Automated Configuration of Windows Network Settings

Graham Mainwaring

Thomas Edison State College

Author Note

This report was prepared for Lib-495 Liberal Arts Capstone, section OL012 2013Oct, taught by Dr. Nina Rosenbaum-Haydel.

## Abstract

This report describes a software development project named NetShuffler which adds automatic network settings management to the Microsoft Windows operating system. Windows already contains a built-in facility, called Network Location Awareness (NLA), for managing connections to multiple networks. However, the built-in NLA feature lacks key capabilities needed by highly-mobile information workers. For example, NLA can change the default printer, but it cannot map or un-map network drives or make VPN connections. NLA can also fail to distinguish between multiple locations within the same Active Directory domain. The NetShuffler software supplements the existing NLA system, increasing the reliability of network recognition and adding new capabilities. NetShuffler also adds the ability to run user-specified commands when a network is recognized, so it is fully customizable in a way that NLA is not.

Table of Contents

[Abstract 2](#_Toc375158703)

[Chapter I: Introduction 5](#_Toc375158704)

[Introduction 5](#_Toc375158705)

[Background of Topic 5](#_Toc375158706)

[Illustrative Example 5](#_Toc375158707)

[Problem Statement 7](#_Toc375158708)

[Main function: 7](#_Toc375158709)

[Sub-functions: 7](#_Toc375158710)

[Professional Significance 8](#_Toc375158711)

[Methodology 9](#_Toc375158712)

[Delimitations 9](#_Toc375158713)

[Definition of Terms 10](#_Toc375158714)

[Summary 10](#_Toc375158715)

[Chapter II: Review of Existing Solutions 11](#_Toc375158716)

[Introduction 11](#_Toc375158717)

[Current Solutions 11](#_Toc375158718)

[Microsoft Windows 11](#_Toc375158719)

[NetSetMan Pro 11](#_Toc375158720)

[Net Profiles 11](#_Toc375158721)

[Apple OS X (Macintosh) 12](#_Toc375158722)

[Sidekick 12](#_Toc375158723)

[Summary 12](#_Toc375158724)

[Chapter III: Methodology 13](#_Toc375158725)

[Introduction 13](#_Toc375158726)

[Development Process 13](#_Toc375158727)

[Graphical User Interface 14](#_Toc375158728)

[Windows Forms 14](#_Toc375158729)

[Active Networks 14](#_Toc375158730)

[Hiding Networks 14](#_Toc375158731)

[Profiles 15](#_Toc375158732)

[Adding Profiles 15](#_Toc375158733)

[Adding Profile Actions 15](#_Toc375158734)

[Recognizers 15](#_Toc375158735)

[.NET Network Enumeration 16](#_Toc375158736)

[Network Location Awareness 16](#_Toc375158737)

[WiFi Network Connections 16](#_Toc375158738)

[Network Operational Status 16](#_Toc375158739)

[Loopback and APIPA Networks 16](#_Toc375158740)

[Multiple Simultaneous Connections 17](#_Toc375158741)

[IPV6 vs IPV4 17](#_Toc375158742)

[Profile Action Threading Model 17](#_Toc375158743)

[Configuration Persistence 17](#_Toc375158744)

[Summary 18](#_Toc375158745)

[Chapter IV: Results of the Project 19](#_Toc375158746)

[Introduction 19](#_Toc375158747)

[Installing NetShuffler 19](#_Toc375158748)

[Compiling NetShuffler 19](#_Toc375158749)

[Chapter V: Summary and Discussion 21](#_Toc375158750)

[Introduction 21](#_Toc375158751)

[Problem Statement & Explanation of Project 21](#_Toc375158752)

[Review of Methodology 22](#_Toc375158753)

[Summary of Project & Discussion of Results 22](#_Toc375158754)

[Relationship of Project to the Field 23](#_Toc375158755)

[Conclusions 23](#_Toc375158756)

[Annotated Bibliography 24](#_Toc375158757)

# Chapter I: Introduction

## Introduction

For people who must work in more than one location, managing their computer settings is becoming a problem. Each time a new location is arrived at, users must go through a cumbersome process of re-establishing their network settings – choosing a default printer, mapping network drives and so forth. The software described in this report, named NetShuffler, provides automated configuration management for highly-mobile workers, allowing the creation of rules like “only when I am at the Maple Street office, my default printer should be HP LaserJet at the front desk.”

## Background of Topic

According to research by McKinsey & Company, “In the United States, 45 percent of temporary employees work in management, in IT or technical occupations, or in health care, and contract work has grown four times faster than total employment over the past decade” (Lund, Manyika, & Ramaswamy, Preparing for a new era of work, 2012). As a result, many workers now split their time between multiple offices. This new type of worker does not simply “go to work” in the traditional sense. For example, a worker with two part-time jobs might work at one location on Monday and Wednesday, a different location on Tuesday and Thursday, and from home on Friday. A contract worker or consultant might visit many employers or customers in a single day.

## Illustrative Example

To better understand the situations in which this software would be useful, consider an imaginary consultant, Fred. Fred is currently working on projects for two customers, Acme and BigCo. Fred sometimes works from home and sometimes works from one or the other customer offices. Each work site has different network requirements.

* At home, Fred has a network drive he uses for backup. This is mapped as a shared drive H:, connected to the server address \\home.lan\backup.
* At Acme, the client has a shared drive M:, and expects Fred to work from documents stored on this drive. The network address is \\server439421.acme.lan\project1321.
* At BigCo, there is a third shared drive S:, connected to \\WINRDU.bigco.lan\SHARED, which Fred is also expected to work from.

Now, consider the scenario where Fred has spent the morning working at Acme and returns to the home office for the afternoon. It is very unlikely that Fred will remember to disconnect the M: drive before shutting down his laptop. When Fred arrives home, he will have to wait for a timeout and click “cancel” on the computer’s attempt to re-connect to the server. To stop this, Fred will now disconnect the drive. But then when Fred arrives back at Acme the next day, he opens Excel and tries to click on a file from the “recently used” list. Because this file is on the M: drive, which is no longer connected, Excel gives an error. Fred now has to go back out to Windows Explorer, find the scrap of paper he wrote the network path on and re-connect the drive–perhaps typing it incorrectly two or three times before succeeding in connecting. This is an ongoing annoyance for Fred.

Further, consider the scenario where Fred works at Acme in the morning and BigCo in the evening. Unless Fred remembers to disconnect before leaving Acme, his laptop will now be making requests to the BigCo network for the Acme file server. These will fail just like they do at home–but Fred is concerned about the possibility that BigCo may be watching his network traffic and may now be able to infer that he also works for Acme, which they would not otherwise have known. (BigCo and Acme may be competitors, or confidentiality may be required for some other reason.)

NetShuffler handles this situation by having Fred configure his network connections within NetShuffler rather than within Windows Explorer. In this example, Fred would have three profiles, each with a Connect Network Drive action. NetShuffler will attempt the connection only when it detects, through some specified method, that it is already connected to the relevant network. So after configuring the software, Fred can simply plug in his laptop, log in, and have the appropriate network drive mapped, without resorting to scraps of paper or worrying about confidentiality issues.

## Problem Statement

The Microsoft Windows operating system, used by 90% of desktop computers, contains a built-in facility for managing this sort of movement between locations. This function is called Network Location Awareness, or NLA. While NLA is capable of managing connections to multiple networks, it lacks key features needed by this new sort of highly-mobile information worker. For example, NLA can change the default printer, but it cannot map or un-map network drives or perform other actions. NLA can also fail to distinguish between multiple locations within the same Active Directory domain, which is a problem for workers who visit multiple sites within the same large organization.

The NetShuffler software supplements the existing NLA system by adding additional functions. These include additional methods for identifying the user’s location based on network information, and additional actions that can be taken when a location is detected.

### Main function:

* Reliably determine the location of the user by observing network connections.
* Execute actions specified by the user when a particular location is entered or departed.

### Sub-functions:

* Use Network Location Awareness, domain name, WiFi network name and other information sources to establish the identity of connected networks.
* Provide a Windows graphical user interface from which the user can assign actions to networks.
* Use Windows .NET and Win32 API functions to monitor the connection state of the computer and respond to changes.
* Execute configured actions at the appropriate moment, based on network connection changes.
* Be robust in the face of real-world issues such as:
  + Connecting to two or more networks at the same time (wired+WiFi or wired+WiFi+VPN or similar);
  + Transient connection states, such as the state when a network is partially-connected for a short period of time while waiting to be assigned an IP address;
  + Multiple networks which share some of the same identifying information;
  + Not re-executing actions unnecessarily, such as when a user switches from wired to WiFi while still at the same location.

## Professional Significance

NetShuffler provides capabilities not available from any other currently-available software. While the Windows Network Location Awareness service provides some of these functions, professional users require much more flexibility.

Microsoft has improved NLA over time, and can be expected to continue to do so, but the pace of such progress has been slow:

* Windows XP was released for general availability on December 31st, 2001 (Microsoft Corporation, 2013). It was the first version of Windows to include network location awareness. In Windows XP, NLA was conceived as a tool for applications programmers and did not make any configuration changes of its own (Microsoft Corporation, 2011).
* Windows Vista (released January 30, 2007) improved the applications programming interface, and also added the ability for the user to add conditional Windows Firewall rules that would only apply within a specific network location (Microsoft Corporation, 2009a).
* Windows 7 (released October 22, 2009) added the ability to change the default printer based on network location (Microsoft Corporation, 2009b).
* Windows 8 (released October 26, 2012) and Windows 8.1 (released October 18, 2013) offer the same NLA system as Windows 7.

We can expect that Microsoft will continue to improve Windows, and may eventually add all the features of the software described in this report. If this occurs, the need for this software will diminish and eventually disappear entirely. However, as we can see above, Microsoft has only added two relevant features to NLA in the past decade. It seems reasonable to conclude that supplementary software, like NetShuffler, will remain relevant for some time to come.

## Methodology

The software will be developed using the following technologies:

* Microsoft Visual Studio 2012
* The C# programming language
* The .NET Framework version 4.0
* The Win32 API

## Delimitations

The NetShuffler software will be built and tested on Windows 7. As a .NET 4.0 application, it should be compatible with Windows XP, Vista, 7 and 8, but the time available for the project does not permit building a test lab and handling all compatibility concerns for multiple versions of Windows.

The project includes only the software itself. No user manual, help file, installer program or other ancillary items will be included.

## Definition of Terms

*.NET*. A software framework developed by Microsoft to provide language independence and improved platform services to application developers, primarily under Windows.

*C#*. A modern, general-purpose programming language, currently the 5th most popular in the TIOBE listing (TIOBE Software, 2013). C# was initially developed by Microsoft as the premier language for the then-new .NET platform. Later, it was formally standardized as ECMA-334 and ISO/IEC 23270:2006.

*Visual Studio*. An integrated software development environment made and sold by Microsoft Corporation, and used by programmers to construct software products such as Windows and web applications.

*Win32 API*. The application programming interface for 32-bit Windows applications. Although this has been replaced to some extent by .NET, many system functions are not yet available through .NET and must still be accessed using “raw” Win32 API calls.

*Windows.* The desktop operating system produced by Microsoft Corporation and running on the majority of the world’s desktop and laptop computers. The terms Windows XP, Windows Vista, Windows 7 and Windows 8 refer to specific versions of this software.

## Summary

This report describes a software project named NetShuffler, which provides capabilities supplementary to the Microsoft Windows Network Location Awareness service. NetShuffler will provide options for determining the user’s location from available network information, and to perform desired actions when a location is entered or exited.

# Chapter II: Review of Existing Solutions

## Introduction

Several software products similar to NetShuffler already exist, but none simultaneously offer silent, “set-and-forget” background operation, support for Microsoft Windows and profile-based administration of a broad selection of configuration settings.

## Current Solutions

### Microsoft Windows

As discussed in the introduction, Microsoft Windows offers a built-in feature called Network Location Awareness (NLA). In Windows 7 and 8, NLA offers the ability to run conditional firewall rules (Microsoft Corporation, 2009a) and to change the default printer (Microsoft Corporation, 2009b). NLA does not offer the ability to map a network drive, connect to a VPN or run an external program.

### NetSetMan Pro

NetSetMan Pro (Herlein, NetSetMan Help File, 2010) offers an extensive, profile-based system for managing network settings, but it lacks any detection capability. It is intended for the situation where the user wishes to issue commands to apply changes to a network profile. For example, network system administrators must often reconfigure their computer to perform diagnostic and troubleshooting tasks. NetSetMan Pro simplifies the process, for example, of switching back and forth between two different IP addresses on the same network. Ultimately, it is a good solution to a different problem.

### Net Profiles

Net Profiles (Milner, netprofiles, 2011) also offers a profile-based system for managing network settings, and adds automatic profile selection based on WiFi connection. However, its primary focus is still the manual selection of a network profile rather than “set-and-forget” operation.

### Apple OS X (Macintosh)

### Sidekick

Sidekick (Oomph, Inc., 2012) offers an easy-to-use profile-based configuration system combined with full network detection features. It is able to perform more than 20 different actions based on network location, and is fully “set-and-forget.” It is only available for Macintosh systems running OS X.

## Summary

While other products exist that provide some of the features of NetShuffler, none combine the essential features of background operation, profile-based administration and support for Microsoft Windows.

# Chapter III: Methodology

## Introduction

NetShuffler is a software project that automates network configuration management for mobile users who connect to many different networks. It collects data from multiple sources to determine the user’s location. Once this is determined, the software can map network drives, set a default printer and run external commands.

## Development Process

Development began with a small project that served as a feasibility study for the network recognition functions, which was the key unsolved technical problem facing the developer. Initially, a prototype recognizer was developed as a console application. The developer then connected a laptop running the prototype recognizer to networks at several businesses and retail locations to verify that the recognizer would be triggered and would detect networks correctly.

During this process, it was found that domain name (DNS) recognition was unreliable when connecting to a network with a slow dynamic IP address assignment (DHCP) server. In this circumstance, the adapter would continue to show the DNS name assigned by whatever previous network had been most recently connected to. As a result, the recognizer would falsely detect the old network. To solve this, a function was added to detect and ignore the case where a dynamically addressed network is connected but no IP address has been assigned.

Once a reliable recognizer had been developed, the rest of the software – the graphical user interface, profile-action system and the individual action modules – was developed in a single “sprint,” in no particular order and with no strongly structured process. The remainder of this chapter describes the key design issues that were considered during this effort, and how they relate to each sub-function of the software.

## Graphical User Interface

NetShuffler interacts with the user by displaying a graphical user interface, or GUI. This GUI is designed to be unobtrusive most of the time. It should only be visible to users when they wish to change the software’s configuration. To accomplish this, it will run as a system tray icon (the area next to the clock on the right hand side of the Windows taskbar). In normal operation, there will be no other visible interaction with the user. When the user wishes to change configuration settings, the icon provides a convenient way to begin the interaction by double-clicking or right-clicking.

### Windows Forms

The NetShuffler GUI is developed using the Microsoft toolkit known as Windows Forms. This provides a traditional-looking interface with menus, lists, buttons and so forth. The application does not use Windows Presentation Framework or a model-view-controller architecture. Because there will only ever be one GUI (this will, for example, never be a web app), a design decision was made to combine GUI and functional code within the same source files.

### Active Networks

The top panel in the GUI shows active networks. These networks have been detected by one or more of the recognizers described below. Networks are displayed here only if we have some evidence that they are actually connected. For example, if a computer has a wired Ethernet port with nothing plugged in, or a WiFi adapter not connected to any network, then these networks would not appear here, because they would not be classified as active.

### Hiding Networks

Some network connections are inherently uninteresting for our purposes. For example, the widely used software package VMware Workstation installs host virtual network adapters used only for communicating with guest virtual machines (VMware, Inc., 2013). To reduce clutter in the user interface, NetShuffler gives the option to hide selected adapters.

### Profiles

The lower panel in the main window is a list of profiles. A profile comprises a detection method and a list of actions. When the result of the detection method transitions from false to true, the configured actions will be run. Only the profile description is shown on the main window. To see the list of actions, the user must double-click on the profile.

### Adding Profiles

Double-clicking on a network connection allows the user to create a profile. The user can also right-click on a network connection to create a profile. The profile creation screens will be based on the network connection selected, but once created, the profile will match any network connection that matches the specification – it is not uniquely associated with the connection from which it was created.

### Adding Profile Actions

Profile actions are not visible on the main screen. To view them, the user can double-click on the profile. From this secondary screen, actions can be added and deleted from profiles.

## Recognizers

Once profiles and actions have been configured, NetShuffler must collect information regarding current network status to determine which profile is active. A profile is considered active when its connection specification evaluates to true. For example, if a profile connection specification is “WiFi SSID = Coffee Shop” then the profile will be considered active whenever we have evidence that any network adapter is connected to a WiFi network with an SSID (network name) of “Coffee Shop.”

### .NET Network Enumeration

The initial list of connected networks is generated using the built-in .NET class called System.Net.NetworkInformation (Microsoft Corporation, 2013b). This class reliably lists network adapters and some of their details such as IP address, but does not give all information about each adapter.

### Network Location Awareness

The Windows Network Location Awareness system can be accessed from .NET programs using a component called Network List Manager (Microsoft Corporation, 2013c). NetShuffler uses this component to retrieve additional information about each network. Specifically, NLA classifies each connection as public, private or domain, and assigns a name to each network. NLA also provides a guess whether the network is connected to the Internet.

### WiFi Network Connections

There is no standard .NET library for retrieving details about wireless networks. Because of this, NetShuffler uses Win32 API calls to determine whether a given network connection is wireless, and if it is, to find the SSID (network name).

### Network Operational Status

Networks are considered operational if the OperationalStatus parameter of the System.Net.Networkinformation class is set to OperationalStatus.Up. No additional checking is done to determine if the network is truly functional.

### Loopback and APIPA Networks

If a network is operational but only contains loopback or APIPA network addresses, then it is ignored. Loopback addresses are those addresses which refer only to the local machine (such as the familiar IPV4 address 127.0.0.1). APIPA addresses are IPV4 addresses within the subnet 169.254.x.x, which are assigned when a dynamically configured network adapter has been unable to obtain an IP address from a server. It is assumed that an adapter with only these addresses is not actually communicating with any viable network.

### Multiple Simultaneous Connections

If multiple networks are connected at the same time and multiple profiles are active, then the actions for each profile will be run. There is no Boolean logic applied to profiles, so it is not possible for a user to specify complex rules like actions to perform only when two or more profiles are active.

### IPV6 vs IPV4

Most Internet-protocol networks use version 4 of the Internet Protocol, known as IPV4. This is the protocol with familiar four-byte IP addresses that like “192.168.0.1.” Because of a shortage of addresses, network operators are beginning to switch to version 6, known as IPV6, with addresses that look like “2001:db8::ff00:42:8329.” Most people using NetShuffler will be using a computer capable of addressing IPV6, but which never connects to an actual IPV6 network. In order to remove clutter from the user interface, we provide the option to hide and ignore these IPV6 addresses.

## Profile Action Threading Model

In order to run quietly in the background without freezing the GUI, all profile actions are executed using parallel processing, in a context known as a thread. The Windows Forms GUI system is not thread-safe and GUI objects can only be accessed from a single thread. In order to ensure there are no problems with thread safety, profile actions are executed using isolated “runner” classes. Once configuration data is copied to the runner class, it executes completely independently from the GUI thread.

## Configuration Persistence

Configuration settings are saved and loaded to an XML file using a built-in .NET capability known as serialization, in which classes are inspected by the .NET framework and their content is converted to or from XML. This system imposes certain restrictions on the kinds of properties that can be included in the class, so the ProfileData and AppConfig classes and their member fields are required to conform to the limitation of being XML-serializable.

## Summary

NetShuffler uses recognizers, profiles and profile actions to detect network settings and execute appropriate commands, and also provides a GUI to allow the user to configure these settings. This chapter described the design issues that were considered during the development process. More details are available in the source code comments of the project itself.

# Chapter IV: Results of the Project

## Introduction

The purpose of this project was to develop a software package, named NetShuffler, to assist highly-mobile Windows laptop users in managing their computer configuration settings as they move between locations. To do this, the software determines the location of the user by observing properties of their network connections. When a location is recognized, the software executes actions defined by profiles specified by the user. The software includes actions for mapping a network drive, changing the default printer, connecting to a VPN and executing an arbitrary external program.

## Installing NetShuffler

The attached file, NetShuffler20131120.zip, contains the source and executable code of the NetShuffler software. To install and run NetShuffler, extract the file NetShuffler.msi from the Installer\Output subdirectory, double-click on it and follow the prompts.

## Compiling NetShuffler

To compile NetShuffler from source, you must have the following software installed:

* Microsoft Visual Studio 2012, available from:  
  <http://www.microsoft.com/en-us/download/details.aspx?id=34673>
* The DotRas SDK version 1.3, available from:  
  <http://dotras.codeplex.com/>
* To build the installer, you must also have the freeware version of Advanced Installer version 10.7.1, available from:  
  <http://www.advancedinstaller.com/>

To review the source code and build the project, extract the entire contents of the zip file to your Visual Studio project folder, and then double-click on NetShuffler.sln. The bulk of the program is in the files named NetShufflerForm.cs and NetShufflerClasses.cs. No special build parameters are required.

# Chapter V: Summary and Discussion

## Introduction

For people who must work in more than one location, managing their computer settings is becoming a problem. Each time a new location is arrived at, users must go through a cumbersome process of re-establishing their network settings – choosing a default printer, mapping network drives and so forth. The software described in this report, named NetShuffler, provides automated configuration management for highly-mobile workers.

To do this, the software provides three main functions. First, it integrates data from multiple sources to determine the user’s network location reliably. Second, it provides a flexible configuration interface that allows the user to specify which actions should be taken when connected to each network. And third, it provides seamless background execution of each configured action, with minimal or no troublesome pop-ups or other unwanted interactions.

## Problem Statement & Explanation of Project

The Microsoft Windows operating system, used by 90% of desktop computers, contains a built-in facility for managing this sort of movement between locations. This function is called Network Location Awareness, or NLA. While NLA is capable of managing connections to multiple networks, it lacks key features needed by this new sort of highly-mobile information worker. For example, NLA can change the default printer, but it cannot map or un-map network drives or perform other actions. NLA can also fail to distinguish between multiple locations within the same Active Directory domain, which is a problem for workers who visit multiple sites within the same large organization.

The NetShuffler software supplements the existing NLA system by adding additional functions. These include additional methods for identifying the user’s location based on network information, and additional actions that can be taken when a location is detected.

## Review of Methodology

NetShuffler was developed using prototyping and incremental development. Each proposed component was evaluated for feasibility and difficulty of implementation. Network recognition rapidly emerged as the most difficult problem, so this subsystem was prototyped prior to investing effort in the rest of the project. The reliability of the prototype was evaluated and improved by carrying a laptop to many different network locations and making appropriate code changes as needed.

Once the recognizer was stable and reliable, the rest of the project was developed using typical iterative methods. The graphical user interface, profile objects and action executors presented mere problems of construction and did not require the same sort of exploratory research as the recognizer.

## Summary of Project & Discussion of Results

When first loaded, NetShuffler displays a graphical user interface (GUI) that initially lists each active network currently visible from the computer. The user can then choose to create a profile for one or more networks, and within that profile, create actions to be executed when the network is connected. These actions can include mapping a network drive, setting the default printer, connecting to a VPN, or running any arbitrary program.

To recognize different networks reliably, the user is given the choice of several detection methods. The issue here is that many networks share identical settings. For example, recognizing networks exclusively by IP address would mistakenly identify home WiFi networks as the same network, because many such networks use the default range 192.168.0.x. By giving the user the choice of multiple methods, we make it more likely that some unique data can be found on each network of interest.

Once a network is recognized and actions are to be performed, it is important that these actions not disturb the user’s work. To accomplish this, each action is executed on a separate multiprocessing thread, with no user involvement or interaction. This should result in a truly “set and forget” user experience. Once profiles have been created, the software should just produce the desired configuration at each network location, without requiring any cognitive involvement on the part of the user.

## Relationship of Project to the Field

NetShuffler solves, for Windows, the same problem that the commercial product Sidekick solves for the Macintosh. No existing product for Windows offers silent, “set and forget” profile-based management of network configuration. As a result, NetShuffler provides a new capability, not previously available to Windows users.

## Conclusions

NetShuffler was initially developed to “scratch an itch,” which is to say, to solve a personal problem faced by the software developer. Over the course of developing the software, a number of people have indicated an interest in receiving a copy of the final result. This gives some informal and tentative indication that a market may exist for this product. The developer hopes to continue improving the software, either in the context of a commercial product, or as an open source project.

# Annotated Bibliography

Herlein, I. (2010, December 1). *NetSetMan Help File*. Retrieved November 17, 2013, from NetSetMan Pro: http://www.netsetman.com/index.php?s=help&hf=en  
Home page for the NetSetMan network configuration software.

Lund, S., Manyika, J., & Ramaswamy, S. (2012, November). *Preparing for a new era of work*. Retrieved November 17, 2013, from McKinsey & Company: http://www.mckinsey.com/insights/organization/preparing\_for\_a\_new\_era\_of\_work  
Summarizes the changes that are occurring in the demographics and nature of work, and the changes employers will have to make in response to them.

Microsoft Corporation. (2009a, December 7). *Network Location Awareness*. Retrieved November 17, 2013, from Microsoft TechNet: http://technet.microsoft.com/en-us/library/cc753545(v=ws.10).aspx  
Describes and documents the Network Location Awareness system as implemented in Windows Vista and Windows 7.

Microsoft Corporation. (2009b, February 24). *Tip: Set Different Default Printers for Different Locations*. Retrieved November 17, 2013, from TechNet Magazine: http://technet.microsoft.com/en-us/magazine/dd542629.aspx  
Provides a "tip" showing how you can configure Windows 7 to change the default printer based on network location.

Microsoft Corporation. (2011, February 8). *Network Awareness in Windows XP*. Retrieved November 17, 2013, from Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/ms700657(v=vs.85).aspx  
Describes the Network Location Awareness system as shipped with Windows XP. This was the first release of the NLA system.

Microsoft Corporation. (2013a, October). *Windows lifecycle fact sheet*. Retrieved November 17, 2013, from Windows: http://windows.microsoft.com/en-us/windows/products/lifecycle  
Lists current versions of Microsoft Windows operating systems, their release dates, and the dates on which support for them will end.

Microsoft Corporation. (2013b, Undated). *System.Net.NetworkInformation Namespace*. Retrieved November 19, 2013, from Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/gg145039(v=vs.110).aspx

Microsoft Corporation. (2013c, October 12). *Network List Manager*. Retrieved November 19, 2013, from Windows Dev Center - Desktop: http://msdn.microsoft.com/en-us/library/windows/desktop/aa370803(v=vs.85).aspx

Milner, D. (2011, May 17). *netprofiles*. Retrieved November 17, 2013, from Google Code: https://code.google.com/p/netprofiles/  
Home page of the Net Profiles network settings management software.

Oomph, Inc. (2012). *Sidekick: Geo-intelligent laptop settings*. Retrieved November 17, 2013, from Oomph: http://oomphalot.com/sidekick/  
Product page for the Sidekick software package for Mac OS X by Oomph, Inc.

TIOBE Software. (2013, November 10). *TIOBE Programming Community Index for November 2013*. Retrieved November 17, 2013, from TIOBE Software: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html  
TIOBE produces a monthly listing of programming languages. It uses search engines, job sites and other sources to estimate the relative popularity of each language.

VMware, Inc. (2013, October 24). *Configuring Host-Only Networking*. Retrieved November 19, 2013, from VMware Workstation 10 Documentation Center: http://pubs.vmware.com/workstation-10/index.jsp#com.vmware.ws.using.doc/GUID-93BDF7F1-D2E4-42CE-80EA-4E305337D2FC.html  
A chapter from the user documentation for VMware Workstation describing how to configure host-only networking using host virtual adapters.