as to the status of the installation or any errors encountered.

To this end, enhanced utilities (often including a graphical interface) exist that allow users and, in some cases, system administrators to install network devices on a workstation either remotely or locally using a client-server communication model. This management model is key to the popularity of the network-capable device paradigm. It is because these advanced installation features support only network-capable devices that the use of network-attached devices under their native paradigm has been so limited.

However, even under this supported feature set, these systems are capable of managing only printer-like devices; this does not include the broader network-attached or network-capable device categories. The developments in printer management do illustrate the elegant management systems that will eventually be extended to a larger scheme. The following section illustrates some of these mechanisms, using network-attached printers as an example.

3.2.2.1 *NIX

Newer versions of the *NIX operating system contain a system called LPRng. This utility simplifies the tasks involved in printer management. Specifically, in the use of network-attached printers that do not require specialized drivers. The installation of drivers for a particular device is still left, primarily, to scripting techniques.

The LPRng system consists of modified versions of standard *NIX printing functions and daemons, such as lpr, lpd, lprm, lpc, etc. [13]. The architecture *NIX print spooler and the interaction of its component functions is shown in Figure 7. Here, the path of a document is shown going directly to a printer (via lpr and lpd), through an additional print filter (the path lpr, lpd, filter), connecting to a remote queue manager (via the path lpr, lpd, lpq/lprm/lpc), and printing via another machine using a print filter (the path lpr, lpd, lpd, filter).

For LPRng, these functions have been modified to accept not only standard configuration options from local files, but to accept information in a client-server model,

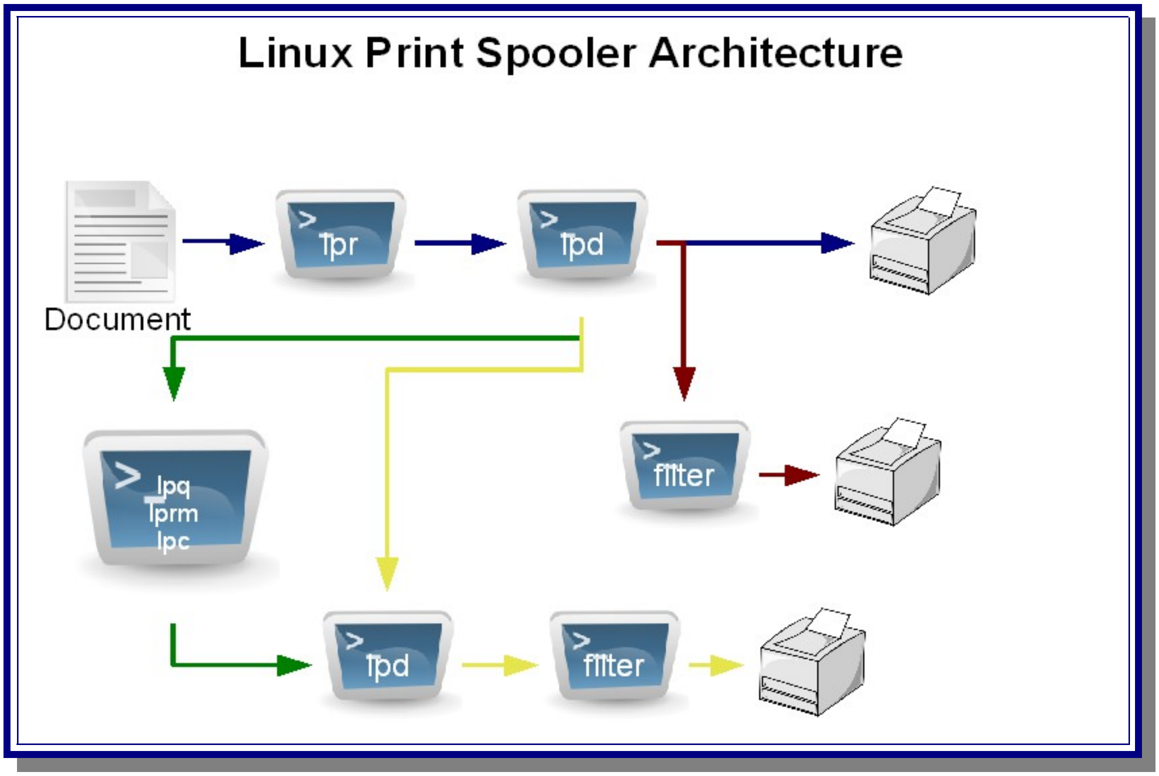


*Figure 7: The general *NIX print spooler architecture*

and through a GUI. The local configuration files used by LPRng are given in Figure 8. This graphic also illustrates the general operation of the printing daemon lpd, which is to utilize configuration files and spool jobs to a specified temporary directory or other location.

In this system, scripting can be used to copy a small set of files containing available printer information to clients. Further, clients can be configured to communicate with specified servers or devices on a network in order to configure themselves. This system is flexible and, quite nearly, ideal for printer management. However, it does not address the need to install other types of network devices, such as fax machines or scanners. Other projects are underway that may address this shortcoming, but they have not been made a standard part of (or stable release for) the *NIX operating system.

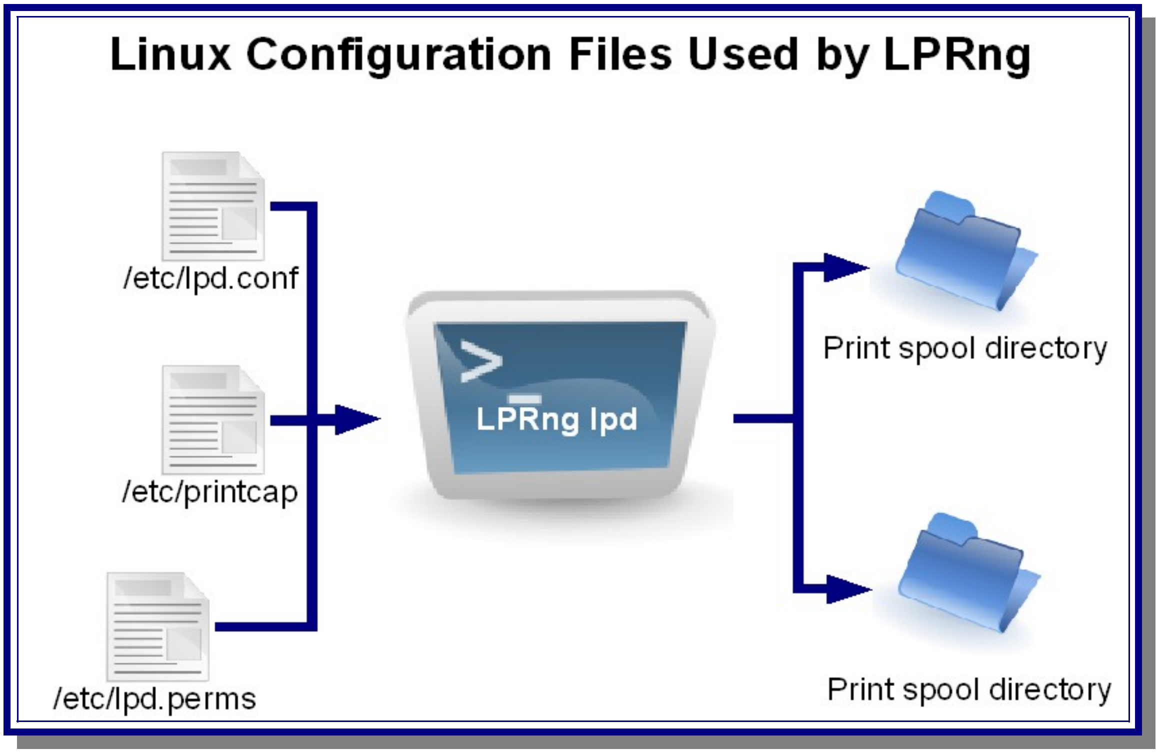


Figure 8: Configuration files used by LPRng and the operation of the lpd application

3.2.2.2 Mac OS X

Although the Mac OS X operating system contains the LPRng printing system, it relies on more advanced configuration mechanisms to manage printers. Mac OS X systems contain domain administration tools similar to those on Microsoft Windows Server®. Computers running Mac OS X as servers have the ability to control all workstations on the network to an extremely specific degree, in terms of printer management.

Directory Services provides a repository of information that can be used to configure network clients on a Mac OS X network [16]. Using this central store of information, workstations can be assigned network resources (such as printers or faxes) as they become available. The client-server model of communication involved periodic, startup, and log on updates to client machines that refresh the local cache and settings. For example, a system administrator may add a network printer to a server and specify that the printer be available to a particular group of users, or to all users on a computer. This feature conveniently allows system administrators to update available devices from a central location.

3.2.2.3 Microsoft Windows

Microsoft Windows Server® systems contain a management system comparable to Directory Services in Mac OS X: it is referred to as Active Directory. Active Directory is a central repository of network information made available to all workstations on the network as necessary. Of particular interest is the ability to view printers that have been listed for publication – these can be installed easily by a user.

This system allows printer listing and, like Directory Services, requires that a single Windows-compatible computer host the queue for the printer in order to manage its jobs. Though it does not allow assignment of printers to users, a sub-component called Group Policy is designed to handle such tasks. Group Policy allows administrators to distribute applications, configuration information, and other information to workstations. Further, this component of the client-server architecture is extensible, meaning that it allows users or developers to create their own extensions for workstation management. A diagram illustrating the application of Group Policy to client computers is shown in Figure 9.

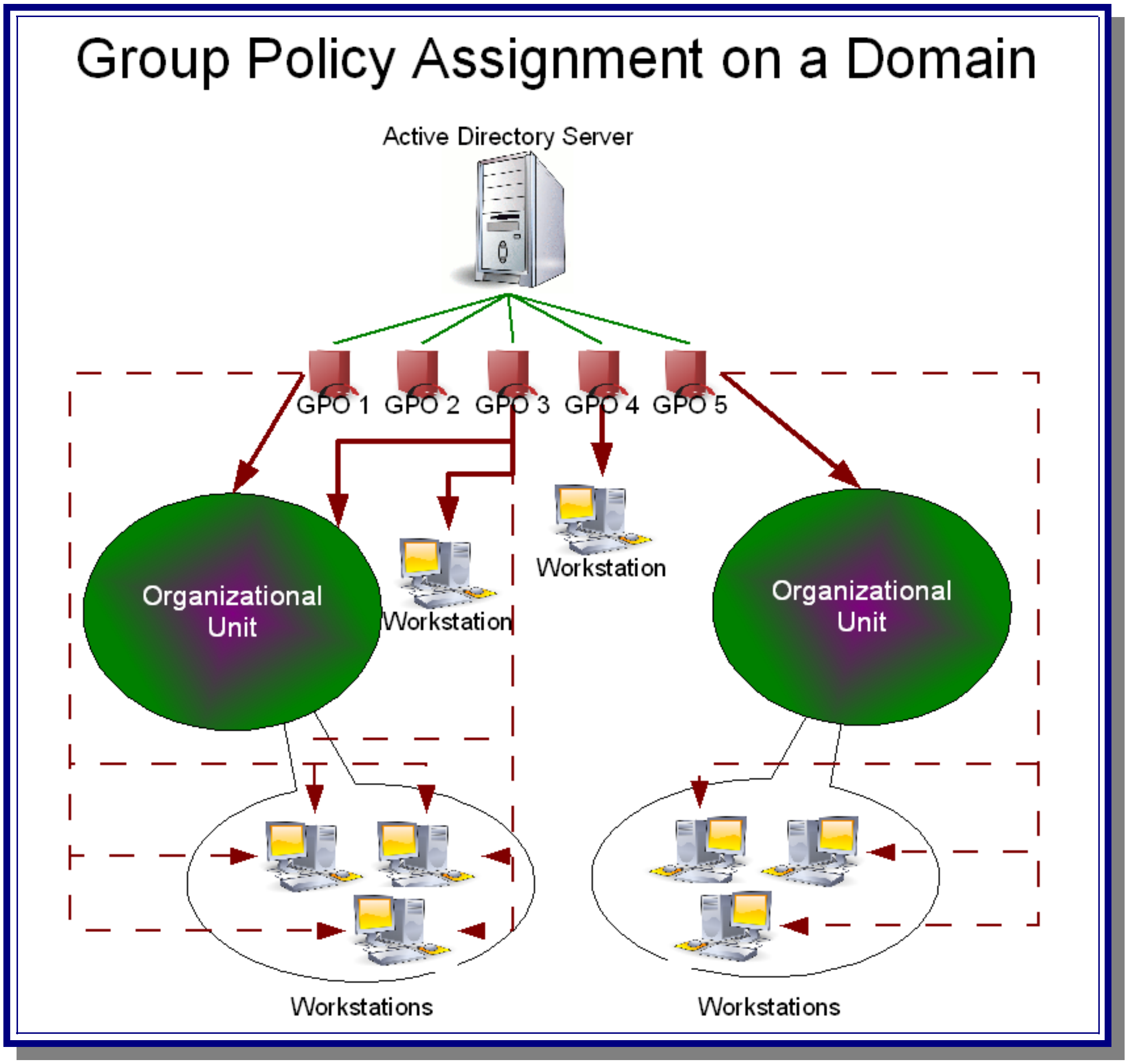


Figure 9: An illustration of the application of Group Policy across a domain, including organizational units and individual machines

Here, the Active Directory server for the domain stores an arbitrary number of Group Policy Objects that define settings and applications to be installed on target machines. This graphic illustrates the application of policies to both target machines and organizational units (consisting of machines and/or users), though policies may also be applied to individual users, and domains as a whole.

Printer management has been integrated into the newest version of Windows

Server® 2003 (RC1) in a rudimentary way – scripting. While the next version Windows Server® (Vista) is slated to contain advanced printer management utilities [17], the current versions of Windows must rely on third-party utilities such as AutoProf PolicyMaker to automate printer installation in an efficient manner. Even in these systems, however, a Windows system is required to host the network printer.

4 An Optimal Network-Attached Device Paradigm

In order to research the benefits of network-attached device management effectively, an extension was created that allows the management of network-attached and network-connected printers and storage devices. This extension, developed as an extensible framework, allows the central management of network-attached devices. The goal was to create a standardized extension that would provide system administrators with a management tool, allowing users and system administrators to retain the ease of configuration available with current network-capable devices, while obtaining the full range of benefits that network-attached devices offer. Further, the extension was developed to incorporate other network-attached devices as necessary, requiring minimal programming.

Using this extension, the network bandwidth improvements achieved by incorporating network-attached devices under their native paradigm could be measured generally. The management features currently available for network-connected devices were incorporated into the extension, allowing both network-attached and network-connected printers to be managed in a central, standard manner that facilitates easier comparison. The remainder of this chapter provides the details of this extension, which has been simply (though not with brevity) entitled the Network Device Management Group Policy Extension.

4.1 Optimal Network-Attached Device Management

The ideal management solution for ant system administrator is that they be able to make a change in a central location, which is then propagated to client machines using the distribution mechanisms available. Depending on the operating system, the mechanisms by which this may be accomplished will vary. For example, the *NIX environment offers scripting support for management tasks, though it allows individual components to be configured for network-centric management. Mac OS X offers other mechanisms which are built into its Directory service. Likewise, Microsoft Windows offers both command-line tools and the centralized Group Policy management mechanism.

In terms of installing a network-attached device, or, for that matter, a network-capable device, system administrators wish to accomplish the following sequence of installation and configuration tasks:

- Install device drivers on network entities that will utilize the device.
- Install device references on network entities that will utilize the device.
- Retain the ability to easily modify the device drivers and references installed on network entities.

The ultimate goal of these tasks can be summarized as follows: to have any network-attached device that administrators specify be available to their users without any intervention on their part. Further, administrators require a simple interface for making these changes, in order to be able to efficiently manage their networks. Thus, any development made should utilize existing facilities of the operating system as much as possible. As a final note, any addition to the system should not, ideally, require system administrators to make any initial modification to the operating system – it should use existing distribution mechanisms to streamline the addition of a new component as much as possible.

4.2 Demonstrating an Efficient Paradigm

In order to demonstrate the benefits, utility, and feasibility of using network-attached devices to their full potential, the efficiency and benefits of the network-attached paradigm were demonstrated by creating an extension to Microsoft Windows Server® Group Policy that allows system administrators to manage network-attached devices in a centralized fashion. This software component was implemented as an extension to Group Policy, as it is already a part of the Microsoft Windows 2000 and XP operating systems that is meant for centralized management and distribution of applications over Windows 2000 and Windows 2003 Server network environments. This is an ideal environment for the development and demonstration of this capability, as the operating system and the components involved were designed with exactly these types of extensions in mind.

In order to achieve an efficient, streamlined management scheme, the project entailed the creation of three things:

- A dynamically-linked library (DLL) that provides the functionality set forth.
- The distribution of the library to client machines via a Microsoft Installer (MSI)

package, which can be distributed and installed on client machines without user intervention.

- The configuration of the system was designed to occur through a new Administrative Template that is installed on the domain servers. This template makes entries in registry keys that are propagated to clients when they perform a policy update.

4.3 Goals

One of the goals of this thesis was to create a software extension to the Group Policy component of Microsoft Windows 2000 and Windows 2003 Server that allows system administrators to automate the installation of network-attached and, as a result, network-capable devices. Using this tool, the benefits of device management and installation under the network-attached paradigm could be researched. The installation tasks that this component supports meet the following criteria:

- Allow automated installation of device drivers from a network location.
- Allow automated installation of references to network-capable devices.
- Allow automated installation of network-attached devices.
- Provide for the management of installed devices, including device removal and reconfiguration.
- Integration as a component to Group Policy.
- Distributable as Microsoft Installer (MSI) package.

The devices supported in this release are:

- Printers
- Printer-like devices, such as fax machines
- Network storage devices

The end result of this project was the implementation of a software mechanism that allows for the optimal utilization of network-attached devices. This extension will be made freely available as an open source component for Windows Server® 2000 and Windows Server® 2003 networks.

The extension illustrated that, using this simple tool, it is possible to not only utilize network-attached devices to their fullest potential, but that it is also possible to control network-capable devices and other network-attached devices in a much more efficient manner. That is, this showed that system administrators need not be restricted by the limitations of the operating systems, but rather that they may use the extensibility mechanisms built into the system to achieve network performance and device management benefits of the network-attached device paradigm with minimal management cost [18]. The utility of this extensible model greatly simplified management tasks, simultaneously allowing greater control over device installation and drivers, yet decreasing the network bandwidth usage as compared to devices controlled under the network-connected paradigm.

4.4 Resources Involved

The implementation of this demonstration of an optimal network-attached device management paradigm entailed the utilization of several technologies and resources. This section details the mechanisms, languages, and tools that were involved.

4.4.1 Operating System Support

The target operating systems for this component are as follows:

- Servers: Windows 2000 Server and Windows 2003 Server
- Workstations: Windows 2000 Professional and Windows XP Professional

These operating systems have no advanced support for printer management beyond that of shared or network-capable devices (through sharing). However, it should be noted that Windows 2003 Server RC2 does contain support for automated network-capable device installation, but the support is extremely limited, consisting of a command-line tool that must be executed by a user log on script to be effective.

4.4.2 Languages and API

The programming language of the component was C++ , using the standard Windows API [19], [20], [21]. This API provides direct, efficient, robust access to Windows resources, allowing the component to access the system in a controlled manner.

Specifically, the printing and print spooler functions of the Windows GDI will be extremely useful in installing both printing devices and the drivers required [22].

4.4.3 Microsoft Installer Packages

Microsoft Installer (MSI) packages are relational databases containing all files and configuration information necessary to fully install an application [23]. This includes file entries, configuration files, registry entries, and system modifications that must take place. Additionally, these entities may be configured before installation using built-in properties of the packages. The theory behind these packages was to create application installers that allow simplified installation, removal, and repair of applications.

As this is a Microsoft technology, support has been built into all Windows operating systems, starting with Windows 2000. These packages are utilized by Active Directory and Group Policy for automated installation of applications to client machines. Using these packages, administrators may completely configure an application, then deploy it to client machines. Such deployments require no configuration or interaction on the part of the user, past that of utilizing the newly-installed application and customizing it to their liking, if applicable.

For the purposes of this project, an MSI was created that contains the dynamically-linked library that will process the Group Policy Objects associated with network-attached device management. This package also specifies the configuration information (such as registry entries) that are required on client machines for the library to be invoked properly. The MSI itself was created using the open source tool MakeMSI, which allows programmers and administrators to script the creation of MSI packages for distribution of applications and configuration files.

4.4.4 Microsoft Group Policy

Microsoft Group Policy is a management technology that, using Active Directory, is capable of applying specific configurations to users and computers [24]. The Active Directory feature of Windows 2000 Server and Windows 2003 Server is a hierarchical representation and interface model through which clients communicate with servers, receiving information in the process [25]. Amongst the configurations that can be applied using the Group Policy component are applications, registry settings, and fully-configured applications.

Under this scheme, administrators can specify policies that are applied to sites, domains, and organizational units, further specifying whether the policies are applied to the machines or users within these groups. The policies are defined within Active Directory containers called Group Policy Objects (GPOs). These objects can contain one or more configuration options or applications for the target machines or users. Creation of GPOs occurs on Domain Servers and application occurs in a hierarchical fashion, with policies originating from the Primary Domain Controller having highest priority. This technology was implemented by Microsoft in an effort to provide administrators with powerful group management tools that allow simplified management of domain computers. An illustration of this deployment scheme is given in Figure 10. This shows the combination of an application and configuration options within a Group Policy Object, which is then distributed to client machines by the Active Directory server (which is usually the Primary Domain Controller).

Updates to Group Policy on client or workstation computers occur both at startup and periodically during operation. During startup, clients query the server for Group Policy settings, comparing their last known configuration to that specified by the servers. When differences are found, registered DLLs that support and process the GPOs are invoked, receiving references to objects that have changed or been deleted. Upon execution of these libraries, policies are applied to the client computers. During the operation of the computer – that is, when a user is active on the machine – Group Policy

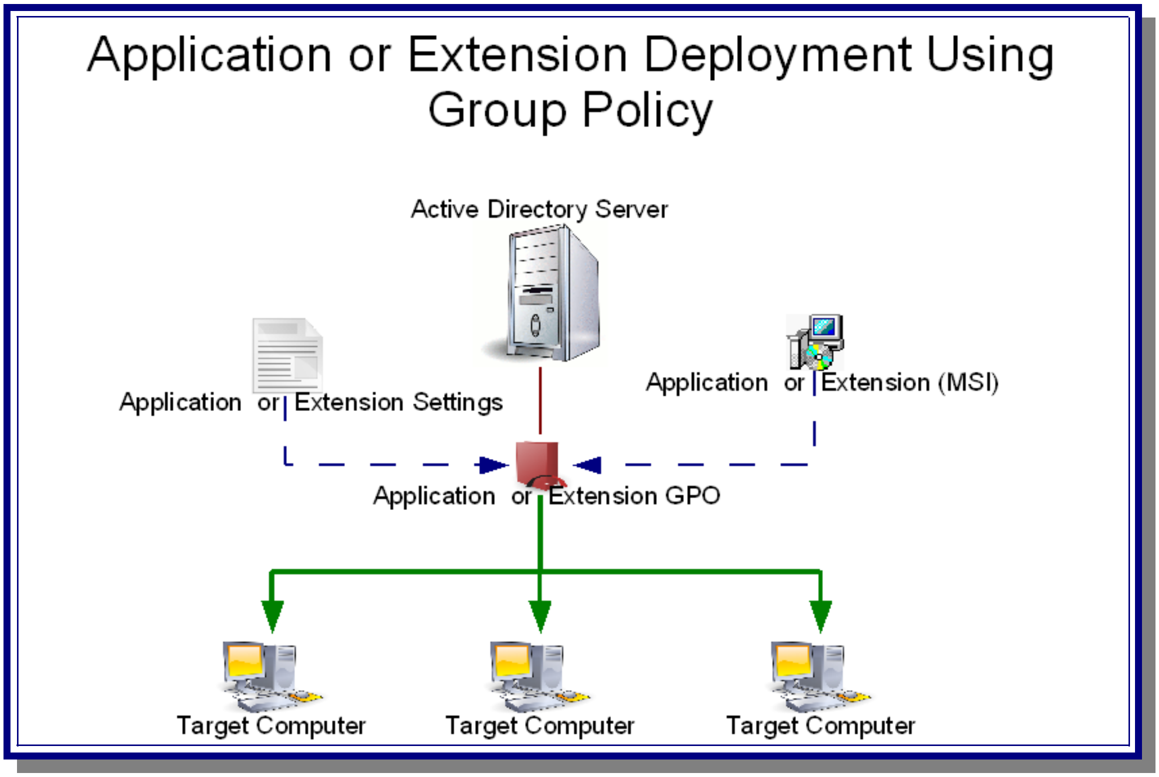
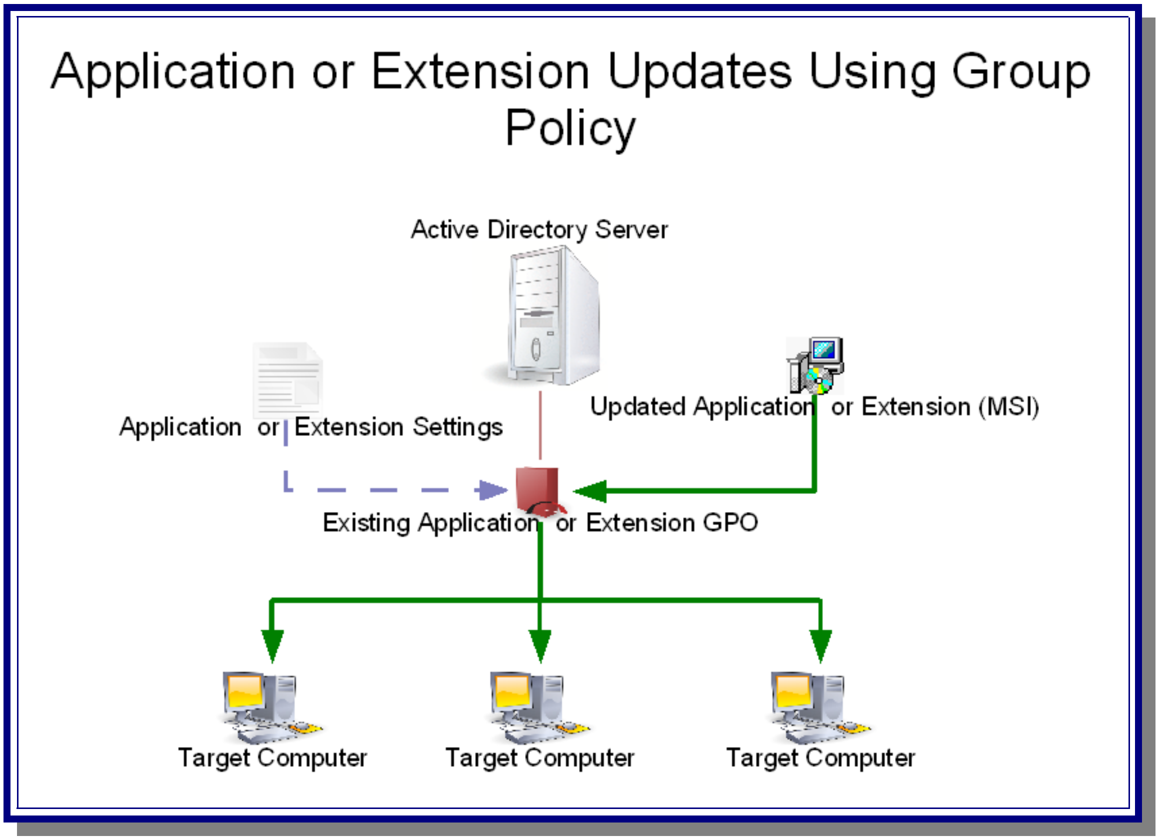

Application or Extension Deployment Using Group Policy
Active Directory Server
Application or Extension Settings
Application or Extension (MSI)
Application or Extension GPO
Target Computer
Target Computer
Target Computer

Figure 10: Example application deployment (with configuration settings) using Group Policy

updates occur, by default, every 90 minutes, within a window of randomization. Any policies that would interfere with the user are not applied until the user logs out; changes that require a restart are not applied until the system is restarted by the user. The process for the application of policies during operation is the same as during startup.

The design of the Group Policy system allows the creation and addition of custom DLLs to process new policy settings. The specifications for these libraries are freely available on the Microsoft website, and are present in the References of this thesis. Once created, libraries need merely be registered within a section of the registry in order to be invoked when policy updates are applied. Thus, by creating a new library to manage the installation of network-attached devices, the application of the network-attached device paradigm to an entire network can be affected easily, with minimal effort and maximum flexibility. This flexibility allows the library to carry out any functionality on the local system that once chooses to implement, so long as configuration information from the server to the clients is passed appropriately.

Finally, the independence of applications and configuration settings within a GPO must be clarified. Specifically, a change in one does not cause the other to be redeployed. For instance, if a GPO contains both an application and configuration settings for that application, an update of the contained application does not cause the configuration settings to be reapplied. Likewise, updated configuration settings do not cause the application to be reinstalled. Only the affected components are processed since the system is concerned only with changed or deleted policy objects. This is illustrated in Figure 11 and Figure 12.


Application or Extension Updates Using Group Policy

Active Directory Server
Application or Extension Settings
Updated Application or Extension (MSI)
Existing Application or Extension GPO
Target Computer
Target Computer
Target Computer

Figure 11: An illustration of the independent nature of configuration settings when the associated policy object is updated

Respectively, these graphically illustrate that an updated application within a GPO does not affect the application of configuration settings, and that a change in configuration settings does not cause an application to be reinstalled. Figure 11 shows that an updated application package does not entail any change or re-application of configuration settings. Note that the domain server and the configuration graphic are both semi-transparent, illustrating that they are not reapplied or involved (beyond distribution) in the change.

Just as in the other graphic, Figure 12 represents that a change in the application settings does not cause the application itself to be reinstalled. Note that the application distributable (the MSI) is semi-transparent, representing its inactive role in the process.

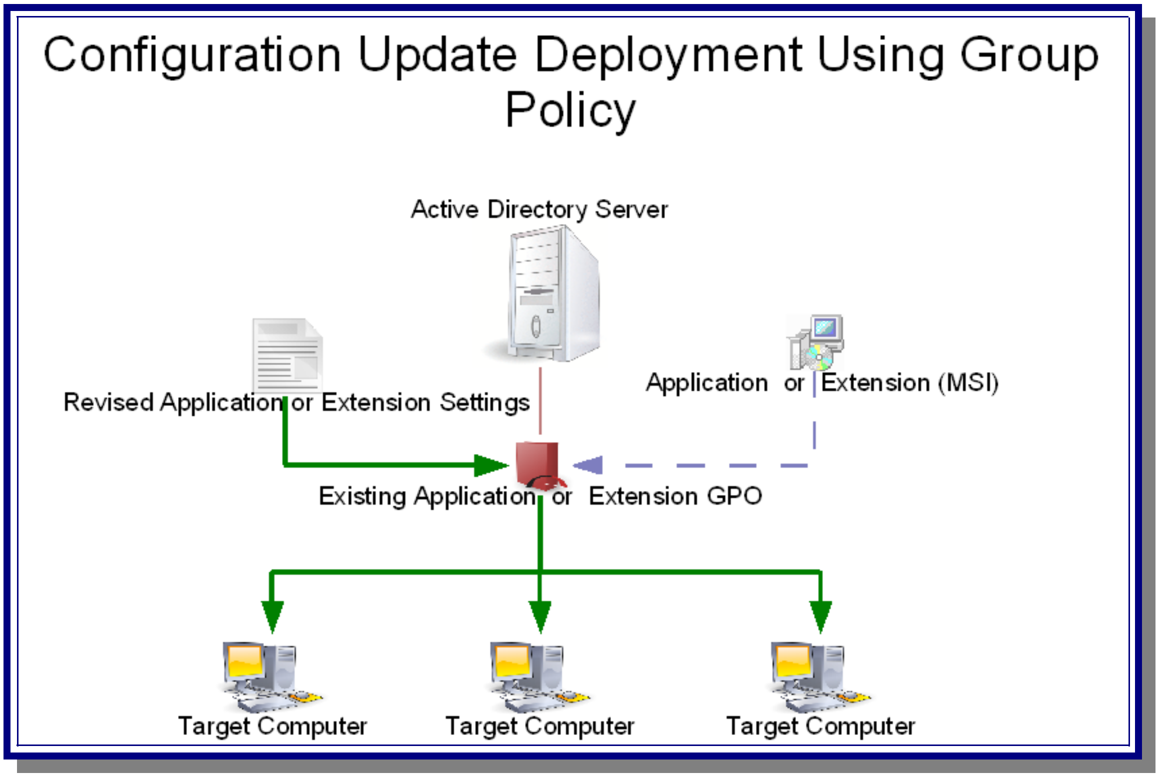

Configuration Update Deployment Using Group Policy
Active Directory Server
Revised Application or Extension Settings
Application or Extension (MSI)
Existing Application or Extension GPO
Target Computer
Target Computer
Target Computer

Figure 12: An illustration of the independence of an application installation from configuration settings within a policy object when the latter is modified

4.4.5 Administrative Templates

Administrative Templates are the mechanism by which Group Policy Objects are created and configured [26], [24]. Any single GPO can utilize any number of Administrative Templates to specify a configuration set. These templates consist of text files that define registry entries that, when set using the options specified in the template, have a specified effect on systems to which they are applied. For example, enabling an option to restrict device installation exclusively to Administrators would specify that one or more registry entries are modified (which may entail the creation, modification, or deletion of such keys) . Once these changes are applied to clients, the desired effect is produced.

The resources indicating the format and specification, as well as other information regarding Administrative Templates, can be found in the References section of this document.

5 Software Model

The current chapter details the organization and implementation of the project. In a release configuration, the executable library code will be of very small size (around 25 kB). However, the extra overhead involved in packaging, distributing, and registering the library properly on clients entailed significantly more effort for safe, efficient operation. This security is reflected in the underlying architecture of the extension, which is detailed in the proceeding sections. The contents of this chapter are details regarding the implementation of the prototype network-attached device extension to Group Policy.

Subsequent chapters elucidate upon implemented prototype, as well as the associated evaluation and comparison of the model against existing tools. The thesis then include an analysis of the network performance improvements and management features determined using the extension. Concluding the thesis is a commentary on future work that should or can be undertaken to enhance the system, accompanied by a summary of the research results. As this system is designed to be operable and freely distributable, enhancements will add to the overall utility of the package.

5.1 Requirements Specification

This section contains a description of all the features and constraints related to the project. In essence, each functional requirement is a statement of a feature that the system must have. Non-functional requirements are constraints placed upon the system, which may entail time constraints, performance requirements, or specific tools that must be utilized.

5.1.1 Functional Requirements

The following constitute the functional requirements of the client-side group policy extension that will facilitate utilization of the network-attached device paradigm.

[FR 1] The library extension shall allow users to administer printers installed on client machines from a central location.

[FR 2] The extension shall include an Administrative Template for server-side configuration.

[FR 3] The extension shall allow the remote installation of printers.

[FR 4] The extension shall allow the remote installation of printer drivers.

[FR 5] The extension shall allow the remote installation of drivers for printer-like devices.

[FR 6] The extension shall allow the remote removal of installed printers.

[FR 7] The extension shall allow the remote removal of installed printer drivers.

[FR 8] The extension shall allow the remote removal of installed drivers for printer-like devices.

[FR 9] The extension shall allow the remote mapping of / connection to network storage.

[FR 10] The extension shall support network-capable printers.

[FR 11] The extension shall support network-attached printers.

[FR 12] The extension shall support network-capable printer-like devices.

[FR 13] The extension shall support network-attached printer-like devices.

[FR 14] The extension shall support connections to network-capable storage.

[FR 15] The extension shall support connections to network-attached storage.

[FR 16] The extension shall store and retrieve registry entries exclusively in locations designed for Group Policy settings.

[FR 17] The extension shall support per-machine configurations.

[FR 18] The extension shall support per-user configurations.

[FR 19] The extension shall have the ability to reinstate policies if they are incorrectly modified by the user [27].

[FR 20] The extension shall comply with Resultant Set of Policy (RSoP) requirements.

[FR 21] The extension shall be configurable via a custom user interface.

5.1.2 Non-Functional Requirements

The following constitute the non-functional requirements of the client-side group policy extension that will facilitate utilization of the network-attached device paradigm. Many of these reflect the framework interface restrictions that the creation of a client-side Group Policy extension entails.

[NF 1] The library extension shall be written in the C++ programming language.

[NF 2] The library extension shall be available as a compiled 32-bit DLL.

[NF 3] The library extension shall be available as a compiled 64-bit DLL.

[NF 4] The library extension shall be registered as a part of the operating system.

[NF 5] The library extension will be deployable from a central location.

[NF 6] The extension will be distributable via a Microsoft Installer package.

[NF 7] The Administrative Template will be distributable via a Microsoft Installer package.

[NF 8] Microsoft Installer packages will be created using the MakeMSI tool.

[NF 9] The extension shall use the Windows API and other built-in operating system functions to achieve functionality.

[NF 10] The extension shall not use any non-standard system modification methods.

[NF 11] The extension shall not access the file system directly.

[NF 12] The extension shall be integrated completely into the operating system.

[NF 13] The extension shall be easily uninstalled, if required.

[NF 14] The extension shall be completely documented and commented.

5.2 Use Cases

This section defines the use cases that the network-attached device extension allows. Since the system is configurable only from the Group Policy Management Console (excluding any custom interface), there are few direct actions that the user can take. This is especially reasonable since the purpose of this system is to automate the installation of devices, removing user interaction. Further, since all client machines merely follow the instructions of the extension, it makes sense that no other entities be elevated to the position of "actor."

In order to differentiate the architecture of this extension from the implementation that will occurred for the thesis, the use cases have been divided into abstract models and specific models. The abstract models are limited, showing only the high-level actions that can be carried out. The lower-level implementation models show the specific interface methods by which users may control the extension.

5.2.1 Abstract-Level

This section contains the use case diagram and use cases for the general architecture defined for the management of network-attached devices. The entries in this section are not meant as specific use cases, but rather as somewhat-defined methods of completing management tasks for a device. This section should be used as a reference model for the addition of any specific network-attached device to the extension.

5.2.1.1 Diagram

As shown in Figure 13, the user – which is assumed to be a system administrator – has only five high-level, generalized actions that he or she can accomplish. In this

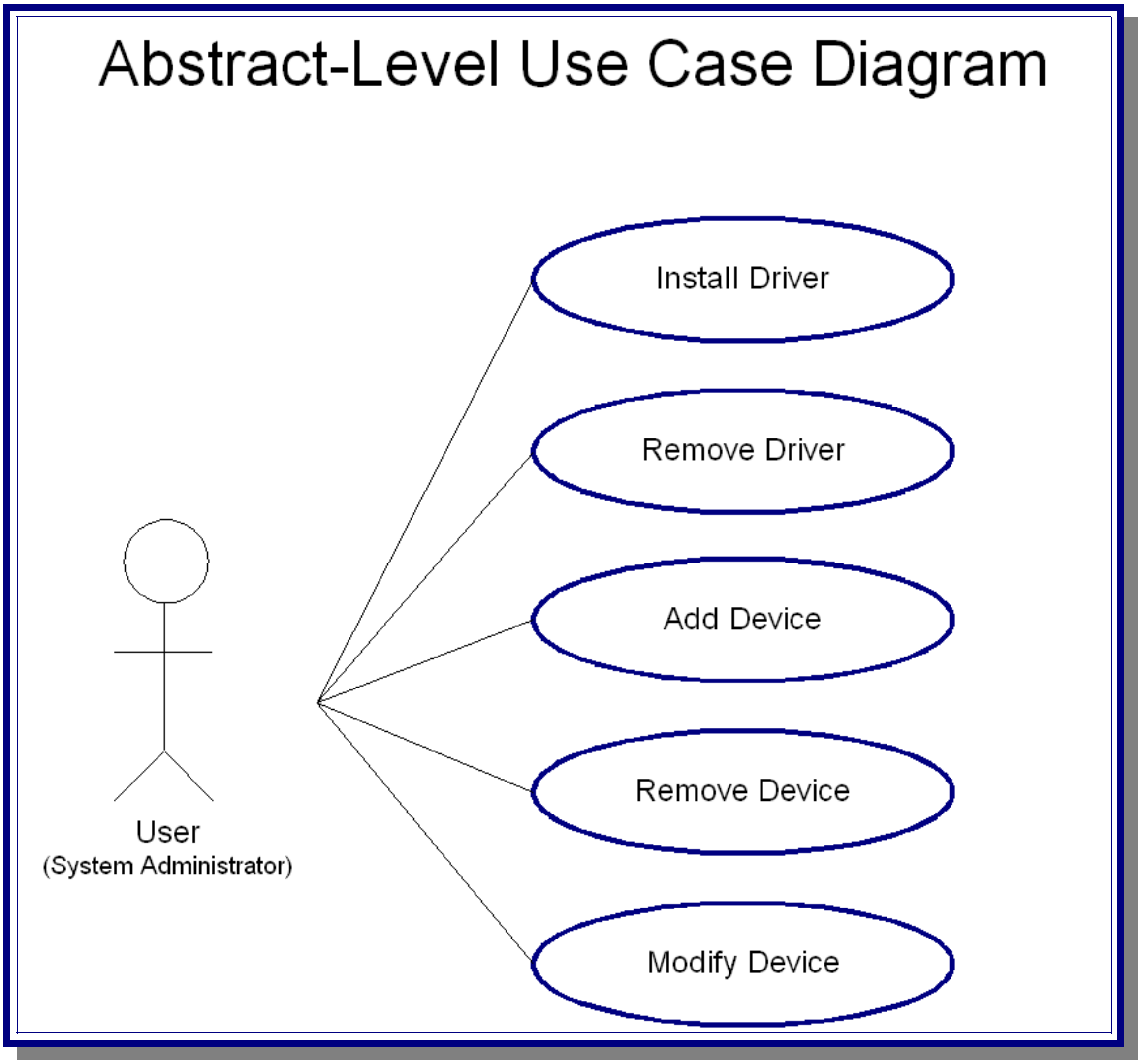


Figure 13: A high-level, abstract view of use cases for the network-attached device management extension

abstraction, administrators interact only with high-level, generic devices, which are assumed to have a uniform installation and management procedure. Since this is unfortunately not the reality, specializations of the generic device entity must be replaced with specific device instances as the extension is modified to manage more devices.

5.2.1.2 Detailed Use Cases

The following use cases are detailed interaction sequences that users will experience with the system when managing generic devices. These use cases will generally be adapted to manage a specific device as the extension is modified to support that device. These cases are meant to guide future design efforts to incorporate new devices into the management scheme. It is also pertinent to point out that since the only input to the system comes via the Group Policy Management Console (using Administrative Templates), there are very few use cases to represent in this section.

Use case: Install Driver
ID: UC1
Actors: System Administrator
Preconditions: ● Extension is installed on clients. ● Extension templates are installed on server. ● Group Policy objects have been configured and assigned. ● The driver is available to users on a network share.
Flow of events: 1. The Administrator creates a new driver entry in the Group Policy object. 2. The Administrator defines the driver location. 3. The Administrator defines the driver name that will be installed. 4. The Administrator saves the entry. 5. The Administrator re/applies the policy.
Postconditions: ● Drivers will be installed when Group Policy is refreshed on client machines. ● Incorrect parameters will not affect clients.

Figure 14: Driver installation use case

Use case: Remove Driver
ID: UC2
Actors: System Administrator
Preconditions: ● Extension is installed on clients. ● Extension templates are installed on server. ● Group Policy objects have been configured and assigned. ● The driver is installed on client systems. ● The driver entry is present in a Group Policy object.
Flow of events: 1. The Administrator deletes the driver entry in the Group Policy object. 2. The Administrator saves the change. 3. The Administrator re/applies the policy.
Postconditions: ● Drivers will be removed when Group Policy is refreshed on client machines. ● Incorrect parameters will not affect clients.

Figure 15: Driver removal abstract-level use case

Use case: Add Device

ID: UC3

Actors:
System Administrator

Preconditions:

- Extension is installed on clients.
- Extension templates are installed on server.
- Group Policy objects have been configured and assigned.
- The driver is installed on the client machines.

Flow of events:

1. The Administrator creates a new device entry in the Group Policy object.
2. The Administrator defines device properties.
3. The Administrator saves the entry.
4. The Administrator re/applies the policy.

Postconditions:

- The device will be made available to users when Group Policy is refreshed.
- Failed installations will not be seen by users.

Figure 16: Device installation abstract-level use case

Use case: Update Device

ID: UC4

Actors:
System Administrator

Preconditions:

- Extension is installed on clients.
- Extension templates are installed on server.
- Group Policy objects have been configured and assigned.
- The driver is available to users on a network share.
- The device has already been installed.

Flow of events:

1. The Administrator opens an existing device entry.
2. The Administrator modifies the device properties.
3. The Administrator saves the entry.
4. The Administrator re/applies the policy.

Postconditions:

- Device changes will become available to clients when Group Policy is refreshed.
- Incorrect parameters will not affect clients.

Figure 17: Device update abstract-level use case

Use case: Remove Device
ID: UC5
Actors: System Administrator
Preconditions: ● Extension is installed on clients. ● Extension templates are installed on server. ● Group Policy objects have been configured and assigned. ● The device entry is present in the Group Policy object.
Flow of events: 1. The Administrator deletes the device entry within the Group Policy object. 2. The Administrator saves the entry. 3. The Administrator re/applies the policy.
Postconditions: ● The device will be removed from clients when Group Policy is refreshed. ● Devices on which the device was never installed will be unaffected.

Figure 18: Device removal abstract-level use case

5.2.2 Implementation-Level

This subsection contains use case diagrams and detailed use cases that reflect the devices incorporated into the current extension release. As such, the greatest difference between the content of this section and that of the previous is the modification of some use cases to reflect the devices supported. For example, the generic device shown in Figure 13 has been replaced by specific devices, such as network storage and network-capable printers. The detailed use cases reflect these refinements.

5.2.2.1 Diagram

Figure 19 shows the options and interaction possibilities available to a system administrator using the current release of the extension. Note that the installation of a printer driver is the same for both network-attached and network-capable devices, requiring only one interaction case. Since network-attached storage does not require a driver, no related chart entity exists aside from the connection to that storage. Further, note that the installation of network-capable and network-attached devices differs, requiring that their interaction with the user be represented individually.

As Figure 19 shows only use cases for specific devices, the generic use cases from Figure 13 that are unchanged by the addition of a specialized device are omitted. Thus, the modification and removal of a device are the same for all devices, and the entities are suppressed. This suppression is indicated by the lowest entity, containing an ellipsis.

5.2.2.2 Detailed Use Cases

The following detailed use cases describe the interaction that a system administrator will have with the network-attached device management extension. This interaction will occur via the Group Policy Management Console on the Active Directory server (or via a remote access method), and will allow the administrator to manage

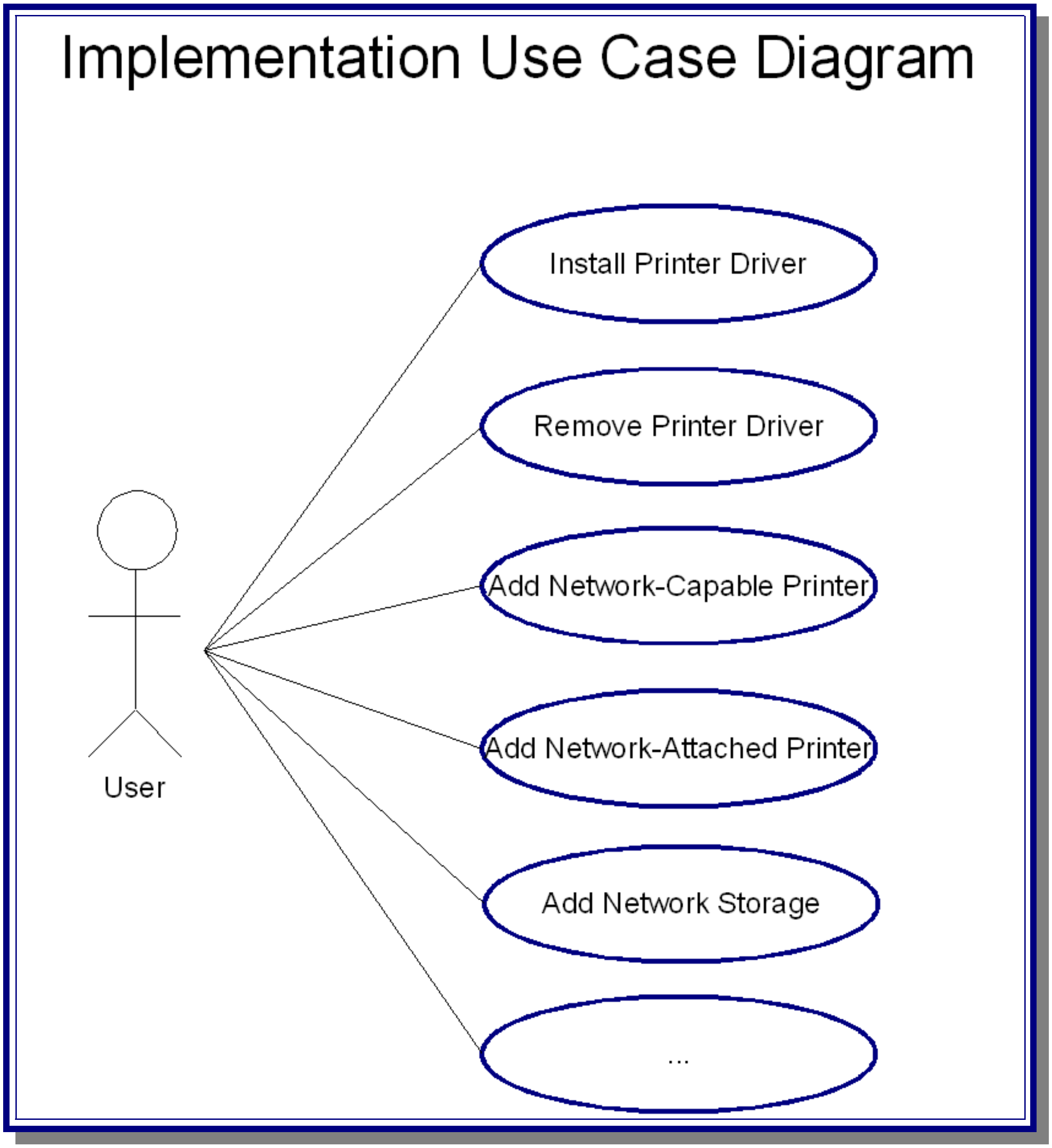

Implementation Use Case Diagram
User
Install Printer Driver
Remove Printer Driver
Add Network-Capable Printer
Add Network-Attached Printer
Add Network Storage
...

Figure 19: An implementation-level use case diagram showing the use cases for the management of specific devices

network-capable printers, network-attached printers, and network storage connections of clients.

Use case: Install Printer Driver
ID: UC1
Actors: System Administrator
Preconditions: ● Extension is installed on clients. ● Extension templates are installed on server. ● Group Policy objects have been configured and assigned. ● The driver is available to users on a network share.
Flow of events: 1. The Administrator creates a new entry in the Group Policy object Driver section. 2. The Administrator defines the printer driver location (the path to the *.inf* file). 3. The Administrator defines the printer driver name that will be installed (as defined within the driver). 4. The Administrator saves the entry. 5. The Administrator re/applies the policy.
Postconditions: ● The printer driver will be loaded into the driver store of the client and installed on the machine. ● Incorrect path or name will not result in a displayed error.

Figure 20: Printer driver installation implementation-level use case

Use case: Remove Printer Driver

ID: UC2

Actors:
System Administrator

Preconditions:

- Extension is installed on clients.
- Extension templates are installed on server.
- Group Policy objects have been configured and assigned.
- The driver is installed on client systems.
- The driver entry is present in a Group Policy object.

Flow of events:

1. The Administrator deletes the printer driver entry in the Group Policy object, located in the Driver section.
2. The Administrator saves the change.
3. The Administrator re/applies the policy.

Postconditions:

- Printer drivers will be uninstalled and deleted from the driver store when Group Policy is refreshed on client machines.
- Incorrect parameters will not affect clients.

Figure 21: Printer driver removal implementation-level use case

Use case: Add Network-Capable Printer
ID: UC3
Actors: System Administrator
Preconditions: ● Extension is installed on clients. ● Extension templates are installed on server. ● Group Policy objects have been configured and assigned. ● The printer driver is installed on the printer server.
Flow of events: 1. The Administrator creates a new printer entry in the Group Policy object under the Network-Capable Printer section. 2. The Administrator enters the full network share path to the printer. 3. The Administrator saves the entry. 4. The Administrator re/applies the policy.
Postconditions: ● The printer will be made available to users when Group Policy is refreshed. ● Failed installations or errors will not be seen by users.

Figure 22: Network-capable printer installation implementation-level use case

Use case: Add Network-Attached Printer

ID: UC3

Actors:
System Administrator

Preconditions:

- Extension is installed on clients.
- Extension templates are installed on server.
- Group Policy objects have been configured and assigned.
- The printer driver is installed on the client machines.

Flow of events:

1. The Administrator creates a new printer entry in the Group Policy object under the Network-Attached Printer section.
2. The Administrator enters the DNS entry or the IP address of the printer.
3. The Administrator enters a unique port name to be used by the printer.
4. The Administrator enters the display name of the printer.
5. The Administrator enters the driver name that the printer will use.
6. The Administrator enters the name of the print processor that the printer will use.
7. The Administrator saves the entry.
8. The Administrator re/applies the policy.

Postconditions:

- The printer and port will be installed and made available to users when Group Policy is refreshed.
- Failed installations or errors will not be seen by users.

Figure 23: Network-attached printer installation implementation-level use case

Use case: Add Network Storage
ID: UC6
Actors: System Administrator
Preconditions: ● Extension is installed on clients. ● Extension templates are installed on server. ● Group Policy objects have been configured and assigned.
Flow of events: 1. The Administrator adds a new storage entry within the Group Policy object under the Network-Attached Storage section. 2. The Administrator enters the server name. 3. The Administrator enters the resource name. 4. The Administrator saves the entry. 5. The Administrator re/applies the policy.
Postconditions: ● The network storage device will be added when the Group Policy on the client is refreshed.. ● Device errors will not be apparent to the user.

Figure 24: Network storage installation implementation-level use case

5.3 Deployment Diagram

The following deployment diagram depicts the run-time distribution of the components that constitute the extension. Figure 25 shows that configuration settings made on the server via the Group Policy Management Console using the Administrative

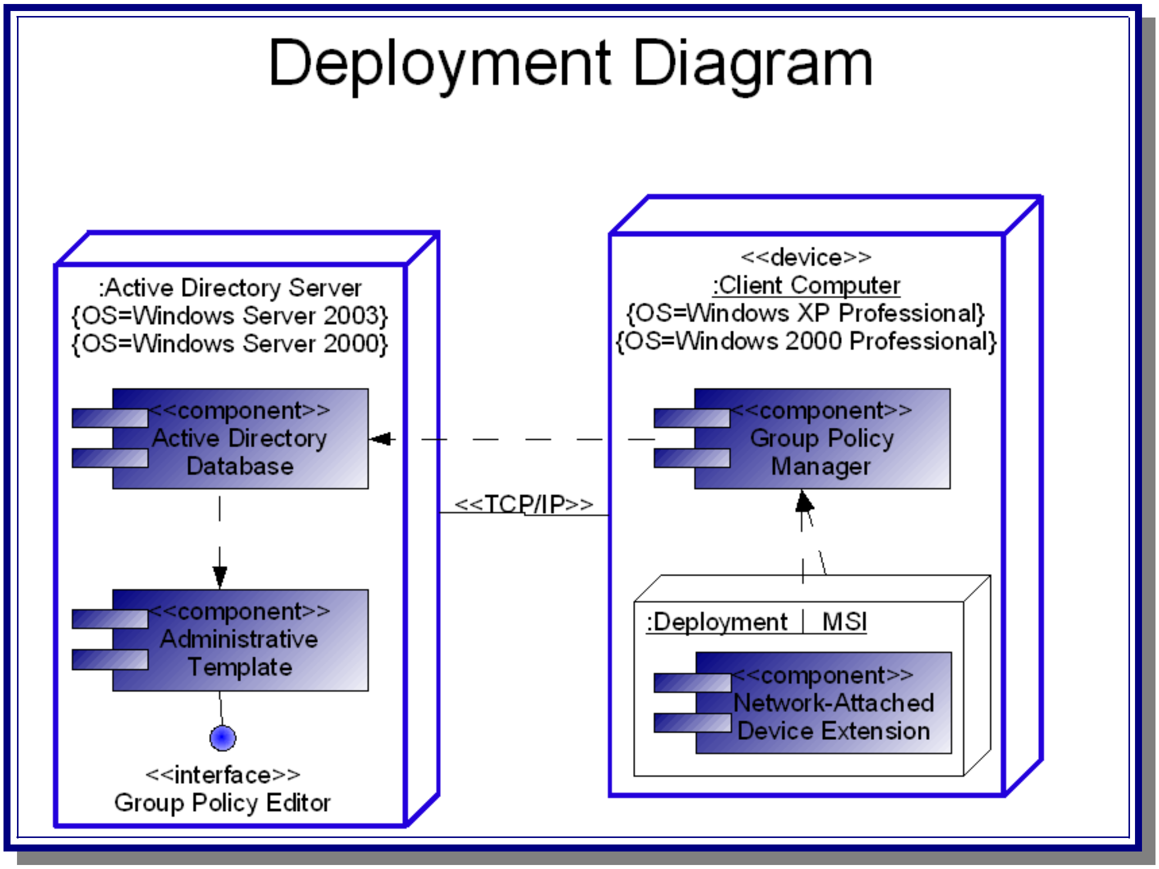


Figure 25: Deployment diagram depicting the run-time distribution of extension components and the communication model between them

Template are distributed via Active Directory. These settings are then transmitted to clients via TCP/IP, to the Group Policy management component that is located on the client machines. This component retrieves and synchronizes Group Policy information, including that of the extension, and executes the appropriate registered extensions, providing them with the changed and deleted policies. In the end, the modules interact to distribute specific information to clients, ultimately allowing the management of network-attached devices in a full, robust manner.

5.4 Sequence Diagram

The sequence diagram in Figure 26 shows the sequence of events that is involved in the management of network-attached and network-capable devices. Since the mechanisms are embedded within the operating systems of both the server and the clients, the calls to the remote components are standardized and largely internal. As such, the calls given are not the precise names used by the system, but rather names that reflect the general action that is carried out. The point of the diagram is not to give design information, but to convey the sequence of temporal and triggering events that cause system changes and, ultimately, the propagation of changes.

5.5 Class Diagram

The diagram presented in Figure 27 shows the classes required for implementation of the prototype network device management system. Since this component will be designed to manage printers and network storage, it must have classes for each. Note that all the classes in the implementation will be static – no instance variables are required since all required information is stored in the registry. Note that, in the illustration, the component classes are all utilized by the core Network Device Management class, which constitutes the central component of the client-side extension that will be utilized.

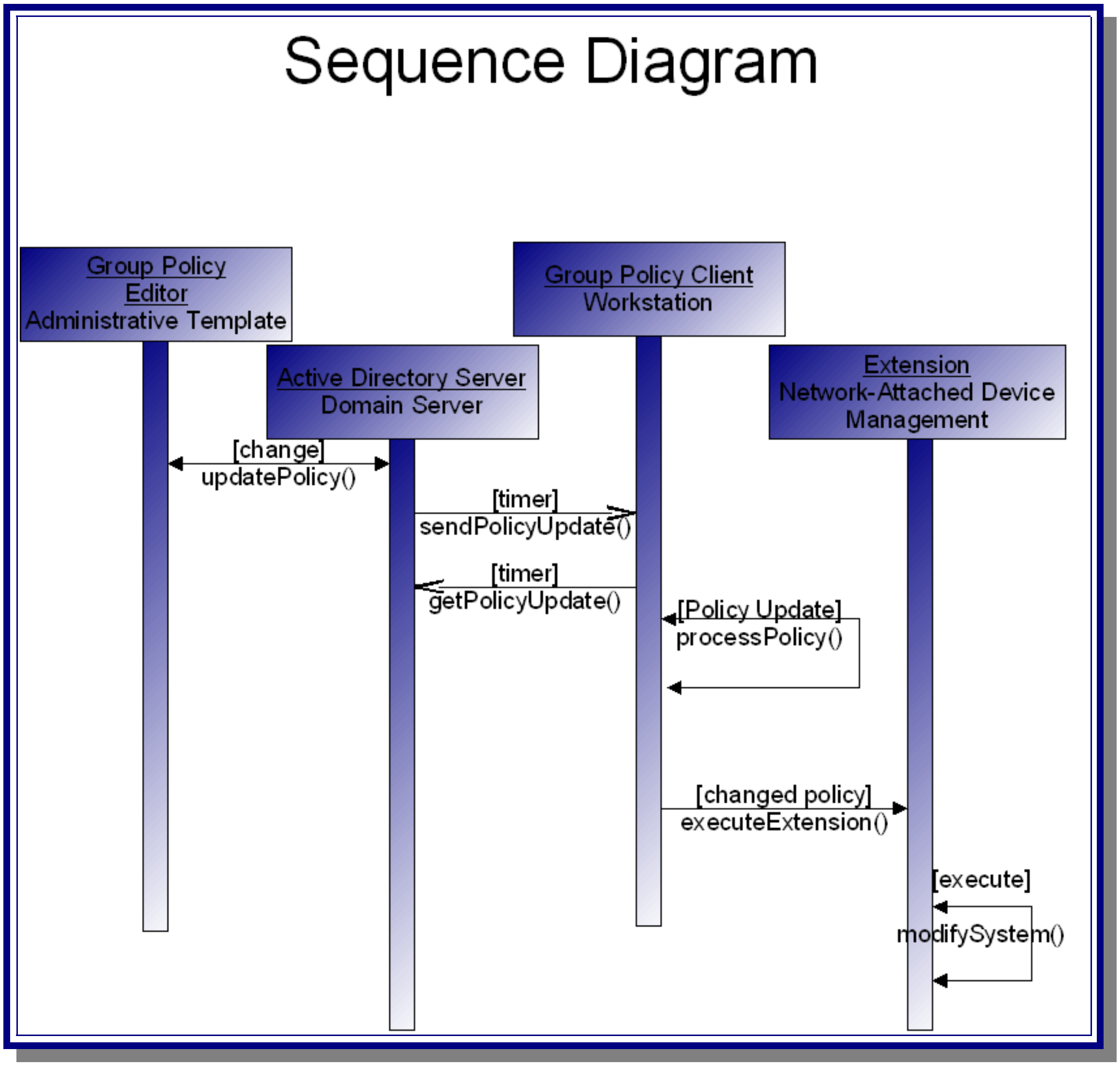


Figure 26: Sequence diagram showing the sequence of events and triggers responsible for the propagation of settings that allow the management of network-attached devices

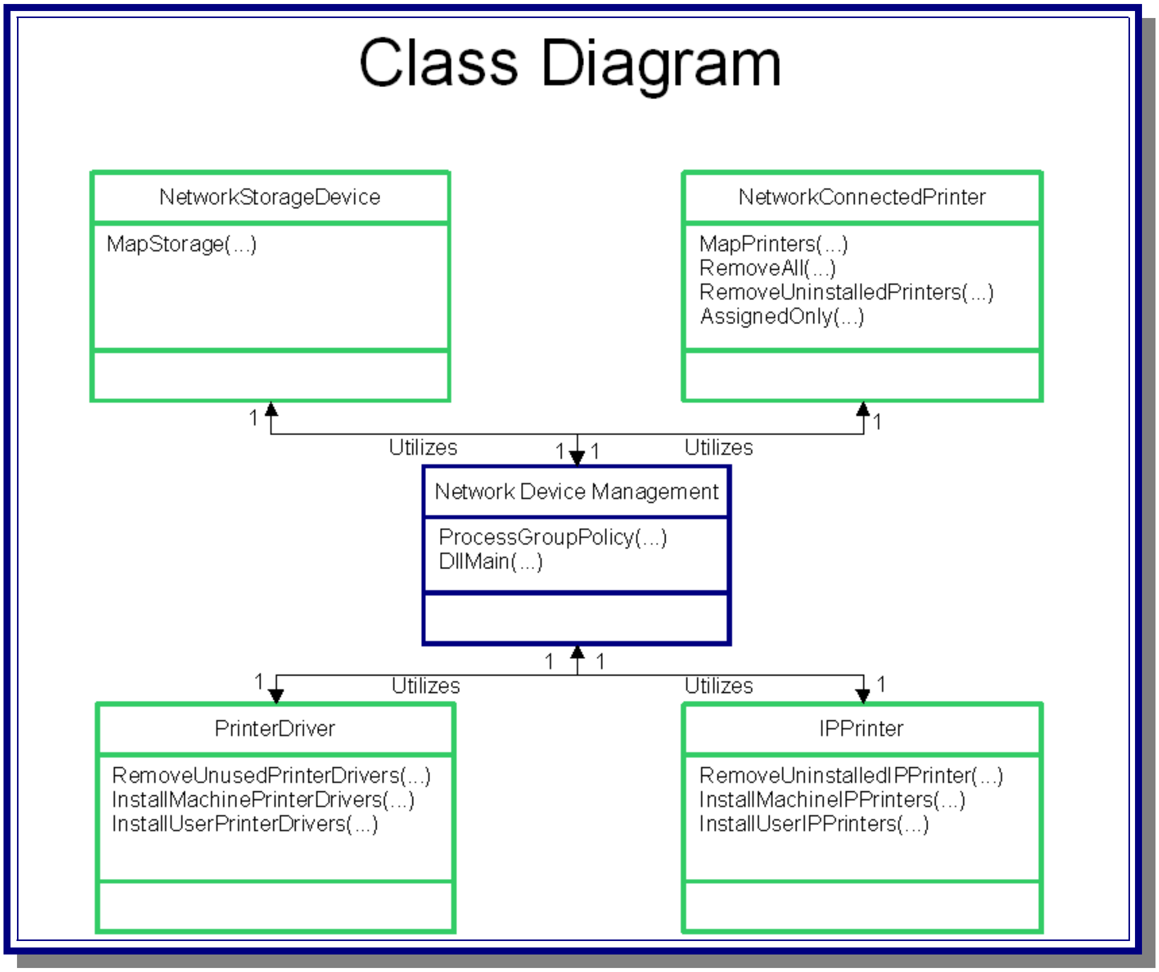

Class Diagram
NetworkStorageDevice
MapStorage(...)
NetworkConnectedPrinter
MapPrinters(...)
RemoveAll(...)
RemoveUninstalledPrinters(...)
AssignedOnly(...)
1
Utilizes
1
1
Utilizes
1
Network Device Management
ProcessGroupPolicy(...)
DllMain(...)
1
1
1
Utilizes
Utilizes
1
PrinterDriver
RemoveUnusedPrinterDrivers(...)
InstallMachinePrinterDrivers(...)
InstallUserPrinterDrivers(...)
IPPrinter
RemoveUninstalledIPPrinter(...)
InstallMachineIPPrinters(...)
InstallUserIPPrinters(...)

Figure 27: Class diagram for the prototype client-side extension showing static classes

6 Prototype: Network Device Management Group Policy Extension

In order that the advantages of network-attached devices were researched appropriately, a prototype was developed that allows system administrators to centrally administer the use of network-attached devices by client machines. The prototype is designed to illustrate the following:

1. The administration of network-attached devices can be as simplified and useful as – if not more so than – that of network-connected devices.
2. The use of network-attached devices decreases resource consumption.
3. Modern network clients do not require devices to be hosted on a server in order that their resources be shared effectively and efficiently.
4. The use of network-attached devices in their native state reduces network complexity, simplifying administration and troubleshooting tasks.
5. Network-attached devices require significantly less network bandwidth than network-connected devices. This makes installation of network-attached printers under their native paradigm extremely attractive.

To this end, the Network Device Management Group Policy Extension was created to address and illustrate these points. Using the Microsoft Group Policy centralized control mechanism, client machines on a network can receive network device assignments, drivers, and maintenance commands using the standard Group Policy

Object Editor. Although the extension only supports printers and storage devices (as the extension is meant to show proof-of-concept, not implement all features at this time), it serves to illustrate that network-attached devices can be configured just as easily as network-attached devices, but offer performance and maintenance benefits.

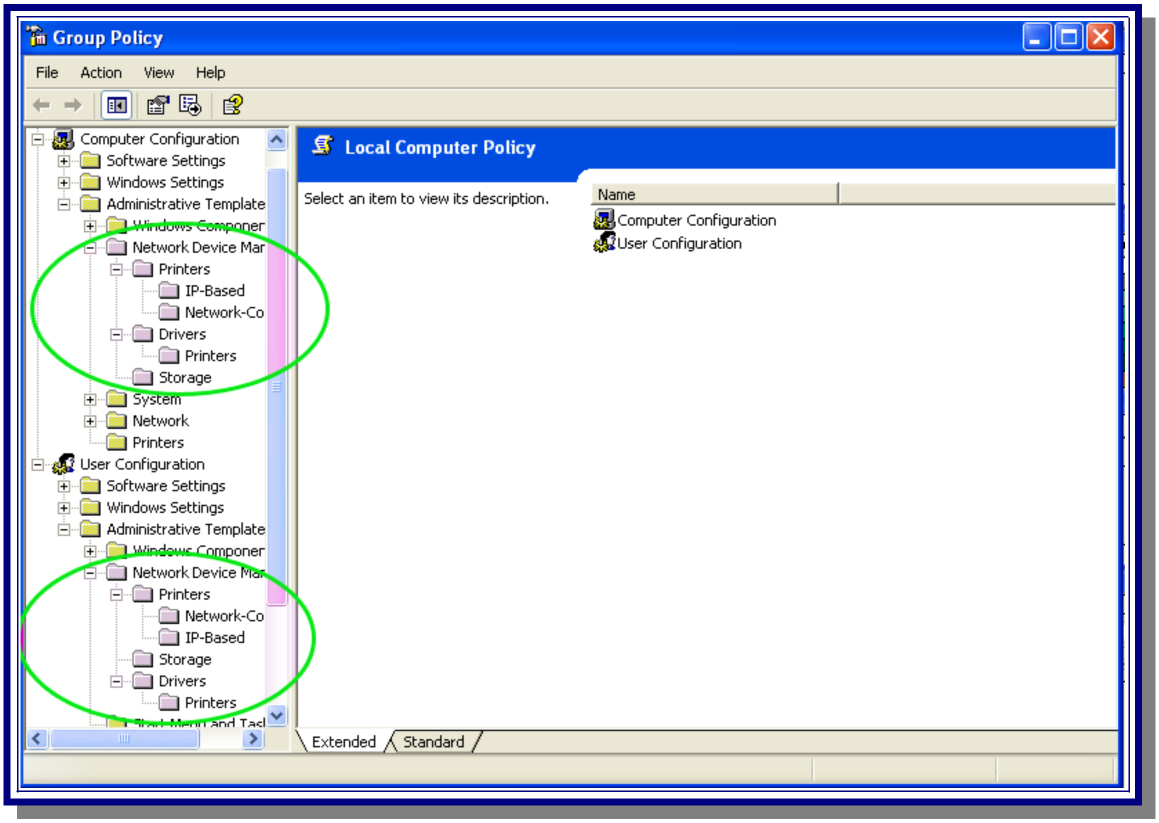
The remainder of this chapter illustrates the interface and activities offered by the extension, while the following chapters present an analysis of the extension and a comparison with other similar tools. The conclusion will then summarize the results, stating the degree to which the implementation goals were met.

6.1 User Interface

The user interface for the extension is neither customized nor distributed with the extension. Rather, it is provided via a set of policies that are configured by the Group Policy Object Editor console, which is a standard component of Microsoft Windows operating systems subsequent to, and including, Windows 2000 Professional. Because this extension is designed to eliminate user interaction, the configuration of the extension via a server is the only interface available. No errors are displayed on the client machines; no configuration options are present. As such, the following screenshots illustrate the configuration options made available to the system administrator via this configuration console.

The configuration of the extension takes place using the Group Policy Object Editor. The MSI distributable contains administrative templates that can be added to the editor, allowing system administrators to configure the extension from a server. Once installed and linked to a Group Policy object, the Network Device Management options shown in Figure 28 become available to the administrator.

These options, as shown, currently include categories for network-attached printer connections, IP-based printer connections, driver installation (for IP-based printers), and network storage connections. Policies are replicated for both the Computer Configuration and User Configuration categories, meaning that device management can be specified for either users, computers, or both. Note that these categories will likely be expanded as new devices are added to the extension. However, in order to minimize confusion as features are added, each policy setting has an extensive explanation which is displayed in the middle column of the screen.

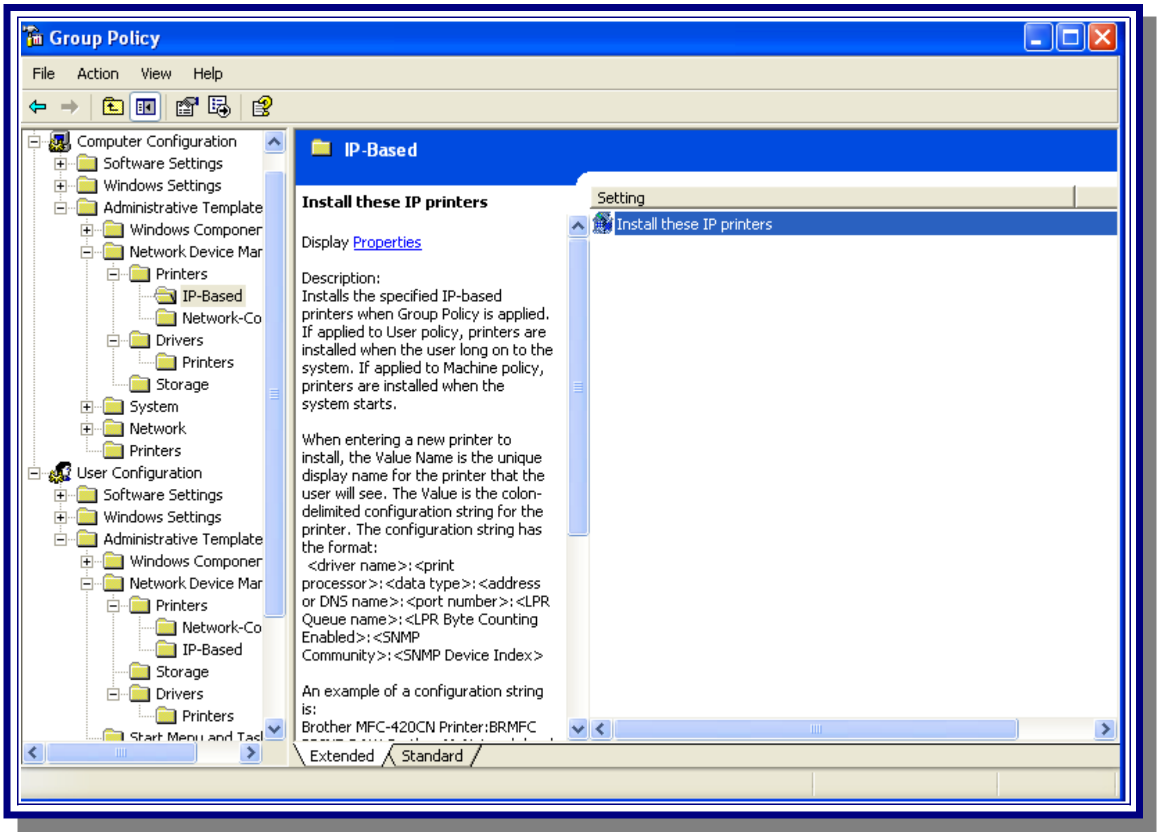


Select an item to view its description.

Figure 28: Group Policy categories added by the extension

Focusing on the individual categories shown in Figure 28, the *Printers* subcategory contains entries for both IP-based and network-connected printer management. The *IP-Based* subcategory contains the entry *Install these IP printers*, as shown in Figure 29. This allows administrators to specify the IP-based printers, as well as their configuration details, that will be deployed via Group Policy and installed via the extension on the client machines.

Inspecting the *Printers* subcategory once again and selecting the *Network-Connected* subcategory yields two options, shown in Figure 30. The first is the option *Install these network-connected printers*, which performs the same function as the equivalent entry in the IP-based printer category – that is, installing specified printer connections to printers that are managed by a server.



Install these IP printers

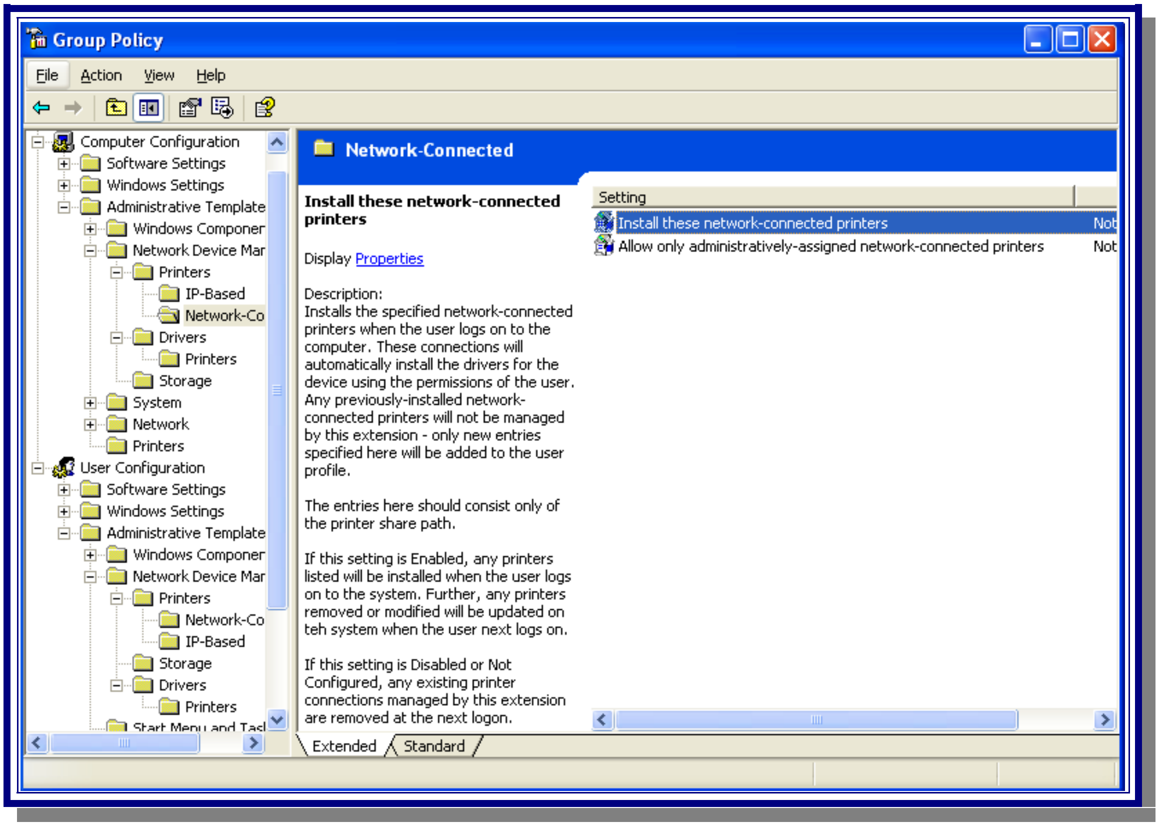
Display Properties

Description:
Installs the specified IP-based printers when Group Policy is applied. If applied to User policy, printers are installed when the user long on to the system. If applied to Machine policy, printers are installed when the system starts.

When entering a new printer to install, the Value Name is the unique display name for the printer that the user will see. The Value is the colon-delimited configuration string for the printer. The configuration string has the format:
<driver name>:<print processor>:<data type>:<address or DNS name>:<port number>:<LPR Queue name>:<LPR Byte Counting Enabled>:<SNMP Community>:<SNMP Device Index>

An example of a configuration string is:
Brother MFC-420CN Printer:BRMFC

Figure 29: IP-Based Printer configuration category



Install these network-connected printers

Display Properties

Description:
Installs the specified network-connected printers when the user logs on to the computer. These connections will automatically install the drivers for the device using the permissions of the user. Any previously-installed network-connected printers will not be managed by this extension - only new entries specified here will be added to the user profile.

The entries here should consist only of the printer share path.

If this setting is Enabled, any printers listed will be installed when the user logs on to the system. Further, any printers removed or modified will be updated on teh system when the user next logs on.

If this setting is Disabled or Not Configured, any existing printer connections managed by this extension are removed at the next logon.

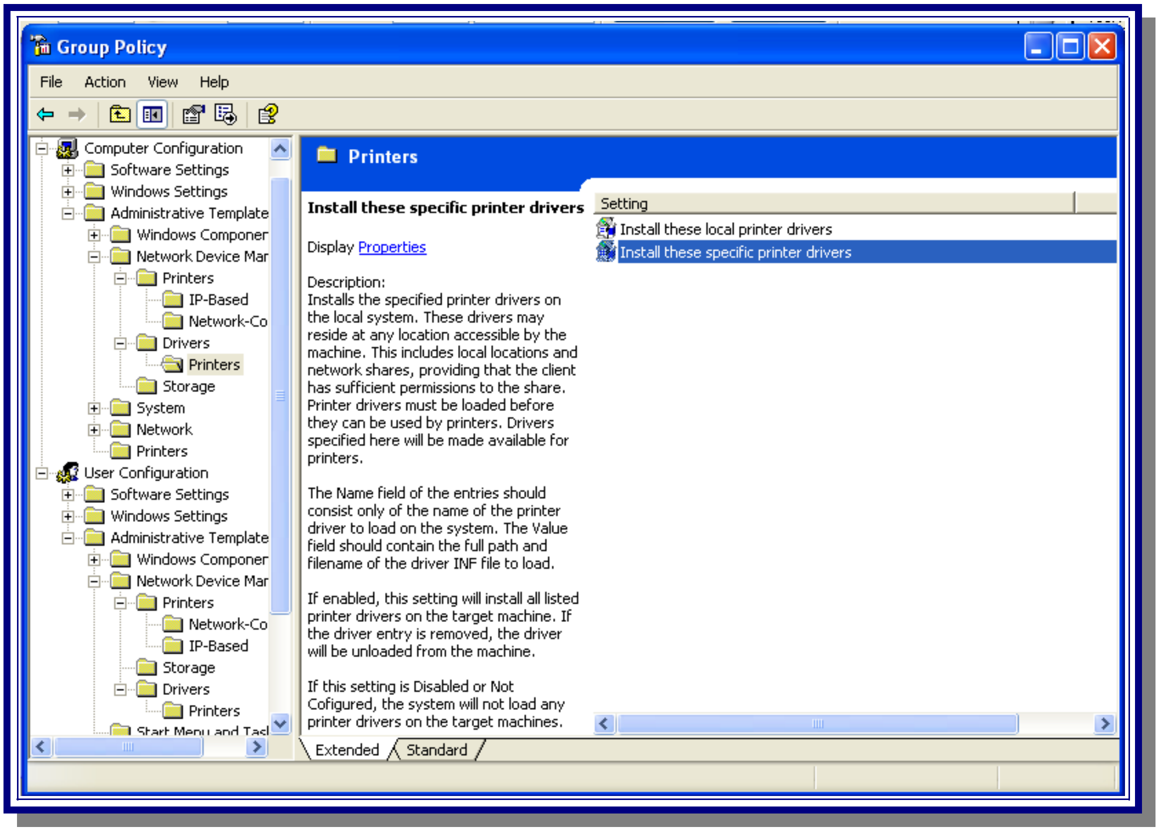
Setting	
Install these network-connected printers	Not
Allow only administratively-assigned network-connected printers	Not

Figure 30: Network-connected printer configuration options

These particular connections are much simpler to configure since a server manages the driver installation and resource management of the printer. This feature was provided in order that system administrators have an integrated system for managing network resources, though the focus is on network-attached devices. The second option, listed as *Allow only administratively-assigned network-connected printers* is a feature provided to control the deployment of the extension. If this policy option is enabled, all existing network-connected printers for a user are removed when they log on to the system. After they are removed, the network-connected printers that have been assigned via this extension are established. This effectively prevents users from creating a permanent connection to a network-connected printer to which they should not have access.

The *Driver* subcategory contains options for installing drivers for managed network-attached devices. Since network-attached devices are not controlled by a server, they require the distribution of appropriate drivers to client machines. Further, for an IP-based printer to utilize a driver – even if it is physically located on the system – the driver must be loaded into memory before it can be utilized. At present, the extension only supports network-attached printers, therefore the listing here only applies to printer drivers. As shown in Figure 31, the category contains two entries: *Install these local printer drivers* and *Install these specific printer drivers*. The first policy setting allows administrators to specify which drivers from the local, pre-installed driver store are loaded for use by printers. These drivers were distributed with the Windows operating system and are listed in the file "*ntprint.inf*". In this section, specified pre-installed printer drivers are loaded into memory for use by IP-based printers.

The second option – *Install these specific printer drivers* – allows administrators to specify a custom, OEM, or third-party driver to load into memory. This driver may reside anywhere on the local computer: drivers residing on a network location are not currently



Install these specific printer drivers

Display Properties

Description:
Installs the specified printer drivers on the local system. These drivers may reside at any location accessible by the machine. This includes local locations and network shares, providing that the client has sufficient permissions to the share. Printer drivers must be loaded before they can be used by printers. Drivers specified here will be made available for printers.

The Name field of the entries should consist only of the name of the printer driver to load on the system. The Value field should contain the full path and filename of the driver INF file to load.

If enabled, this setting will install all listed printer drivers on the target machine. If the driver entry is removed, the driver will be unloaded from the machine.

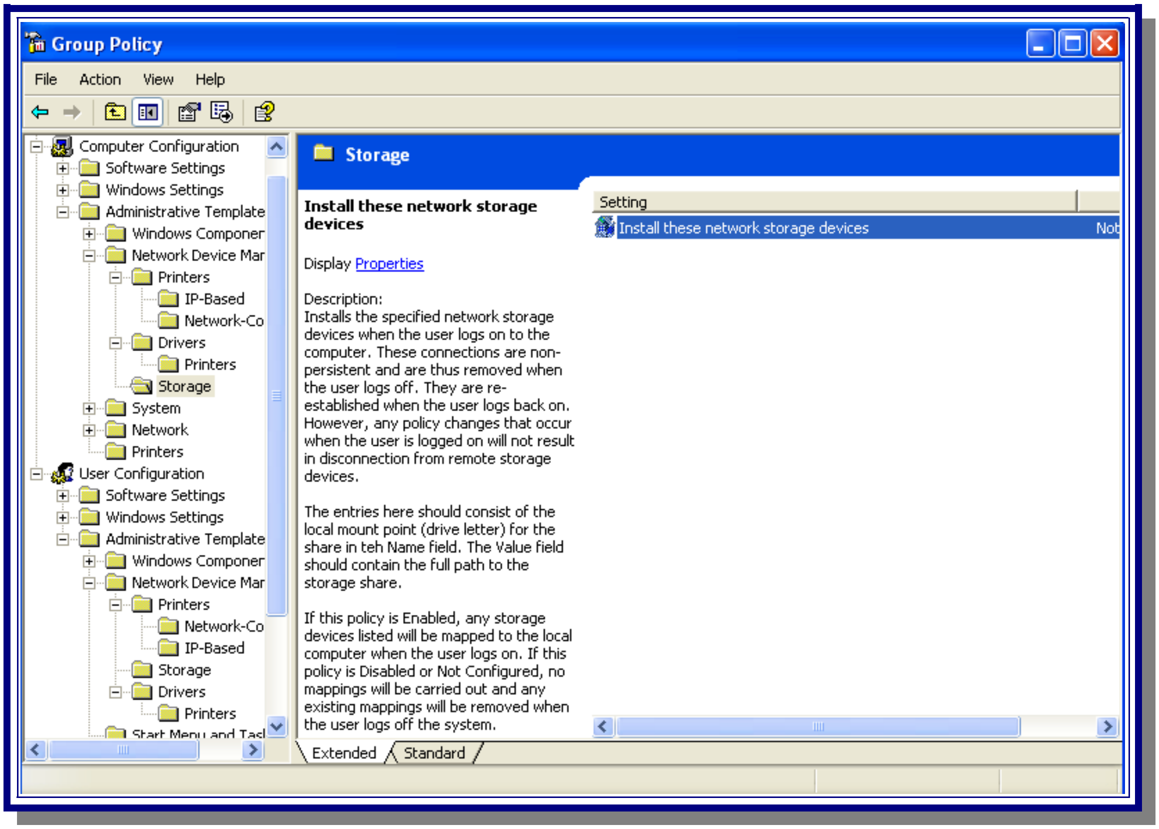
If this setting is Disabled or Not Cofigured, the system will not load any printer drivers on the target machines.

Install these local printer drivers
Install these specific printer drivers

Figure 31: IP-based printer driver installation configuration options

supported, though future modifications to the driver installation routines may add this functionality. The present paradigm entails the distribution of printer drivers using MSI packages via Group Policy, meaning that system administrators will have complete flexibility in the version and types of drivers installed on the systems they control. In specifying the specific driver they wish to use, administrators need provide the name of the driver and the local path to the INF file containing the driver. The rest is handled by the extension.

The final category listing is for *Storage*. Since network-attached and network-connected storage is accessed in the same manner (via standard protocols), the extension provides administrators a way to specify how users see the network resources. As Figure 32 shows, the policy *Install these network storage devices* is available.



Install these network storage devices

Display Properties

Description:
Installs the specified network storage devices when the user logs on to the computer. These connections are non-persistent and are thus removed when the user logs off. They are re-established when the user logs back on. However, any policy changes that occur when the user is logged on will not result in disconnection from remote storage devices.

The entries here should consist of the local mount point (drive letter) for the share in teh Name field. The Value field should contain the full path to the storage share.

If this policy is Enabled, any storage devices listed will be mapped to the local computer when the user logs on. If this policy is Disabled or Not Configured, no mappings will be carried out and any existing mappings will be removed when the user logs off the system.

Figure 32: Network-based storage configuration options

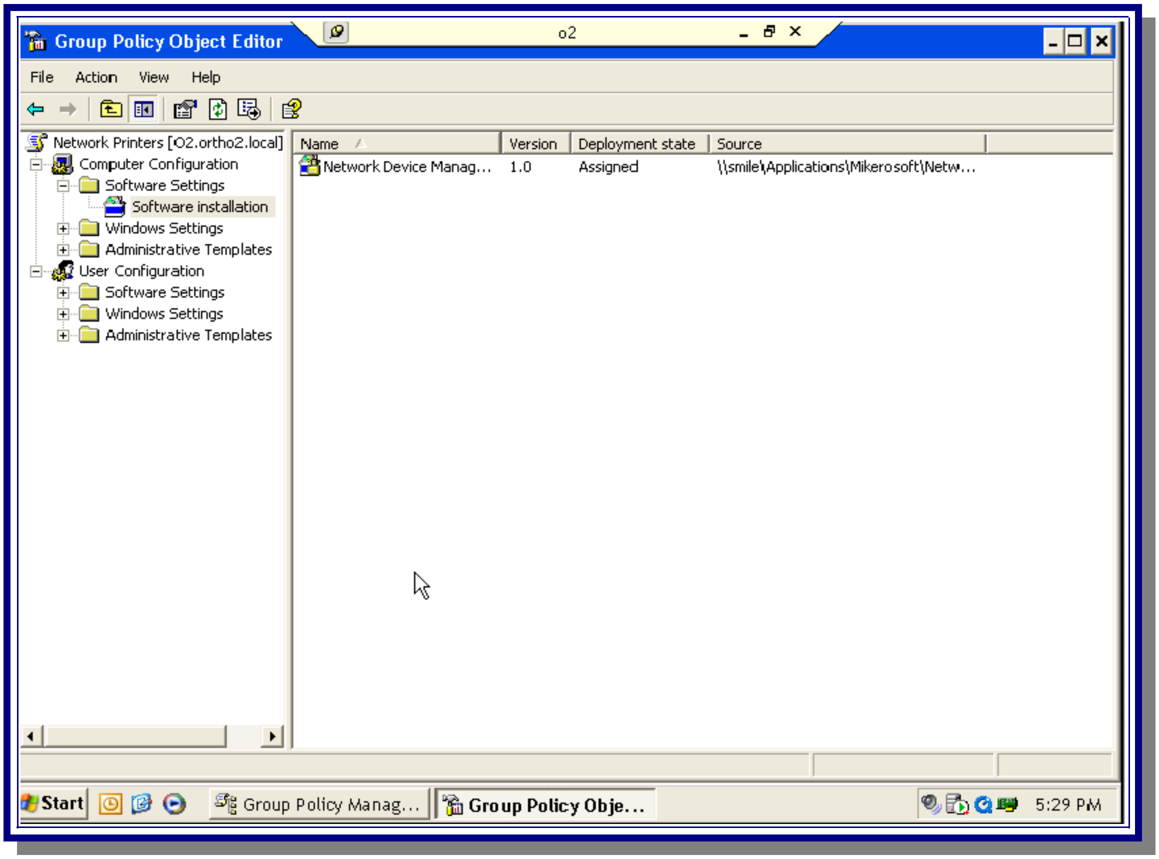
This policy allows administrators to specify the local mount point (for example, “*F:*” or “*L:*”) and the network share path (e.g. “*\\server\driveShare*”) for the storage device they wish mapped to the client computer. This gives administrators complete control over the drive mappings, and thus the network storage device access, their users possess.

6.2 Installation

The installation of the Network Device Management Group Policy Extension has been designed to make network deployment as easy as possible. The extension is distributed as a Microsoft Installer (MSI) file containing all the registry entries, files, and templates required to mange or install both the client-side extension and the server management components.

6.2.1 Client Machines

In order to install the extension on client machines, administrators need simply add the MSI to a Group Policy Object as a software package for installation, as shown in Figure 33. When the client machines reboot, the MSI is installed. No additional configuration is necessary as the MSI has been configured to perform a client installation unless instructed to do otherwise.



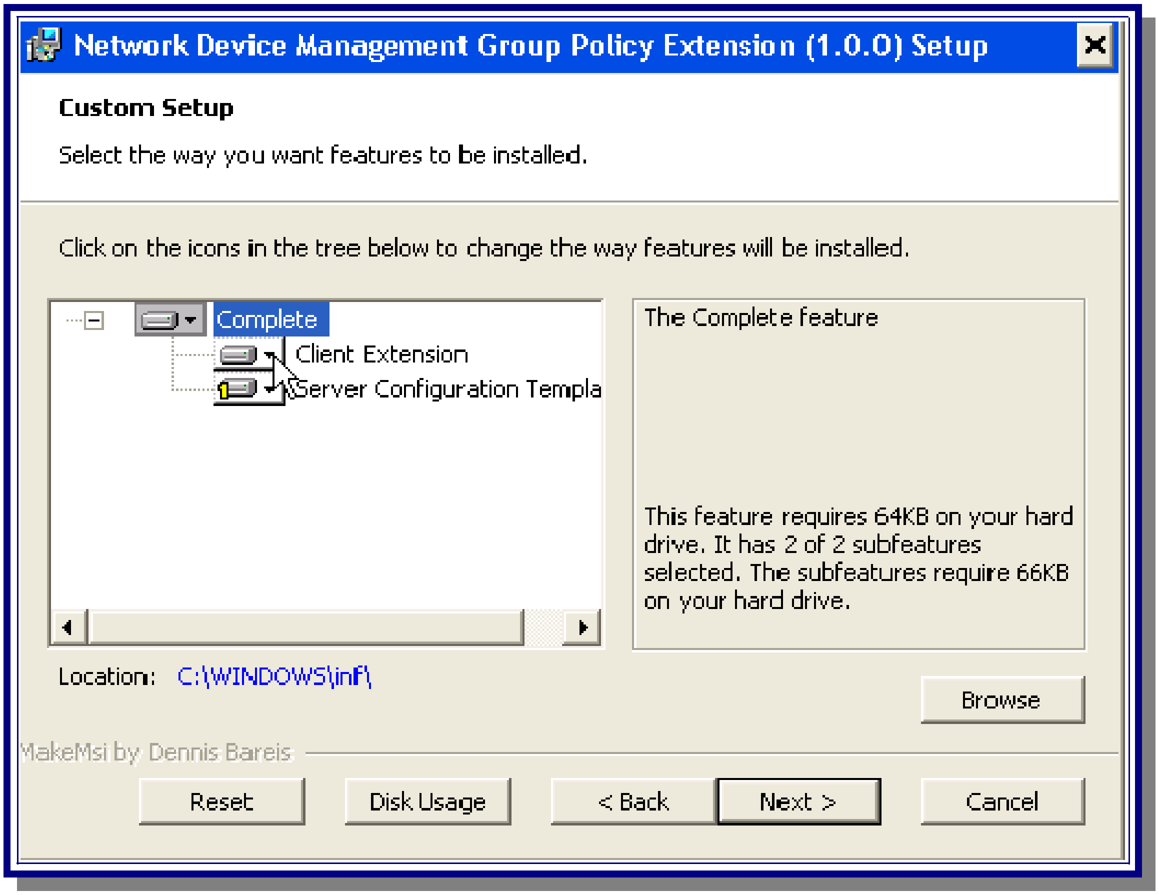
Name	Version	Deployment state	Source
Network Device Manag...	1.0	Assigned	\\smile\Applications\Mikerosoft\Netw...

Figure 33: Deploying the extension to clients via a Group Policy Object

6.2.2 Servers

As mentioned previously, the MSI package contains both the client and server files associated with the extension. In order to control the extension from an Active Directory server, administrators must install the administrative template for the extension on the server. This can be accomplished by executing the MSI and performing a custom installation when prompted. This customization screen gives the user the option of installing the administrative template and/or the extension, as shown in Figure 34.

Further, the installation path may be customized, though the default directory is the



Network Device Management Group Policy Extension (1.0.0) Setup

Custom Setup

Select the way you want features to be installed.

Click on the icons in the tree below to change the way features will be installed.

The Complete feature

This feature requires 64KB on your hard drive. It has 2 of 2 subfeatures selected. The subfeatures require 66KB on your hard drive.

Location: C:\WINDOWS\inf\

Figure 34: Installation of administrative templates via custom MSI installation options

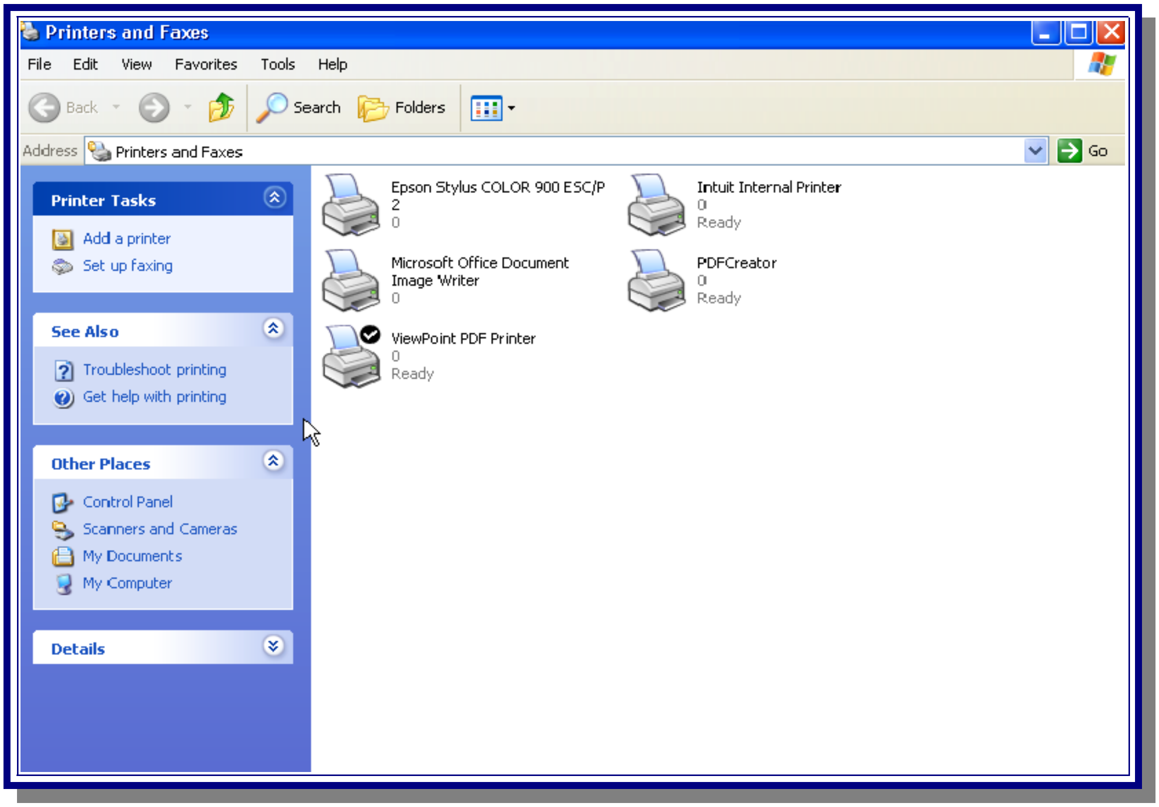
standard location for administrative templates. After finishing the installation, administrators can add the template to their Group Policy Objects as they normally would and configure the extension.

6.3 Configuration Example

While a description of the system and its policy options is often sufficient, obtaining a greater understanding of the extension and its abilities is facilitated via an example. An example configuration will be shown, illustrating the client in its initial state, the configuration changes that take place on the server, and the result on the client.

6.3.1 Initial Client State

The initial client state is one in which only local printers are installed. These printers include physically attached to the system and those installed by applications. The initial printer listing is shown in Figure 35. Note that the installation of the Network Device Management Group Policy Extension is irrelevant at this point: without configuration data, the extension performs no actions on the system.



Epson Stylus COLOR 900 ESC/P 2
0

Intuit Internal Printer
0
Ready

Microsoft Office Document Image Writer
0

PDFCreator
0
Ready

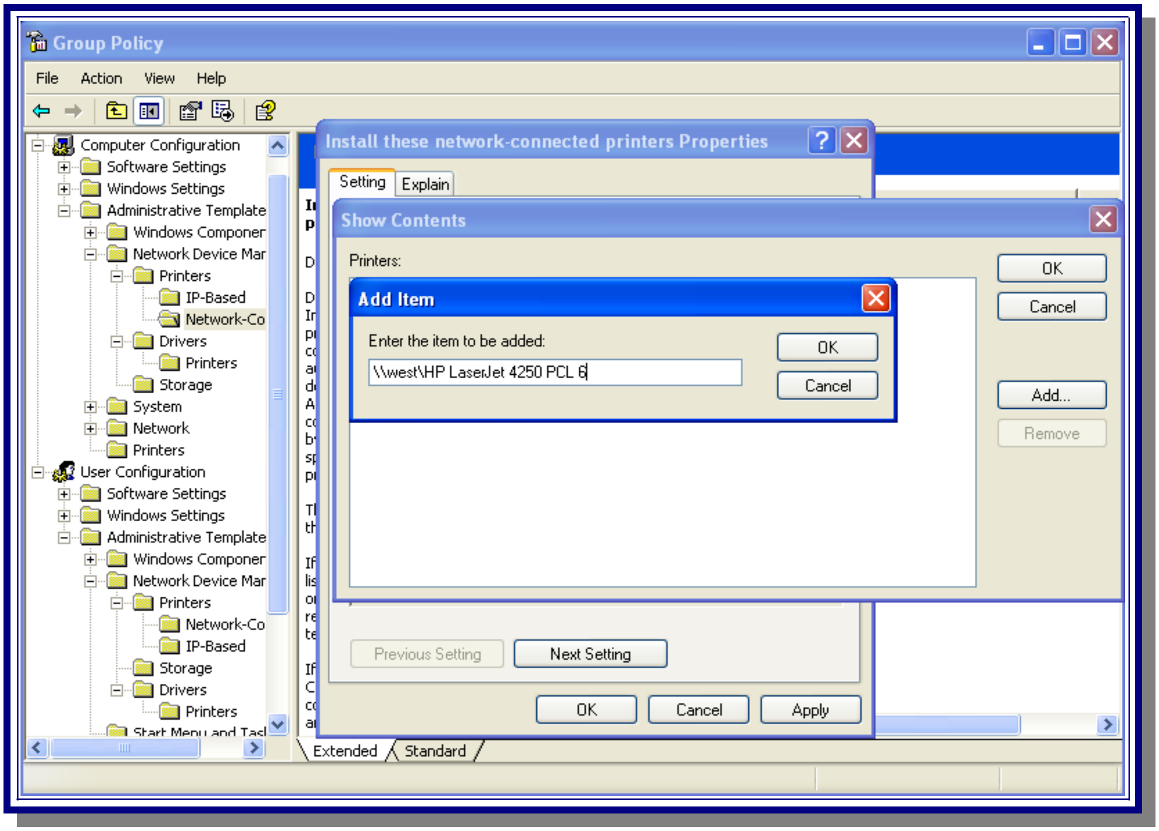
ViewPoint PDF Printer
0
Ready

Figure 35: Initial client printer listing

6.3.2 Server Configuration

The configuration options specified on the server determine the resulting set of printers on the client. In this example, a network-attached printer, and storage connection, and an IP-based printer (with corresponding driver) will be configured for installation on the client systems. For the sake of brevity, the administrative templates for the extension have already been installed on the server and linked to an active Group Policy Object.

The first step is then the addition of the network-connected printer. Since this is to be applied as a part of the machine policy, the corresponding listing under Machine Configuration is activated and configured as shown in Figure 36. Here, the only required



Enter the item to be added:

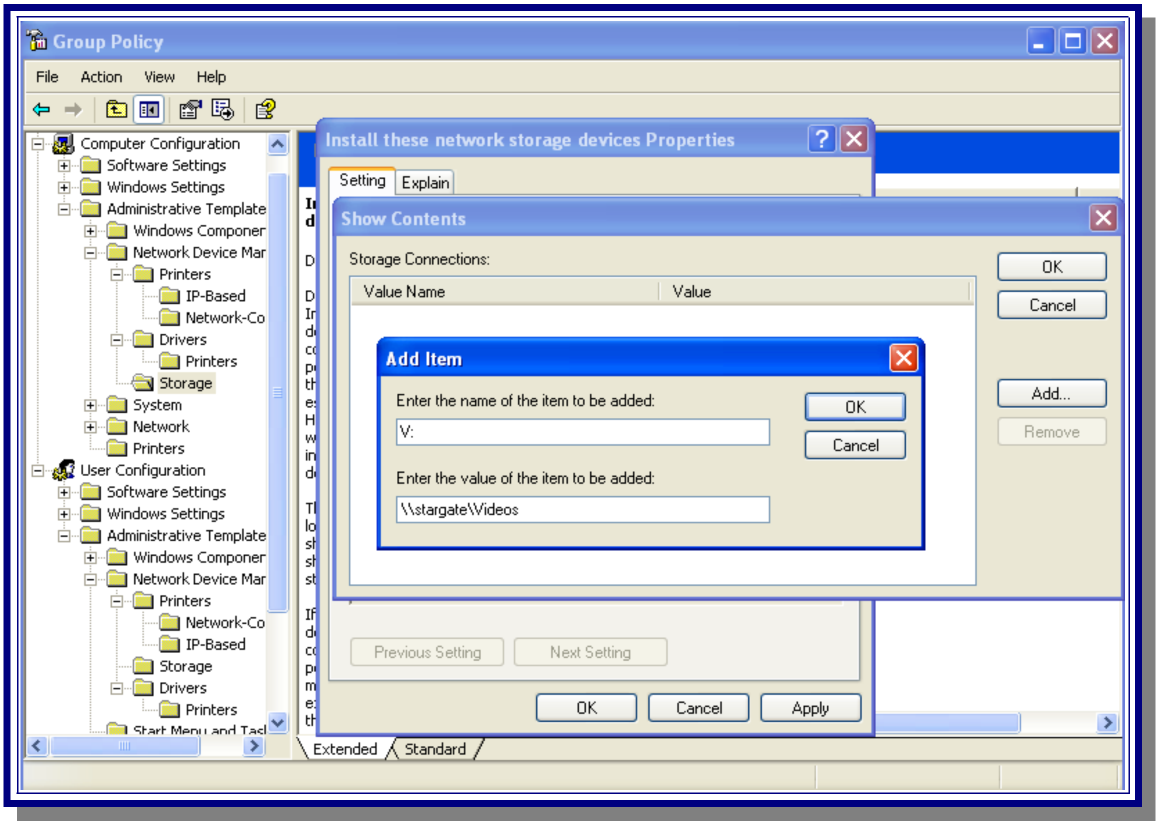
\\west\HP LaserJet 4250 PCL 6

Figure 36: Addition of a network-connected printer under Machine Configuration

field is the full share name and path of the printer; since the printer is managed by a server, driver installation will take place automatically.

The next step in the successful deployment of the example policy is the addition of the storage mapping. In a manner similar to that of the network-connected printer, the network storage device information is added via the appropriate policy object. Again, this mapping should be available to all users of the computer, so it is placed in the *Storage* subcategory under the *Machine Configuration* heading. The required fields here are the mount point - "*V:*" in this case – and the path to the share - "\\Stargate\Videos" here. This configuration is shown in Figure 37.

The third step is the addition of the IP-based printer and its associated driver. Again, the printer should be available to all users on a computer so it is configured under



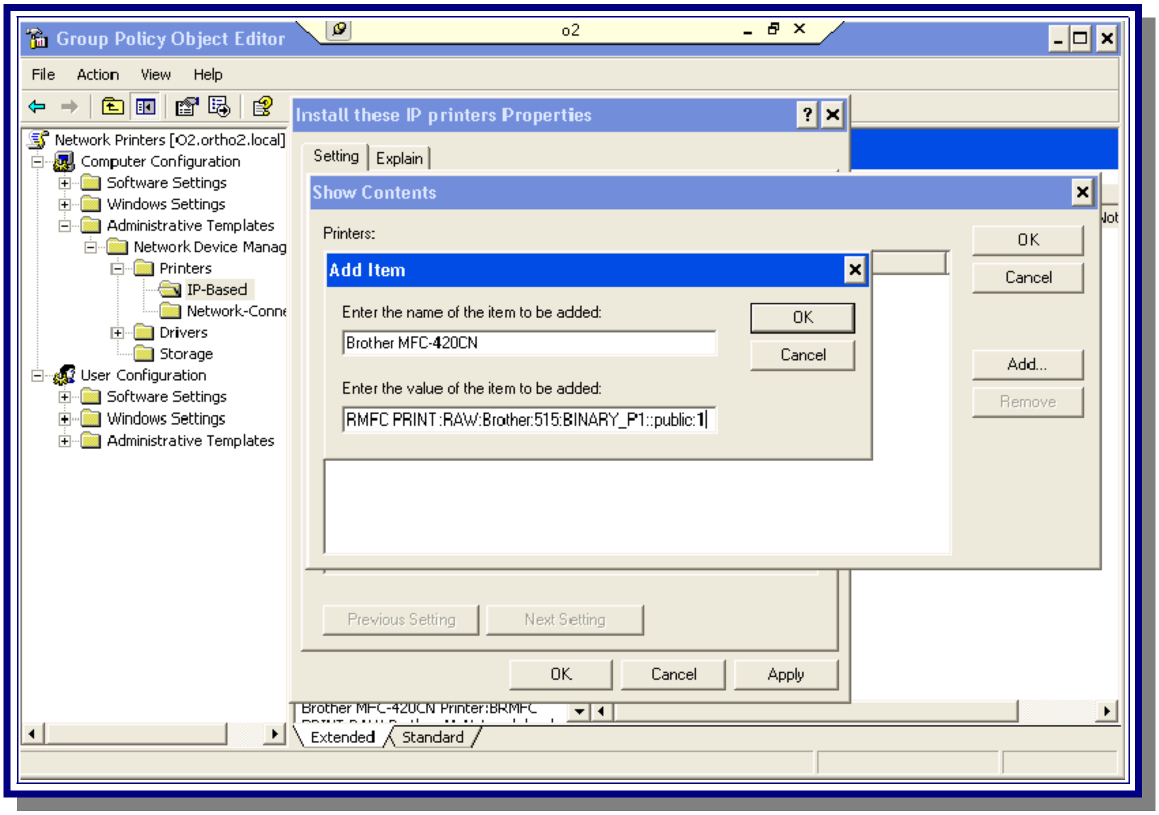
Enter the name of the item to be added: V:

Enter the value of the item to be added: \\stargate\Videos

Figure 37: Addition of a network storage device under Machine Configuration

the *Machine Configuration* category. Enabling the *Install these IP printers* policy and selecting the *Show...* option to add a mapping, the user is presented with the dialog shown in Figure 38. Here, the name should be the display name of the printer; the value should be the configuration string for the printer, as described in the description for this policy (a colon-delimited string of values). In this case, a Brother MFC-420CN network-attached printer is being installed.

In order for the IP-based printer to function properly, the driver must be installed on the client system. For the purposes of this example, it has already been installed on all client machines – using an MSI via a Group Policy Object – to the local location "*%ProgramFiles%\Brother\Driver\MFC-420CN\USB*" where the INF is named "*brprbh34.inf*." Therefore, the driver policy is enabled and a new entry is selected. Here,



Add Item

Enter the name of the item to be added:

Brother MFC-420CN

Enter the value of the item to be added:

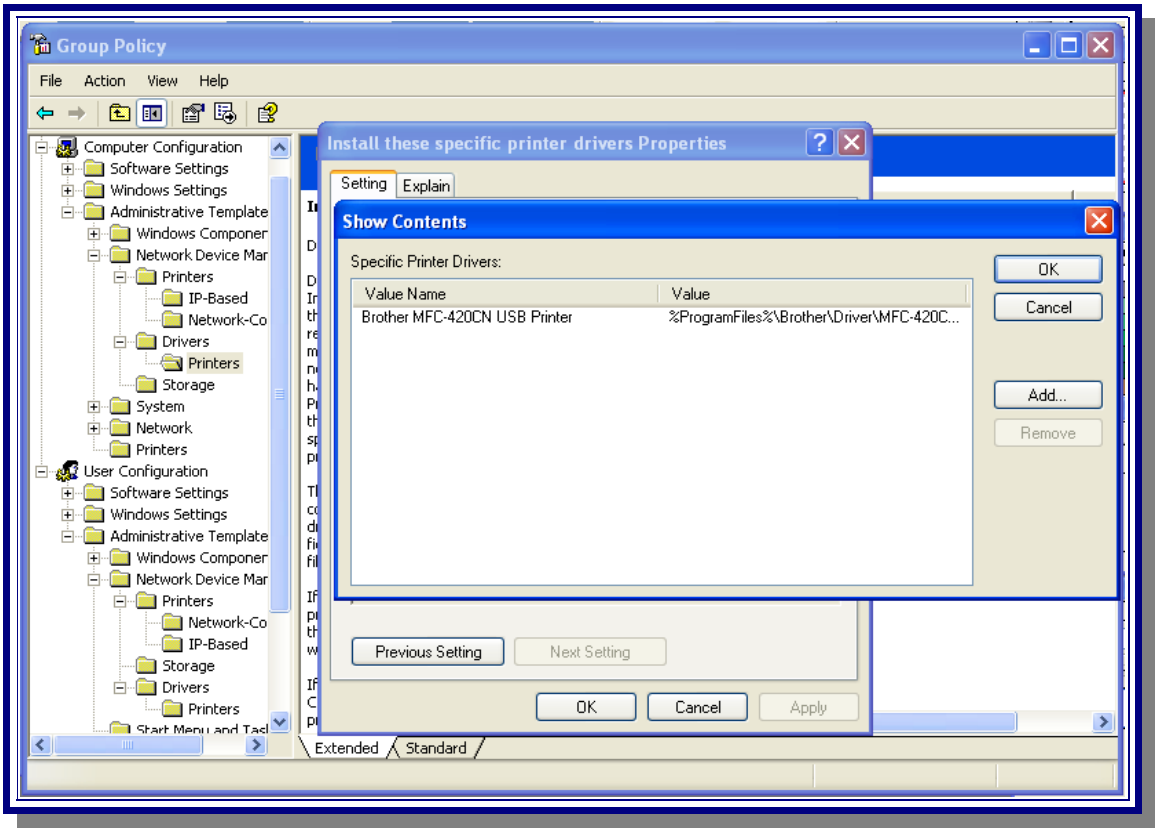
RMFC PRINT:RAW:Brother:515:BINARY_P1::public:1

OK | Cancel

Figure 38: Adding an IP-based printer under Machine Configuration

the name is the name of the driver and the value is the name of the INF file (including path) on the local machine. This configuration is shown in Figure 39.

It is important to note that the driver has been installed under the policy *Install these specific printer drivers* as the driver is a manufacturer driver that was not distributed as a part of the operating system. The extension differentiates between pre-installed and manufacturer-provided drivers in order to simplify the utilization of built-in Windows drivers. Pre-installed drivers are those listed in the "*ntprint.inf*" file on the local system. Windows contains a minimal set of drivers for each of these drivers in order to make device installation as easy as possible, and to provide immediate access to many devices. Utilizing one of these drivers is accomplished via the *Install these local printer drivers*, requiring only the name of the driver to load.



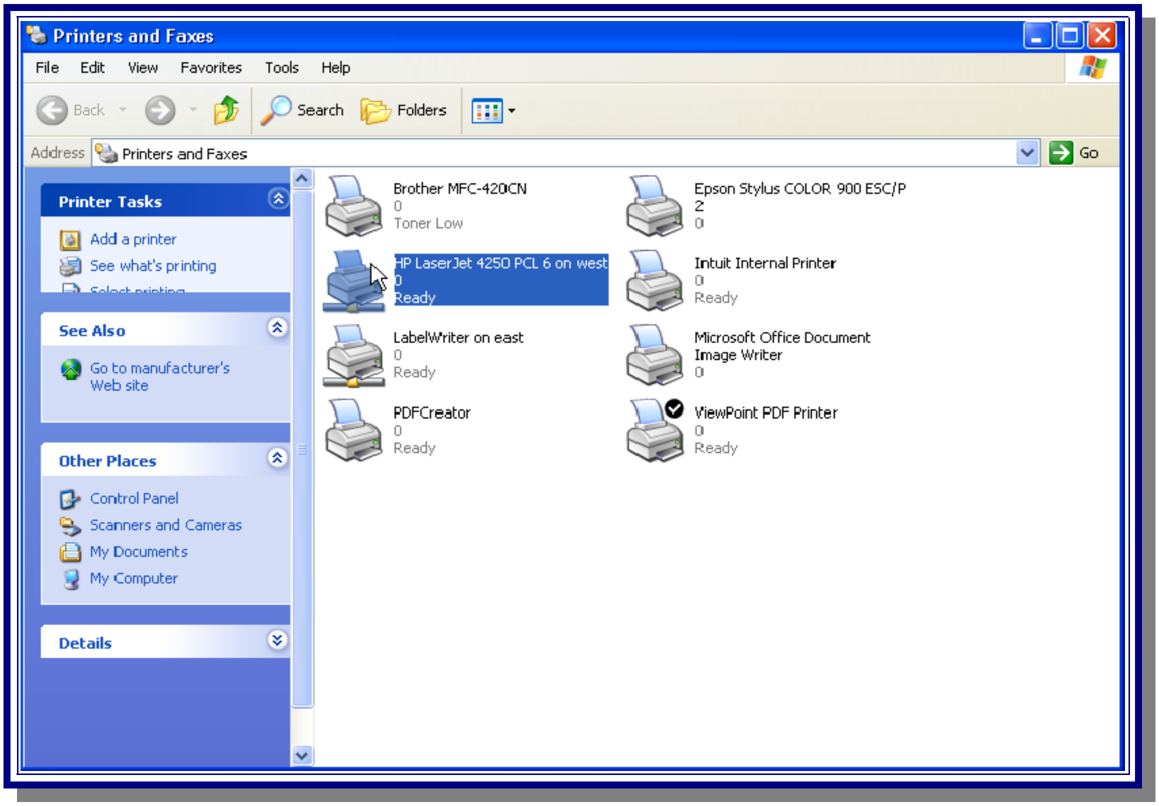
Specific Printer Drivers:

Value Name	Value
Brother MFC-420CN USB Printer	%ProgramFiles%\Brother\Driver\MFC-420C...

Figure 39: Addition of a specific printer driver under Machine Configuration

6.4 Effect on Clients

Now that policy has been configured and applied at the server level, the clients need only undergo a standard Group Policy refresh. These occur when the computer starts, when a user logs on, when forced, or after a randomized time interval. In any case, the result is the same: the printer drivers are loaded into memory (in the case of IP-based printers) and the printers are installed. The effect of the configuration changes made on the server is shown for a client machine in Figure 40. Note that all printers have been installed and, though not shown, the drive mappings have been successfully carried out as well.



Brother MFC-420CN
0
Toner Low

Epson Stylus COLOR 900 ESC/P
2
0

HP LaserJet 4250 PCL 6 on west
0
Ready

Intuit Internal Printer
0
Ready

LabelWriter on east
0
Ready

Microsoft Office Document Image Writer
0

PDFCreator
0
Ready

ViewPoint PDF Printer
0
Ready

Figure 40: Client printer listing after server configuration is applied via a Group Policy refresh

6.5 Network Effects

Since administrators now have a tool to allow them to manage network-attached devices (not just network-connected devices), they are free to migrate their printers and other network-attached devices to a native access mode (that is, network-attached: without a server to manage their data). The research done using this extension has demonstrated that the use of network-attached devices under their native paradigm, as compared to a network-connected paradigm, posed several significant improvements. The following sections detail these effects.

6.5.1 Network Topology

An effect brought on by the migration of network-connected printers to a network-attached state has been the simplification of the network topology. Instead of mapping traffic paths for printer data through servers, data can easily be viewed as reaching the printer directly without sacrificing detail. This makes troubleshooting and network analysis tasks significantly easier.

A further effect in this regard was the removal of servers from network traffic paths. That is, by eliminating a dependency of a printer on a server, the server can be eliminated in the network topology diagram. In some depictions, this would eliminate loops and more complex traffic paths.

6.5.2 Network Bandwidth

The bandwidth involved in the client submission of printer requests depends heavily on the access paradigm used. In the case of network-connected printers, the data path taken goes form client to server, then to the physically attached device. In the case of a network-attached printer operating in a network-connected mode, the data travels form the client to the server, then back across the network to the printer itself, which is attached to the network. However, the path taken for purely network-attached devices is direct from the client to the printer.

6.5.2.1 Test Configuration

In order to accurately examine the effects of implementing network-attached printers under their native paradigm, a test scenario was created. Here, a network-attached printer was configured in various ways on a network on which one client and a domain controller were present at all times.

The first configuration involved a network-attached printer configured for use by clients as a network-connected device. Here, traffic from clients to the printer traveled through an intermediary server. This intermediary server was activated and configured specifically for this task on the network.

In the second configuration, the printer was configured as a network-attached device wherein the client data traveled directly to it. The print server present in the first configuration was deactivated.

Measurements of network bandwidth usage and total data transferred were taken from the network switch used to connect the devices. The data, in its raw form, was the total amount of data transmitted across the network when print requests were made from the client to the printer. The print request itself consisted of a medium-sized file incorporating graphics and text.

6.5.2.2 Network-Connected Paradigm

Experimental results under the network-connected paradigm indicated that, ostensibly, when a server was used to manage data, the network traffic involved in submitting a 2.5 MB print job from a client to the printer was approximately 5.1 MB. This result reflects the retransmission of printer data from the client to the server, then from the server to the printer.

Although the expected bandwidth usage should have been twice the size of the print job, there was a small increase of the data transfer that was related to management data sent from the server back to the client, acknowledging the print job. These measurements do not take into account the management overhead involved in administering the printer. The assignment of the printer to the machine (by either the extension or manually) involved an additional 3.5MB of bandwidth (driver and extension data), but occurred only one time.

6.5.2.3 Network-Attached Paradigm

Experimental results using the network-connected paradigm indicated that the same 2.5 MB print job, submitted directly to the printer, required only 2.5 MB of bandwidth. However, in this case, the overhead of the extension that managed the printer installation must be taken into account for a fair comparison of network-attached and network-connected paradigms – it is not reflected in the previous measurement.

Using the same measurement techniques, the bandwidth required to administer and install the network-attached printer was measured. In this case, the driver and extension data consumed 6 MB of bandwidth which, under the current design, occurs only once. The increase in the size of the driver and extension data can here be attributed to the fact that the driver deployed to the client the complete device driver. When printers are used a network-connected devices, the printers hosting these devices do not transmit all driver files, simply those required to relay the print job to the server. Because the full driver must be deployed to clients under the network-attached paradigm, this bandwidth increase is unavoidable.

6.5.2.4 Comparison

Summarizing the results from the previous two sections, the use of a network-attached printer under a network-attached paradigm requires only 49.02% of the bandwidth necessary when the same printer is configured under the network-connected paradigm. This performance increase is extremely attractive: in even moderately busy printing environments, the release of 51.1% of the daily bandwidth requirement can have significant performance benefits for other network services.

When one takes into account the overhead required to manage the device installations, one sees that the administration of network-attached devices required 71.43% more bandwidth than the administration of network-connected devices. Again, this is the result of the deployment of the full printer driver under the network-attached device paradigm using the network switch data logging features; other tools may also be used to measure the performance of the extension [28], [29].

Combining the bandwidth results of this trial – including the management overhead with the print job data – one sees that the net bandwidth improvement was 1.16% for this single print job. As the number of print jobs increases, however the bandwidth required for management becomes insignificant. For example, if a user sends 200 MB of data for printing, the benefit is a bandwidth decrease of 48.95%.

Thus, the benefits become more significant in environments where the total amount of data sent for printing is greater than the overhead involved in printer administration. Since most organizations produce significant amounts of paper in a given day, the benefits of the network-attached device paradigm will become attractive for even small, yet busy, offices. The comparative decrease in the amount of network bandwidth consumed will then asymptotically approach the 51.1% experimental maximum as network print traffic increases. Such a decrease or minimization of bandwidth is critical to the proper development and maintenance of networks [30].

7 Comparison with Similar Tools

The Network Device Management Group Policy Extension is not the first client-side extension to allow administrators to control client device and printer settings. However, it is the one of the few mechanisms by which administrators can remotely install IP-based printers and their associated drivers. Three other tools of similar functionality are DesktopStandard Policy Maker Standard Edition [2], FullArmor IntelliPolicy [3], and Microsoft Print Management [4]. This chapter will analyze the features and effects of these tools against those of the Network Device Management Group Policy Extension.

7.1 Feature Comparison

In order to evaluate the utility and uniqueness of the Network Device Management Group Policy Extension, it is necessary to compare it against other packages that provide similar or comparable functionality. The main difference, however, is that the Network Device Management Group Policy Extension is meant to deal with *all* network-attached devices, not simply a small set. Further, it is meant to be extended to include more hardware devices whereas the others are closed, proprietary systems. Table I illustrates the key differences between the Network Device Management Group Policy Extension, [2], [3], and [4]. Green entries indicate the presence of a feature whereas red sections indicate the lack of a particular feature. Details on each product will appear in the following chapter sections.

Table I: Feature comparison between Network Device Management Group Policy Extension and similar tools

	Network Device Management Group Policy Extension	**DesktopStandard Policy Maker Standard Edition [2]**	**FullArmor IntelliPolicy [3]**	**Microsoft Print Management [4]**
Client-Side Extension	Yes	Yes	Yes	No
Map Network-Connected Printers	Yes	Yes	Yes	Yes
Map Storage Drives	Yes	Yes	Yes	No
Install IP-Based Printers	Yes	Yes	No	No
Install Printer Drivers	Yes	Yes	As Files	No
Update Printer Drivers	Yes	Yes	As Files	No
Extensible	Yes	No	No	No
Designed for Network-Attached Devices	Yes	No	No	No
Open Source	Yes	No	No	No
Advanced Group Policy Functions	No	Yes	Yes	No

7.2 Individual Analysis

The following information is an extended comparison of the similar management tools listed in the previous section. This is meant to clarify the comparison represented in Table I by describing the alternate product and its functionality.

7.2.1 DesktopStandard PolicyMaker Standard Edition

PolicyMaker Standard Edition [2] is a software package including 21 Group Policy extensions that extend the scope and granularity of standard Group Policy settings significantly. Not only can administrators control client desktops in much greater detail, but they can also install TCP/IP-based printers in addition to network-connected printers. Extended features include client file management, security enhancements, performance modifications, file modification, Microsoft Office integration, and many more settings.

In comparing this software to the functionality offered by the Network Device Management Group Policy Extension, the most significant feature they share is the installation of IP-based printers. Both systems allow for the installation and upgrade of printer drivers. A key difference in driver management is that PolicyMaker can utilize UNC share names for printer drivers, whereas the extension developed for this thesis requires the use of *managed* MSI-installed drivers to ensure clean driver management. While PolicyMaker allows a more convenient method of driver installation and management, that offered by this extension is more structured and organized. However, PolicyMaker is not geared toward the installation and management of network-attached devices, making it useless for the management of devices such as network-attached scanners. Though the current implementation of this extension also lacks support for network-attached scanners, future integration is both planned and possible – this is not the case for PolicyMaker.

In this comparison, the key advantage of the Network Device Management Group Policy Extension is that it is small and efficient at accomplishing its task. While it does not provide the broad range of features provided by PolicyMaker, it does add IP-based printer driver management within a framework that is meant to be extended for other network devices.

7.2.2 FullArmor IntelliPolicy

FullArmor IntelliPolicy [3] is a software package that, like PolicyMaker, adds a significant amount of functionality to the Microsoft Group Policy templates that ship with the operating system. Using this package, administrators can modify registry keys and values, map printers and drives, deploy files and folders, assign security settings, and create application access rights. Additionally, the software provides new, “intelligent” policies that are otherwise unavailable, giving administrators even more control over custom desktop settings.

Both software system provide network-connected device functionality with printer and drive mapping features. IntelliPolicy also provides a plethora of policy settings that are not addressed by the Network Device Management Group Policy Extension. The Network Device Management Group Policy Extension, however, is capable of adding IP-based printers and drivers. Though IntelliPolicy can theoretically distribute driver updates as files, it does not appear to have a mechanism by which they are installed. This gives the Network Device Management Group Policy Extension an additional advantage.

Coupled with the fact that it is open source, the Network Device Management Group Policy Extension is superior in terms of managing network-attached devices. Though it falls short when comparing the raw number of features, the focused scope of the extension makes this reasonable.

7.2.3 Microsoft Print Management

Microsoft Print Management [4] is an extension to Group Policy that shipped with Windows 2003 Server RC2. This extension, though it is not a client-side extension, allows administrators to specify network-connected printers that should be assigned to users and computers on a network. This functionality is achieved using an executable placed in the logon script of the users. Thus, this solution is a semi-elegant, though very much brute-force method, that effectively allows printer mapping and installation.

This feature, though it offers a small degree of useful functionality, does not make it comparable to the features provided by the Network Device Management Group Policy Extension. The features added by the prototype extension are significantly more advanced than those provided by the Microsoft solution. It is important to note that this functionality may be enhanced as Microsoft incorporates better printer management into Windows Vista Server.

8 Future Work

The development of the current Network Device Management extension proceeded according to schedule, resulting in an effective, deployable solution for management of both network-connected and network-attached devices. Despite the success, the extension can still be modified to increase efficiency, add functionality, and utilize newer system technologies. The following sections detail the advances that this project should undergo in future revisions, as well as the advantages each modification will garner.

8.1 Rewrite in C#

The key decision involved in the selection of the C++ language for development of the extension was the simple fact that most Microsoft Windows APIs were developed for that language. However, during the implementation research, several documents referenced accessing Windows API functions written in C++ using C#, via DLL calls. This made it possible, though prospectively more difficult, to implement the extension using the C# programming language.

In order to implement the project using C#, any Windows API functions would be accessed via calls to the DLL containing the compiled API function. The key to this feature is the "DLLImport" keyword in the C# language. This means that, although the functions utilized are written in C++ and will execute efficiently, applications written in

C# can access and manage those functions, executing in a managed environment. Aside from modifying calls to API functions, rewriting the extension will also require complete reimplementation, entailing the conversion of the language, but no the program logic.

The benefits of this conversion are notable. Firstly, C# creates managed code with advanced data structures that pose fewer inherent security risks than C++ (e.g. buffer overflows). Further, documentation for C# code is considerably more advanced, making maintainability and future upgrades significantly easier. Secondly, the registry access functions provided by C# are much more elegant than those provided by the C++ Windows API. Utilization of these functions will make registry changes – a key component of this extension – significantly easier and less prone to error. Thirdly, the utilization of the C# language will make the implementation of newer management frameworks, such as Windows Management Instrumentation, possible. As such, the functionality and robustness of the extension can be improved markedly, replacing older API calls with newer, more detailed and extensible management functions.

8.2 Driver Management Improvements

The current implementation has very basic device driver management abilities. Currently, the system supports only printer driver implementation – no other devices are supported. For the extension to remain useful, it must be extensible for all future devices that become network-attached, or that require management via this extension.

The printer driver management feature that is currently implemented is the weakest aspect of the extension. At present, it is limited to the installation of drivers located on the local system, which has been distributed in some manner (preferably within MSI packages). Future releases should allow the installation of drivers on network locations for increased ease. Additionally, the installation of devices and drivers should be more closely coupled to avoid repeated, unnecessary installations of drivers. This will increase client performance and reliability without impacting management.

8.3 Asynchronous Policy Application

Synchronous application of Group Policy settings is entirely effective at making the necessary system changes that this extension facilitates. However, this process requires that the user wait while operations such as printer installation, driver updates, and network connections are established. Prospectively, this delay could grow to be quite large in larger organizations.

In order to decrease this delay, the current Group Policy callback used by the extension should be modified such that is is asynchronous – that is, does not delay the user. If the callback is configured such that it indicates to the operating system that it performs policy application asynchronously, then a separate process will be launched wherein the policy settings of this extension are processed. Therefore, during the log on process, the user is not forced to wait for all policies to be applied before they are allowed to proceed with their tasks. The benefit here is decreased log on time, which results in greater productivity, and further minimizes off-line time for systems during maintenance involving the extension.

The effort required to implement this change is minimal. In essence, the callback function for the extension must be modified to utilize the asynchronous callback associated with completion of policy application. This function is provided to the extension callback when policy is refreshed, requiring only that it be called with some parameters when policy is successfully applied. The only other modification is a registry setting on the client machines that indicates the policy extension is asynchronous – this can be changed in the MSI configuration file.

8.4 Enable RSoP and Logging Support

Group Policy client-side extensions have the capability of sending detailed information about the policy that they apply to the operating system – a feature known as Resultant Set of Policy (RSoP) [31]. This information can be utilized to determine what settings have been applied to the client system or what prospective changes in Group Policy will do to client machines. This support is not required for client-side extensions, but the feedback provided by this functionality can aid administrators in troubleshooting policy problems. The incorporation of logging to either local or remote event logs can be implemented alongside RSoP. The two should be implemented simultaneously as they are both components related to administrative feedback and will record similar information.

The current extension does not provide feedback for RSoP analysis and logging.

This decision was made because it is not a necessary feature (especially during development), and because it is not supported on Windows 2000 systems. Since the development stage has yielded a usable prototype that is ready for deployment, implementation of these advanced features would be beneficial to system administrators, which constitute the end-users of the system. Clearly, the benefits include a clearer understanding of how policy settings will affect a network, as well as how individual and groups of machines on a network will react to policy changes.

In order to implement RSoP support, a small structural change must be made in the program to enable RSoP logging support. Specifically, a new entry point into the DLL must be created that provides RSoP data as the extension executes. This requires a new entry function, not a replacement. This is an important distinction because the existing entry point must be maintained for Windows 2000 systems, as they have no support (nor interest) in an RSoP callback. The other modification required is a registry modification (achievable in the MSI configuration file) that indicates the name of the RSoP-enabled Group Policy callback function. And, of course, the RSoP data must be generated, as necessary, and returned to the appropriate callback functions. More information on Group Policy callback functions and Resultant Set of Policy are available at Microsoft's website [32].

8.5 Incorporate Additional Devices

The purpose of this extension is to provide system administrators with a mechanism by which they can control the installation of network-attached and network-capable devices from a central location. Therefore, the greatest strength of this extension is its ability to add new devices to its management infrastructure. The modular design of the system allows the simple addition of other devices to the system, allowing virtually any kind of network-attached or network-controlled device to be centrally managed. Therefore, other network-attached devices, such as scanners (a part of many network-attached multifunction devices) should be added to the extension as the option becomes tenable.

The key limitation to the extension of the system resides in the available API calls. Because device access is difficult, if even available, using the C++ Windows API, it is almost certainly necessary to upgrade the system to the C# language. At the very least, utilization of WMI functionality using the available C++ API is required in order to expand the system.

As an ancillary note, modification of the system to support additional devices may require revised or completely new driver installation routines. The current design principle is that each supported device have an independent driver installation and management routine. As devices are added, a pattern, resulting in a more unified and streamlined management system, may be discovered. In any case, device addition will require some modification of the driver system.

8.6 Expanded Testing

Testing any software package thoroughly is a difficult task; testing a component that depends heavily on an individual network environment is even more so. The Network Device Management extension has been tested – and successfully deployed – on a production network with great success. However, not all possible system faults will emerge on this network. Therefore, it is necessary that the system be tested on a large number of networks in order to discover any hidden vulnerabilities or faults that the system possesses.

Since the extension itself is free of cost to system administrators, feedback should be easily available as they discover faults. In fact, given the number of websites dedicated to system administration and centralized management, the expectation of a significant amount of feedback from the end-users of the extension is almost guaranteed. This will ultimately result in the development of a much more powerful, secure extension that is tailored specifically for use by system administrators.

9 Conclusions

The benefits of utilizing network-attached devices under their native paradigm are significant. Potential benefits include network performance increases, decreased maintenance costs, and simplified network structure. The limitation in the implementation of this efficient paradigm has been the lack of management software capable of dealing with network-attached devices. Because of this, system administrators have opted to use network-attached devices as network-capable devices, which are dependent on a server for data management and access.

In order to demonstrate the viability of using network-attached devices under their native management paradigm, the Network Device Management Group Policy Extension was created. This component is a client-side extension to Group Policy that is designed to allow administrators to manage the installation of network-attached devices on client machines. In effect, this allows client machines to access device resources without an intermediary server. In this implementation, the extension supports network-connected printer connections and drive mappings. Further, it supports network-attached (IP-based) printers and the installation and upgrade of the associated drivers.

The distribution of this extension on a production network has been shown to greatly simplify the administration tasks related to printer management. For example, a change in the list of available printers no longer requires users to remove and reinstall a printer. Rather, the administrator simply changes an entry on the server and allows the change to be applied when the policy is refreshed.

Research using this extension also revealed that network bandwidth usage related to print data decreased 51.1% in test cases. Even though the overhead involved in the administration of network-attached printers is slightly higher than that of network-connected printers, the bandwidth decrease was shown to be more pronounced and asymptotically approach the 51.1% decrease as print traffic increases.

Finally, the research showed that the network topology was simplified using this extension. By removing dedicated print servers, troubleshooting and network analysis were simplified via the reduction of obviated nodes. The network and organizational benefits were exactly as expected. This has been shown an efficient method of retaining centralized management while allowing network-attached devices to operate under their optimal utilization paradigm – that is, managing themselves as autonomous devices.

Finally, the Network Device Management Group Policy Extension developed for this research provides functionality found in only one other product (PolicyMaker). Even in this case, the extension provides an open, structured framework that has the potential to manage all network-attached devices (e.g. scanners, label printers, etc.) - a feature unique unto itself. Despite this similarity, the Network Device Management Group Policy Extension demonstrates the importance, feasibility, and benefits of managing network-attached devices under their native management and utilization paradigm. Management hurdles need no longer be a stumbling block to efficient network design and utilization.

10 References

[1] WPClipart, "WPClipart," in WPClipart, [Online document], (2006 November), Available at HTTP: http://www.wpclipart.com

[2] DesktopStandard, "PolicyMaker™ Standard Edition," in DesktopStandard, [Online document], (2006 October), Available at HTTP: http://www.desktopstandard.com/PolicyMakerStandard.aspx

[3] FullArmor, "IntelliPolicy," in IntelliPolicy, [Online document], (2006 October), Available at HTTP: http://www.fullarmor.com/products-intellipolicy.htm

[4] Microsoft, "Deploy printers by using Group Policy," in Microsoft TechNet, [Online document], (2005 August), Available at HTTP: http://technet2.microsoft.com/WindowsServer/en/library/ab8d75f8-9b35-4e3e-a344-90d7799927231033.mspx?mfr=true

[5] Martin J. Garvey, "Most Important Products of 98," Information Week, December, 1998, Available at HTTP: http://www.informationweek.com/713/13iuprd8.htm.

[6] Silbershatz, Galvin, & Gagne, Operating System Concepts, 6th ed., New York, NY: John Wiley & Sons, Inc., 2003.

[7] Phillips, B., "Have Storage Area Networks Come of Age?," Computer, Vol. 31, July, pp. 10-12, 1998.

[8] Brother International, "MFC-440CN," in Brother International Multifunction Centers, [Online document], (2006 November), Available at HTTP: http://www.brother-usa.com/mfc/mfc_detail_AREA=MFC_1&PRODUCTID=MFC440CN.aspx

[9] Christopher McQueeney, "Paperless Office Be Damned!," Network Computing, August, 2004, Available at HTTP: http://www.networkcomputing.com/showitem.jhtml?docid=1516buyers.

[10] Hewlett-Packard Development Company, L.P., "Color HP LaserJet Printers," in HP Small & Medium Business Products, [Online document], (2006 November), Available at HTTP: http://h10010.www1.hp.com/wwpc/us/en/sm/WF02a/18972-236251-236268.html?jumpid=re_R295_prodexp/busproducts/printing/color-

laserjet-printers&psn=printing_and_multifunction/printers/color_hp_laserjet_printers

[11] Mark Burgess, "Theoretical System Administration," in 14th System Administration Conference (LISA 2000), 2000, pp. .

[12] William R. Stanek, Microsoft Windows NT Server 4.0 Administrator's Pocket Consultant, 1st ed., Redmond, WA: Microsoft Press, 1999.

[13] Nemeth, Snyder, Seebass, & Hein, UNIX System Administration Handbook, 3rd ed., Upper Saddle River, NJ: Prentice Hall PTR, 2001.

[14] Bruce Boardman, "Network Monitoring Systems," Network Computing, October, 2004, Available at HTTP: http://www.networkcomputing.com/showArticle.jhtml?articleID=47900819&pgno=1.

[15] Narayan Desai, Rick Bradshaw, Scott Matott, et. al., "A Case Study in Configuration Management Tool Deployment," in 19th Large Installation System Administration Conference, 2005, pp. 39-46.

[16] Apple Computer, Inc., Mac OS X Server: Deploying Mac OS X Computers for K-12 Education, 1st ed., Cupertino, CA: Apple Computer, Inc., 2004.

[17] Microsoft Corporation, "Windows Vista Print Management Step by Step Guide," in Microsoft TechNet, [Online document], (2006 November), Available at HTTP: http://www.microsoft.com/technet/windowsvista/library/34989e40-31f8-45e5-8fa2-9f2e076cbc93.mspx

[18] Alva L. Couch, Ning Wu, and Hengky Susanto, "Toward a Cost Model for System Administration," in 19th Large Installation System Administration Conference, 2005, pp. 125-141.

[19] Dale and Teague, C++ Plus Data Structures, 2nd ed., Sudbury, MA: Jones and Bartlett Publishers, 2001.

[20] D.S. Malik, C++ Programming: Program Design Including Data Structures, 2nd ed., Boston, MA: Thomson Course Technology, 2004.

[21] Gary J. Bronson, C++ for Engineers and Scientists, 1st ed., Pacific Grove, CA: Brooks/Cole Publishing Company, 1999.

[22] Microsoft Corporation, "Windows GDI," in Microsoft Development Network, [Online document], (2006 October), Available at HTTP:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdi/prntspol_7mgj.asp

[23] Microsoft Corporation, "Windows Installer," in Microsoft Developer Network, [Online document], (2006 October), Available at HTTP: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msi/setup/windows_installer_start_page.asp

[24] Microsoft Corporation, "Group Policy," in Microsoft Developer Network, [Online document], (2006 October), Available at HTTP: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/policy/policy/group_policy_start_page.asp

[25] Microsoft Corporation, "Windows Server® 2003 Group Policy," in Microsoft Windows Server® TechCenter, [Online document], (2006 October), Available at HTTP: http://technet2.microsoft.com/windowsserver/en/technologies/featured/gp/default.mspx

[26] Jeremy Moskowitz, Group Policy, Profiles, and IntelliMirror for Windows 2003, Windows XP, and Windows 2000, 1st ed., Alameda, CA: Sybex, Inc., 2004.

[27] Mark Burgess, "Automated System Administration with Feedback Regulation," Software - Practice and Experience, Vol. 28, December, pp. 1519-1530, 1998.

[28] Cristian Estan and Garret Magin, "Interactive Traffic Analysis and Visualization with Wisconsin Netpy," in 19th Large Installation System Administration Conference, 2005, pp. 177-184.

[29] Seong Soo Kim and A. L. Narasimha Reddy, "NetViewer: A Network Traffic Visualization and Analysis Tool," in 19th Large Installation System Administration Conference, 2005, pp. 185-196.

[30] Evan Hughes and Anil Somayaji, "Towards Network Awareness," in 19th Large Installation System Administration Conference, 2005, pp. 113-124.

[31] Microsoft Corporation, "RSoP WMI Classes," in Microsoft Developer Network, [Online document], (2006 November), Available at HTTP: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/policy/policy/rsop_wmi_classes.asp

[32] Microsoft Corporation, "ProcessGroupPolicyEx," in Microsof Development Network, [Online document], (2006 November), Available at HTTP: http://msdn.microsoft.com/library/default.asp?url=/library/en-

us/policy/policy/processgrouppolicyex.asp