An Extensible Architecture for Network-Attached Device Management

Michael J. McMahon, Jr. Sergiu M. Dascalu Frederick C. Harris, Jr. Juan Quiroz
University of Nevada, Reno
mike@mikerosoft.org dascalus@cse.unr.edu fredh@cse.unr.edu quiroz@cse.unr.edu

Abstract

The development of network-attached devices has ushered in an era of autonomous, multi-function equipment demanding minimal human interaction: the only requirements are data and electricity. Despite these advances, these machines continue underutilized in network environments due to operating system limitations regarding the management of these devices. These limitations force the use of these devices via other network hardware, such as a server, that manage the device access and data. While effective, this results in increased resource consumption and ignores the capabilities presented by network-attached devices. In order to facilitate optimal utilization of these devices, we have designed a new, extensible management architecture for all network-attached devices. This architecture, presented here, supports the central management of network-attached devices while allowing client machines access to the device without intermediate server hardware. Implementation of this paradigm on test networks has decreased resource consumption – especially bandwidth – considerably.

1. Introduction

Network-attached devices are a modern innovation in which the functionality of a single device – or multiple, integrated devices – is designed in such a way that access, management, and utilization of a device occur over a network connection independent of any other systems. Such devices have many advantages in any network when used under their native network-attached access paradigm: from a client perspective, network-attached devices allow clients to send data directly to the device via the network connection without any intermediate servers to manage access to the device, lowering access time and increasing productivity. From a management perspective, they provide self-hosted central management via the network connection, allowing administrators to make global changes easily.

Despite these advantages, most administrators do not utilize network-attached devices under their native access paradigm. The reason for this is a lack of management architecture for network-attached devices in modern operating systems [2]. All mainstream operating systems provide support for sharing, utilizing, and managing devices attached to a server running the same operating system. They do not, however, support the utilization of a device which is, itself, a server running a neutral operating system.

While operating systems lack the support required to manage and utilize network-attached devices fully, they do have support for extending their capabilities with new access and management architectures. Utilizing these features, it is therefore possible to add an architecture for network-attached device management and access. This would allow the benefits of these devices to be fully realized.

While it is not possible to incorporate every network-attached device into the initial architecture (there is no limit to the types of devices that can be connected directly to the network), it can be designed in such a way that it is easily extensible, allowing for the simple addition of newly created device classes. Such an architecture was designed, implemented, and tested for the purposes of improving network resource utilization.

The remainder of this paper is organized such that the “Background” section details the features of network-attached devices and introduces terminology relevant to the remainder of the paper. The “Management Paradigms” section details the management features currently available in operating systems, as well as those required in an optimal access and management architecture. In “Optimal Architecture,” the architecture developed and implemented for the optimal utilization of network-attached devices is detailed. Testing results are given in the “Analysis” section and the paper concludes with a summary of the findings. Finally, future work directions and tasks are addressed in the "Future Work" section.

2. Background

The term "network-attached device" is applied to a great number of 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. As such, a good number of devices and associated categories fit within this classification. It is therefore necessary to clarify the term in order to convey the precise meaning within the context of this paper. As such, the subsequent sub-sections detail the meaning of, and differences between, network-attached and network-capable devices.

2.1. 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. These devices are autonomous – requiring no external data management or support – and require only energy and data to carry out their specific (often specialized) function(s). The resources or functionality that they provide is designed to be accessible to any type of client via a network connection.

2.2. Network-Capable vs. Network-Attached

A network-attached device, within the context of this paper, 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.

3. Management Paradigms

The management of network-attached devices is significantly easier and more efficient than that of network-capable devices, which require dedicated or shared servers. However, the modern use of network-attached devices is not as autonomous entities, but rather as network-capable devices that require a server. This choice results from limitations within the operating systems of the network entities that wish to utilize the network-attached device [2]. Specifically, this deficiency is within the installation and maintenance mechanisms that the systems provide for centralized device management.

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. This is the easiest and most common scenario in organizations, as the server can often broadcast device information and automate device installation to clients to some degree, thus simplifying management and client installation.

If the device is installed with no supporting hardware as an autonomous device (which it is), then it utilizes the network-attached paradigm (see Figure 1). This is an optimal paradigm in terms of network bandwidth use and device autonomy.

The following sub-sections are designed to illustrate the management differences and trade-offs between the utilization of a network-attached device under its native paradigm (which has been described here), and as a network-capable device [4],[5].

3.1. Network-Capable Devices

When treated as a network-capable device, administrators can often utilize the driver installation facilities of the operating system to alleviate the effort required for a successful installation for a network-attached device. Allowing a server to host the device allows administrators to script the installation – a sub-optimal process in itself – and centrally orchestrate the installation of the device (removal is not always as easy) [6],[9]. The trade-off here is that management of the device is simplified, however additional resources are required: bandwidth, servers, electricity, wiring, maintenance, etc.

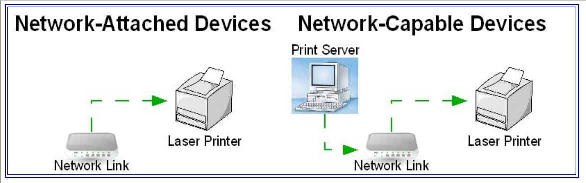

Figure 1. Illustration of the difference between network-capable and network-attached devices [1].

3.2. Network-Attached Devices

The installation of an autonomous network-attached device under a network-attached (native) 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.

4. Optimal Architecture

In order to allow the optimal utilization of network-attached devices under their native management paradigm, an extension was created that embodies the efficiency of network-attached devices, yet retains the ease-of-management inherent in network-capable devices [7],[8],[10]. This architecture was test-implemented [11],[12],[13],[14],[15] under the Microsoft® Windows® operating system as an extension to Group Policy [16],[17],[18],[20]. The choice of the Windows® operating system was made because it is the most commonly-used client operating system in business environments and, as such, implementation of the architecture under this operating system stands to benefit the most people. The architecture itself is indifferent to the host operating system and may be implemented for any operating system.

The architecture is designed for networks in which a single (or small set) or servers can supply configuration information to all client machines on the network. Since all major operating systems are capable of this in varying forms, the particular implementation of the framework will vary, but the core organizational and design principles should remain the same.

4.1. Structure

The basic structure of the architecture places management of all network-attached devices within the purview of one central module. This module is executed directly by the operating system at reasonable intervals (e.g. startup, shutdown, or periodically) to perform management tasks that update the client system with new configuration information related to network-attached devices present on the network.

The central module then executes each registered module that has been added to it. These individual modules represent a particular type of device, which may be very generic or extremely specific to the device it manages. This may range from a module to manage all network-attached printers to a module that manages only network-attached fax machines. In any case, each module completely manages a particular type of device independently of any other module.

As shown in Figure 2, this creates a branching structure from the central module to all other modules. The architecture is thus easily extensible to any future or current network-attached device. The addition of a module to the central module requires only that a function call be placed in the appropriate section of the central module code.

4.2. Modules

The central module is responsible for providing an executable interface to the operating system, as well as retrieving configuration information from a central server. This configuration information consists of data necessary for the installation or modification of network-attached devices on client machines. In the case of printers, this would include driver and IP port information; storage devices would require only network addresses.

Individual device type management is broken up into autonomous modules, each of which manages a particular type of device. Each module has full access to configuration information passed to it by the central module. Adding an additional device type to the system is thus merely a matter of adding a function call to the central module and ensuring that the newly-added module stores its information in the same location as other modules. In the case of a Windows system, this location is a particular registry key that is

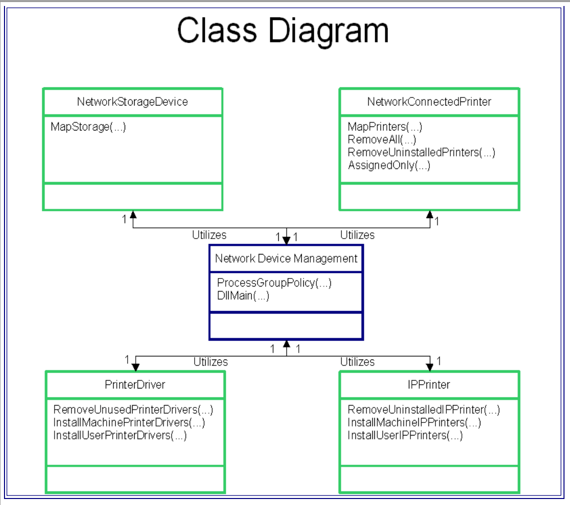
Class Diagram
NetworkStorageDevice
MapStorage(...)
NetworkConnectedPrinter
MapPrinters(...)
RemoveAll(...)
RemoveUninstalledPrinters(...)
AssignedOnly(...)
1
Utilizes
1
1
Utilizes
1
Network Device Management
ProcessGroupPolicy(...)
DllMain(...)
1
1
1
Utilizes
Utilizes
1
PrinterDriver
RemoveUnusedPrinterDrivers(...)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

Figure 2. Class diagram for the prototype client-side extension showing static classes.

automatically populated by Group Policy with settings provided by a server.

Each registered device type that the central module manages is independent of any other. Since the types of devices that are network-attached are (and surely will be) disparate, the configuration information for each device type is compartmentalized. When executed, each individual module processes all configuration information pertaining to the type of device it manages. For example, when the IP printer module executes, it processes all configuration information for IP-based printers. As such, it installs, updates, and removes all IP-based printers before the next module is executed.

4.3. A Word on Drivers

After careful experimentation, it was decided that the architecture would mandate that device driver installation not be a separate module. Rather, driver installation for a device must be handled within the same module that installs and configures the device. To illustrate, consider the installation of network-attached printers and their drivers: the options are to install all drivers and then all printers that use those drivers, or to have each printer installation routine install the driver for that particular printer. Since the former case would, ultimately, result in inconvenient module dependencies, the latter case is preferable and the defined method for printer installation/management under this architecture. The architecture with a separated driver module is shown in Figure 2.

4.4. Execution Sequence

The actual execution of any implementation of the management extension architecture will, just as its actual implementation, vary by operating system. However, the general method of execution will entail a query by a client to a server for configuration information, with which the server will respond with the new or current configuration information. This query may occur at any time that the system is running and will also depend on the properties of the operating system – the most important property being any built-in synchronization features that the operating system provides.

Because the architecture defined here is only an extension to the operating system, it is free to be configured (as much as necessary for the particular system it operates on) to utilize existing system features. An example of this is given in Figure 3, which represents the execution sequence for an implementation of the architecture under the Microsoft Windows environment when created as an extension to Group Policy. As this illustrates, because the extension utilizes Group Policy, it is subject to the same execution and refresh constraints as the Group Policy implementation.

5. Analysis

In order to verify the performance of the newly-developed architecture, it was implemented as an extension to Microsoft® Group Policy on the Microsoft® Windows® platform. After extensive preliminary testing, the extension was deployed on a test network and the performance of the systems was analyzed.

The test environment consisted of a network of 1 client computer, 2 servers, and a network-attached printer (Brother MFC-420CN [3]) connected to the network.

In order to comparatively test the native network-attached paradigm against the network-capable usage, two test scenarios were run for identical print jobs. The first involved utilizing a network-attached printer as a network-capable device under the paradigm provided by the operating system. Here, the server was configured to manage the network-attached device. In the second scenario, the network-attached printer was utilized under its native paradigm using the newly-developed extension.

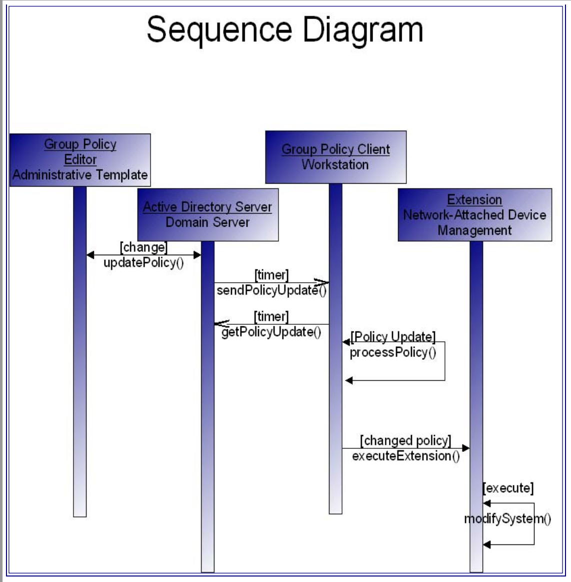
Sequence Diagram
Group Policy Editor Administrative Template
Active Directory Server Domain Server
Group Policy Client Workstation
Extension Network-Attached Device Management
[change] updatePolicy()
[timer] sendPolicyUpdate()
[timer] getPolicyUpdate()
[Policy Update] processPolicy()
[changed policy] executeExtension()
[execute] modifySystem()

Figure 3. The sequence of events and triggers responsible for the propagation of settings that allow the management of network-attached devices.

Under these scenarios, bandwidth measurements were made using the monitoring tools located on the gigabit switch to which the devices were connected. The results of these measurements are presented in the proceeding sub-sections.

5.1. Management Bandwidth

In this phase of testing, the bandwidth consumed for the installation of a printer under the two paradigms was monitored. The results were consistent across all iterations of the test – a sensible result since the printer was the same in all cases, as was the driver.

In all trials, the installation of the network-attached printer under the network-capable paradigm (operating system supported) required 3.5MB of bandwidth. This consisted of both management data and the driver installation files from the server hosting the device to the client.

The network-attached paradigm test required 6MB in all trials. The data transferred consisted of management and driver installation files held on the Active Directory server.

The results here show that the use of the network-attached device architecture implementation require 71.43% more bandwidth than the network-capable scenario. Upon examination, it was discovered that the difference was due to the fact that the operating system-specific driver was installed in the network-capable case, whereas the entire driver was transferred in the network-attached case.

5.2. Print Job Bandwidth

For this phase of testing, the bandwidth consumed on the network during the transmission of a print job from the client to the printer was measured. A single print job of 2.5 MB was sent to the device and the results averaged.

The average performance under the network-capable paradigm was that the 2.5 MB print job consumed 5.1 MB of bandwidth. The data path observed involved a 2.6 MB transmission from the client to the device server and a 2.5 MB transmission from the server to the device. The slight increase in the initial transmission size has been attributed to control data sent to the server by the client in the passing of the print job.

Under the network-attached paradigm, the average bandwidth consumed by a 2.5 MB print job was 2.5 MB. This was the expected result, as no retransmission of data occurred.

5.3. Consequences

Based on the results of the tests, the network-attached paradigm clearly decreases print job bandwidth 51.1% as compared to the network-capable alternative. Although the initial management bandwidth was higher, this was a one-time consumption. One must consider that with even moderate print usage, the bandwidth savings will quickly nullify the increased cost of installation. Over time, the network-attached paradigm will approach the observed bandwidth savings of 51.1%.

6. Conclusion

This paper has outlined an architecture for the centralized management and utilization of network-attached devices. This architecture is the first attempt to codify the client-side utilization of these devices such that the devices themselves are used in an efficient manner, i.e., having minimal resource consumption and administration requirements.

Implementation and testing of this module-based architecture under the Microsoft® Windows® operating system indicated that bandwidth consumption under the network-attached paradigm decreased 51.1% as compared to the network-capable case. Despite a short-term management bandwidth increase of 71.43% over the network-capable instance, the network-attached architecture was shown to asymptotically approach a 51.1% bandwidth savings over time.

Clearly, the benefits of the implementation of this architecture have noteworthy practical benefits.

7. Future Work

The current implementation of the architecture is rough, at best. Future implementations should refine the driver installation process – a goal that may significantly reduce the bandwidth consumed during management updates.

While the current implementation of the architecture provides the necessary functionality using C++ API calls to Windows®, better solutions exist. Under development is a new system utilizing Windows Management Instrumentation [19] and the C# language. This change stands to significantly improve performance and simplify the incorporation of additional device classes in the future.

On a broader scale, the adaptation and implementation of the architecture on other platforms should be completed. While this is not an immediate goal, it is one that can be accomplished either independently or by operating system manufacturers. Other target platforms include Linux and Mac OS X.

8. References

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

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

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

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

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

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

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

[8] 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.

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

[10] 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.

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

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

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

[14] Microsoft Corporation, "Windows GDI," in Microsoft Development Network, [On-line document], (2006 October), Available at HTTP: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdi/prntspol_7mgj.asp

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

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

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

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

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

[20] Microsoft Corporation, "ProcessGroupPolicyEx," in Microsoft Development Network, [On-line document], (2006 November), Available at HTTP: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/policy/policy/processgrouppolicyex.asp