University of Southern Queensland
Faculty of Engineering and Surveying

Development and Testing of a Networked PC Power Management Tool

A dissertation submitted by

Simon Brooks

in fulfilment of the requirements of

Courses ENG4111 and ENG4112 Research Project

towards the degree of

Bachelor of Engineering (Computer Systems)

Submitted: October 2010

Abstract

One of the major themes of this project is Personal Computer (PC) power usage, which is a significant issue as many PCs run 24 hours per day and therefore potentially waste large amounts of power. This power wastage is unfortunate given that most operating systems support hibernation

As issues such as fossil fuels, carbon footprint, global warming etc., are highly topical and politicized, energy efficiency in electrical devices is a significant issue. Additionally, through capability developments such as 'green' computing, virtualization, and even edge computing, the Information and Communications Technology (ICT) industry is one of the few industries that seem to be taking ecological issues seriously.

The main objectives of the project are:

- The research and evaluation of a number statistics that relate to PC energy consumption.

- Investigation of power spikes and network traffic floods which may be caused by simultaneous PC start-up.

- Development of a software tool to control the start-up and shutdown of a PC either via scheduling or manual control.

- Development of a software tool that remotely controls the start-up and shutdown of either singular or grouped PCs.

In addition to the aim of reducing wasted power, this project aims to improve network traffic efficiency by minimizing network traffic congestion through controlling the sequence of PC start-up in a networked environment. This controlled

start-up has a secondary benefit in potentially reducing the severity of power spikes due to a simultaneous PC start-up.

It is hoped that further development and testing of the ‘PowerMan’ application will provide greater program functionality. Additionally, it is envisaged that this tool could be used in conjunction with more energy efficient PCs and peripherals and therefore provide an overall package that is power efficient whatever its state of operation, with little control or interaction from the user.

Certification

I certify that the ideas, designs, and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Simon Brooks

Student Number: 0011120394

____

Signature

____

Date

Acknowledgements

This project was carried out under the supervision of Dr Alexander Kist and I would like to thank Alexander for his guidance and support during the project.

I am also extremely appreciative for the support I have gained from my loving and growing family. In particular, I would like to thank my wife who has supported me over many years while I completed this program.

Table of Contents

Appendices

List of Tables

List of Figures

List of Abbreviations

ACPI	Advanced Configuration and Power Control Interface
API	Application Programming Interface
ARP	Address Resolution Protocol
BCL	Base Class Library
BIOS	Basic Input Output System
CLR	Common Language Runtime
CPU	Central Processing Unit
DLL	Dynamic Link Library
GUI	Graphical User Interface
ICT	Information and Communications Technology
IDE	Integrated Development Environment
IP	Internet Protocol
IT	Information Technology
LAN	Local Area Network
MAC	Media Access Control
MPS	Multiprocessor Support
OS	Operating System
PC	Personal Computer
PM	Project Manager / Project Management
PMI	Project Management Institute
PMP	Project Management Plan
ROI	Return On Investment
RTC	Real Time Clock
TCP	Transmission Control Protocol
UDP	Universal Datagram Protocol
WMI	Windows Management Instrumentation
WoL	Wake on LAN

Chapter 1. Introduction

1.1 Project theme and background

The chosen topic for this dissertation is the 'Development and Testing of a Networked PC Power Management Tool'. The primary theme of this project is Personal Computer (PC) power usage, which is a significant issue as many home and office PCs run 24 hours per day, and therefore potentially waste large amounts of power. This power wastage is unfortunate given that most Operating Systems (OS) support either hibernation (Kist 2009), or other forms of low power standby.

Building on the theme described above, the primary aim of this project was the development of simple software tool that would control the start-up and shutdown of PCs via a user defined schedule. The developed software tool comprises of both a localised client control capability in addition to a functionality allowing a networked or supervisory control of a scheduled PC start-up, and also singular remote PC shutdown capability.

In addition to the broad development objectives outlined above, the project has also included research into the following areas:

- Average PC energy use and related statistics in common operating environments.

- Impact of simultaneous PC start-up on network traffic, and mains power supplies.

- The environmental impact of generated energy.

Given that issues such as fossil fuels, carbon footprint, global warming etc., are highly topical and politicized, there is quite a broad range of research material available that supports both the mainstream and extreme views on current environmental issues. Additionally, through capability developments such as ‘green’ computing, virtualisation, and even edge computing it would appear that the Information and Communications Technology (ICT) industry is one of the few industries that are actually taking ecological issues seriously.

This leading approach by the ICT industry which may be seen by many as unnecessarily restrictive given current laws, may in fact work in the industries long run favour. That is, by adopting a market leading approach on power efficiency now, it is likely that the ICT industry will not be as significantly effected should more rigid power related statutory regulations or governance be put in place in the future.

1.2 Dissertation layout and description

The approximate singular Full Time Equivalent (FTE) project hours against the major project functional categories are listed below to demonstrate the ‘time’ component of this project:

- Research – 150 hours
- Coding and testing – 200 hours
- Documenting and reporting – 50 hours

The following paragraphs provide a brief description of the layout of this dissertation:

- **Chapter 2** describes the primary project drivers and hence the rationale and associated benefits of the project.

- **Chapter 3** details the research into PCs and networks and forms the base level of knowledge that this project and subsequent application are built on.

- **Chapter 4** examines in detail the objectives for the project and subsequently provides some discussion on the rationale behind the objectives, thus aiding in the provision of a robust scope for the application development.

- **Chapter 5** looks at PC power usage and other network related statistics, which form the critical components of the research requirements for this project.

- **Chapter 6** details the project management methodologies and processes which are critical to complex software development, and hence this project.

- **Chapter 7** details the software development considerations including the development environment and methodology.

- **Chapter 8** provides a functional description of both the server and client applications, in addition to the key features.

- **Chapter 9** details the testing methodology, test plan and test results for the developed applications.

- **Chapter 10** details the project conclusions in relation to the project objectives, and also further work considerations related to the project.

Chapter 2. Project drivers

2.1 Introduction

This chapter describes the primary project drivers, and hence presents the rationale and the associated benefits for this chosen project.

2.2 Energy production

According to Silver (2008), the majority of energy utilised today is generated from the burning of fossil fuels. This combustion process which results in the by-product of carbon dioxide, is responsible for the increases in carbon dioxide in the atmosphere to levels 35 percent greater than 150 years ago. The exponential generation of carbon dioxide and other greenhouse gases is likely to continue throughout this century, and thus if the present strong belief by scientists and policymakers proves to be correct, then these gases will be a significant driver for global climate changes in the future (Williams & Balling 1996).

Figure 2.1 below provides an illustration detailing the breakdown of the biggest contributors to greenhouse gas emissions by sector, while Figure 2.2 illustrates the breakdown of the US energy sector by carbon dioxide emissions. From this it can be seen that the burning of coal in the production of energy is one of the major contributors to global greenhouse gas production.

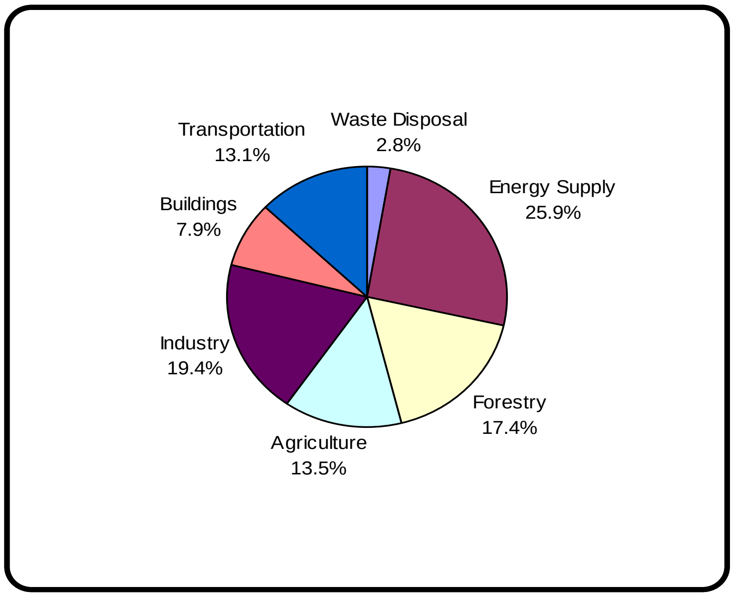


Figure 2.1 – Greenhouse gas emissions by sector

(Source: Silver 2008)

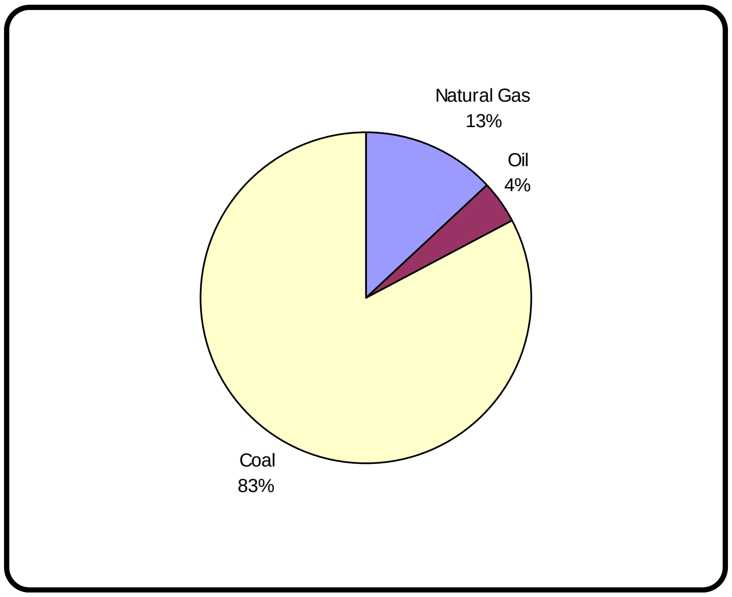


Figure 2.2 – Carbon dioxide emissions from US electricity generation

(Source: Silver 2008)

2.3 Energy usage

From above it can be readily seen how power consumption directly contributes to greenhouse gas emissions, and thus possibly global climate change. While greenhouse gas emissions and climate change are important topics within the broad context of this project, they are not the primary focus. The primary focus of this project is the potential for energy savings or the greater efficiency of a standard PC utilised in a home or small office environment. Table 2.1 below details the energy consumption for a PC. Importantly, it also compares energy usage between the different ‘states’ of the computer.

Appliance	Wattage Rating
CPU (awake/asleep)	120/30 or less
Monitor (awake/asleep)	150/30 or less
Laptop	50

Table 2.1 – PC electrical energy usage
(Source: Silver 2008)

Table 2.2 below provides a more detailed breakdown of PC power usage in terms of common PC variants and peripherals. This highlights that the actual PC is not the only factor in the overall power saving consideration, but just as important are the attached peripheral devices that often spend their life in a ‘standby’ mode.

Appliance	Energy Use (watts)
Desktop CPU	100
LCD Monitor (15-17 inches)	50-150
Laser printer	100
Inkjet printer	12
Multifunction printer/copier	15
Laptop	22
Wireless router	6
Computer speakers	7
USB hub	3

Table 2.2 – Average Power Consumption: Computer and Peripherals
(Source: Leonhard & Murray 2009)

Figure 2.3 below complements the previous graphics detailing greenhouse gas and carbon dioxide production, and importantly illustrates the energy usage according to industry sector.

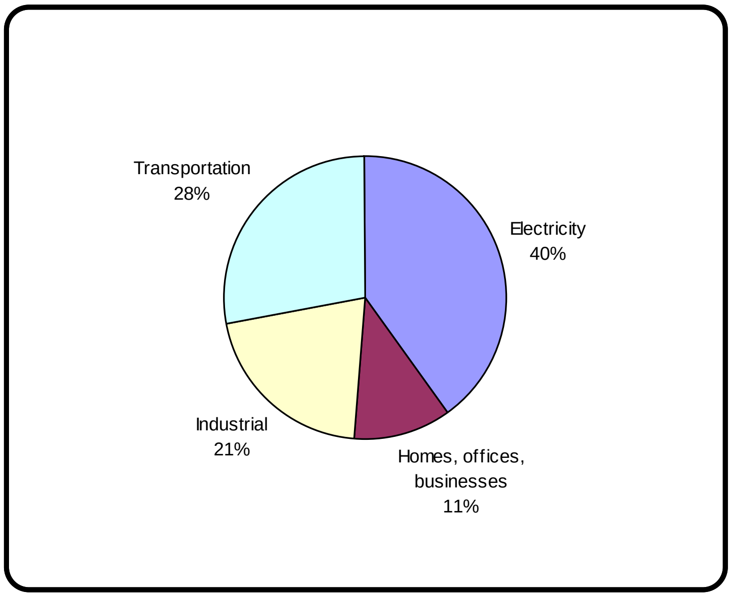


Figure 2.3 – Energy usage by sector
(Source: Leonhard & Murray 2009)

Using the abovementioned figures and illustrations, it can be seen how switching to a 'sleeping' (or like) state of the PC can result in an energy usage of approximately one quarter of the normal usage. Furthermore, if we consider that a hard shutdown or power off will have a wattage usage rating of zero, then the energy and potential cost savings are even more significant.

2.4 Significance of PC energy consumption

Although it may at first seem that PC power usage is insignificant in terms of magnitude of the issues described above, given that Phillipson (2008) suggests that 67.81% of Australian homes have a desktop PC, 40.66% of homes have a laptop, and at least 75.80% of homes having either a desktop PC or laptop, it can be seen that the combined power usage for PCs will be significant. Furthermore, when the use of PCs within the modern workplace is also considered, the aggregated power usage for all PCs, whether home or office based, will be even more significant.

2.5 Network transmission efficiency

Another component of this project is the management of networked PCs start-up in order to minimise the potential for network traffic problems. According to Addie (2007), network traffic can be affected by either data loss or delay, with the three major causes of delay being:

- Propagation delay,
- Transmission delay, and
- Queuing delay.

Given that propagation delay is due to the physical transmission distance (Held 1996), it is likely to be insignificant when compared to other forms of delay in a small office environment. Furthermore, transmission delay which is the delay due to the time taken to feed a signal onto the communications medium (Slone 2000), is likely to be much more significant than propagation delay over short distances.

Last but by no means least is queuing delay, which is the delay due to storage and retransmission of data within a network (Hancock 1990). Queuing delay is likely to have some impact within a small office environment depending upon the amount and type of data switching that takes place within the network.

In addition to above, data arriving at a destination will have to be either buffered or discarded in cases where the destination device is busy. Furthermore, where a buffer reaches maximum capacity, discarding of data will also occur. Where this data is discarded, it is effectively lost, and thus is required to be retransmitted, which then places an additional traffic load on the network. Lewis (2008) suggests that another significant factor in data corruption or data loss is network collisions, which are common on an Ethernet network, and are the result of two hosts transmitting frames simultaneously.

Combining the above, it can be seen that both buffering and collision issues are likely to arise when multiple devices attempt to communicate on the network at the same time, and made worse where communication is with a common or shared device. This combination of collision and buffering issues is then likely to result in data corruption and/or loss, and therefore retransmission. This will ultimately result in increased networked traffic.

Simultaneous PC start-up would be one such network event where multiple devices attempt to communicate over the network at the same time, possibly with the same destination device. Consequently, a better management of start-up, which included a phased PC start-up sequence, is likely to reduce network traffic floods due to a reduction in data retransmissions caused by data collisions and/or data loss by buffers.

2.6 Efficient and consistent power draw

The final major driver for the project is efficient power consumption, with a focus on the minimisation of controlled events which could contribute to a power spike. A power spike is a short duration high voltage condition (Smith 2010) that occurs across a power circuit. Theses spikes can have a number of effects including damaging an electronic device's internal power supply in addition to damaging or degrading other sensitive components such as motherboards and processors.

While the majority of power spikes can be attributed to environmental factors such as storms or lightning, power spikes can also be caused by the switching of large electrical loads (ThomasNet 2010). In the case of a modern office environment where a vast number of PCs may be either started or shut down simultaneously, there is a frequent net effect of large electrical load being either applied or removed from a power circuit. This therefore increases the chances of a power spike being generated within the circuit.

Furthermore, while the switching of loads may cause power spikes, and hence damage to internal equipment, the switching of large loads will also impact upon the primary power feeder into an office or building. This could then have flow on effects of impacting upon the reliability or consistency of the power network.

Similar to above, a more controlled management of PC start-up, which included a phased PC start-up sequence, is likely to reduce the chances for power spikes. This reduction in power spikes, which in the in the long run will minimise the power related damage to network hardware, will also provide greater availability for the overall network.

Chapter 3. A brief review of PCs and networks

3.1 Introduction

The following paragraphs detail the research into PCs and networks and consequently form the base level of knowledge that the project and/or application is built on. Importantly, while the following paragraphs contain a significant amount of the overall project research, the remaining research is provided within the remaining dissertation chapters.

3.2 Advanced Configuration and Power Interface

The Advanced Configuration and Power Interface (ACPI) is a critical component of this project. According to Hewlett-Packard et al (2010), the Advanced Configuration and Power Interface (ACPI) was developed to provide industry standard interfaces. Furthermore, these interfaces allow: Operating Systems (OS) directed device configuration; and power management of both single devices and entire systems.

The ACPI was built on the legacy power management systems and processes including: power management Basic Input Output System (BIOS) code, Advanced Power Management (APM) Application Programming Interfaces (APIs) and Multiprocessor Specification (MPS) tables in order to provide a structured and well defined configuration and power management interface configuration (Hewlett-Packard et al 2010).

One of the key points justifying the creation of the ACPI according to Hewlett-Packard et al (2010) is that through the transfer of the power management into the OS through the use of an abstract interface, the OS can evolve independently of the hardware, and similarly the hardware can evolve independently of the OS.

The ACPI has a number of state definitions which include expected or designated performance criteria. The categories of the ACPI state definitions are:

- ACPI Global states,
- ACPI Device states,
- ACPI Sleeping states,
- ACPI Processor power states, and
- ACPI Device and processor performance states.

The states relevant to this dissertation and subsequently described in the following paragraphs are:

- Global states,
- Sleeping states, and
- Processor power states

3.2.1 ACPI Global States

The ACPI has a number of global state definitions which are visible to the user. These global states are described together with some relevant criteria in table 3.1 below

Global State	Software Runs	Power Consumption	OS Restart Required
G0 Working	Yes	Large	No
G1 Sleeping	No	Smaller	No
G2/G5 Soft Off	No	Very Near 0	Yes
G3 Mechanical Off	No	Real Time Clock (RTC) Battery	Yes

Table 3.1 – ACPI Global States
(Source: Hewlett-Packard et al 2010)

3.2.2 ACPI Sleeping States

The ACPI has a number of sleeping states which reside within the global ‘G1’ state. The ACPI sleeping states are described in table 3.2 below

Sleeping State	Description
S1 – sleeping state	In this state, no system context is lost (CPU or chip set) and hardware maintains all system contexts.
S2 – sleeping state	This state is similar to the S1 sleeping state except that the CPU and system cache context is lost
S3 – sleeping state	The S3 sleeping state is a low wake latency sleeping state where all system context is lost except system memory. CPU, cache, and chip set context are lost in this state.
S4 – sleeping state	The S4 sleeping state is the lowest power, longest wake latency sleeping state supported by ACPI. In order to reduce power to a minimum, it is assumed that the hardware platform has powered off all devices.
S5 – soft off	The S5 state is similar to the S4 state except that the OS does not save any context. The system is in the “soft” off state and requires a complete boot when it wakes.

Table 3.2 – ACPI Sleeping States
(Source: Hewlett-Packard et al 2010)

A more simplistic and perhaps pragmatic view of the above ACPI sleeping states is provided in table 3.3 below. From this it be easily seen that state S0 represents a fully operational PC power state, with states S1-S4 providing various states of reduced power and functionality when compared with state S0. Noteworthy are the use of the terms ‘standby’ and ‘hibernate’ which respectively relate to states S3 and S4. These states will be discussed in more detail in a later chapter.

Sleeping State	Description
S0	On and fully operational.
S1	System is in low power mode (a.k.a. sleep mode). The CPU clock is stopped, but RAM is powered on and being refreshed.
S2	Similar to S1, but power is removed from the CPU.
S3	Suspend to RAM (a.k.a. standby mode). Basically, most components are shutdown except RAM.
S4	Suspend to disk (a.k.a. hibernate mode). The memory contents are swapped to the disk drive, and then reloaded into RAM when the system is awakened.
S5	Power off.

Table 3.3 – Practical View of ACPI Sleeping States
(Source: Intel 2010)

Figure 3.1 below provides an illustration of the transitions between the aforementioned ACPI sleeping states. From this illustration, the following can be noted:

- **Wake event** – This will raise the system to a working of S0 'full-on' state from the 'S1-S4' sleeping states.

- **Idle related event** – An event generated from measured idle time could transition the system from the S0 'full-on' state to the 'S1-S4' sleeping states.

- **Power switch** – A OS generated, or user initiated, switch can transition a PC as follows:

  - Between the S5 or 'soft-off' states and the S0 'full-on' state, and
  - Between the 'S1-S4' sleeping states and the S0 'full-on' state

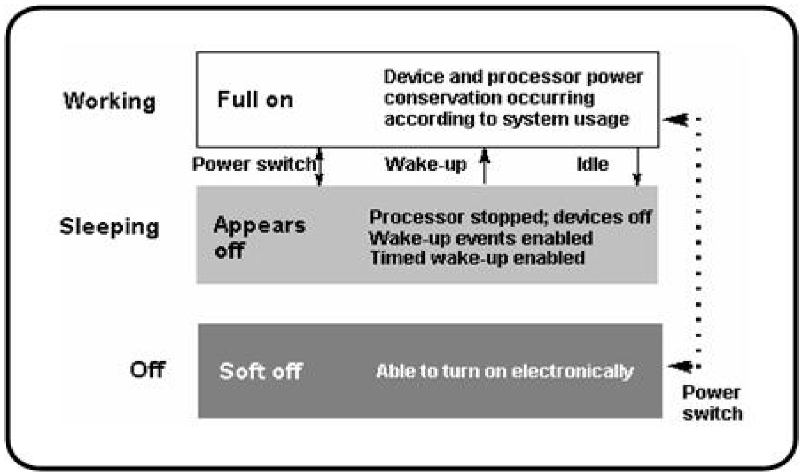

Working
Full on
Device and processor power conservation occurring according to system usage
Power switch
Wake-up
Idle
Sleeping
Appears off
Processor stopped; devices off
Wake-up events enabled
Timed wake-up enabled
Off
Soft off
Able to turn on electronically
Power switch

Figure 3.1 – System states and transitions

(Source: MSDN 2010)

3.2.3 ACPI Processor Power States

The ACPI also has a number of processor power states which reside within the global 'G0' state. The ACPI sleeping states are described in table 3.4 below

Processor State	Description
C0	While the processor is in this state, it executes instructions.
C1	Aside from putting the processor in a non-executing power state, this state has no other software-visible effects.
C2	The C2 state offers improved power savings over the C1 state. The worst-case hardware latency for this state is provided via the ACPI system firmware and the operating software can use this information to determine when the C1 state should be used instead of the C2 state.
C3	The C3 state offers improved power savings over the C1 and C2 states. The worst-case hardware latency for this state is provided via the ACPI system firmware and the operating software can use this information to determine when the C2 state should be used instead of the C3 state.

Table 3.4 – ACPI Global States

(Source: Hewlett-Packard et al 2010)

3.3 Operating System Considerations

The following paragraphs cover some of the noteworthy points relating to the Windows Management Instrumentation (WMI) and the Windows Application Programming Interface (API)

3.3.1 Windows Management Instrumentation

Windows Management Instrumentation (WMI) is provided on Windows based operating systems and is the infrastructure for management data and operations on such systems. Importantly, WMI scripts or applications can be created to automate administrative tasks on remote computers (MSDN 2010). Furthermore, WMI has been designed for programmers who use the C, C++, C# languages within Microsoft’s Visual Studio.

In simple terms, the WMI is an OS interface that provides information and notifications. This then allows for the local and remote management of Windows computers and servers.

3.3.2 Application Programming Interface

According to Silberschatz, Galvin and Gagne (2005), the Application Programming Interface (API) details a list of functions that are made available to a programmer. Included in this list are the parameters that are passed to each function and also the value types that are expected to be returned. The three most common APIs are:

- Win32 API for Windows Systems,
- POSIX API for POSIX-based systems, and
- Java API for programs that run on the Java virtual machine.

Given that Windows is the OS that will be used as a basis for the development of the project software tool, only the Windows API is relevant to this dissertation. According to MSDN (2010), the Windows API allows for applications to exploit the full power of Windows. Furthermore, the Windows API (formerly referred to as the Win32 API) consists of the following functional categories:

- Administration and Management,
- Diagnostics,
- Graphics and Multimedia,
- Networking,
- Security,
- System Services, and
- Windows User Interface.

3.4 Wake on LAN

The Wake on (Local Area Network) (WoL) functionality is an important consideration in this project. According to Intel (2010), the remote management of PCs has evolved over the last few years from a simple power-on function to a complex system that considers a variety of device and OS power states.

3.4.1 Wake on LAN Details

WoL is a functionality of some network interface devices that causes a PC to 'wake' from an ACPI sleep state. WoL is very much hardware dependant and in most cases will allow the PC to wake from the S1 to S5 states. However, some devices will not respond or 'wake' when in an S5 state. The WoL process requires the transmission of a 'magic packet', which is a packet that begins with a six byte sequence of 'FF' bytes. This sequence is then followed by the Media Access Control (MAC) address of the network device that is being prompted to 'wake', repeated 16 times (Neumann 2008). The 'magic packet' composition is illustrated in figure 3.1 below.

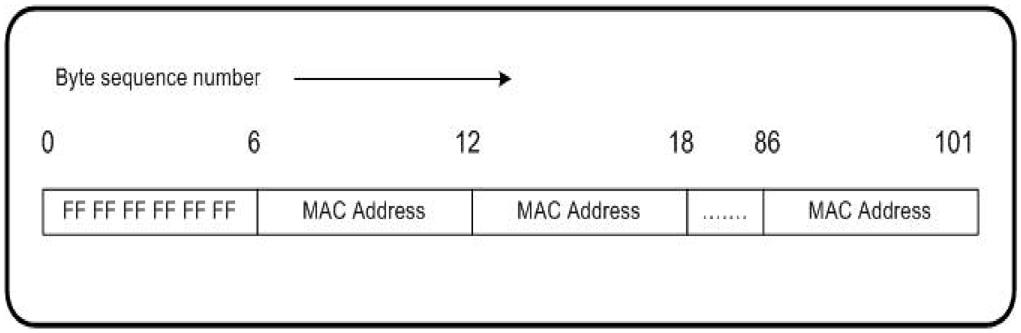


Figure 3.1 – 'Magic packet' composition

The MAC address, which is stored in the flash memory of network devices, is utilised as the device identifier as no other identifiers such as the Internet Protocol (IP) address are consistently available across the ACPI S1 to S5 sleeping states. The MAC address is repeated 16 times to ensure that the network device does not generate a ‘wake’ event for normal network traffic, but rather will only respond to a specific ‘wake’ directive.

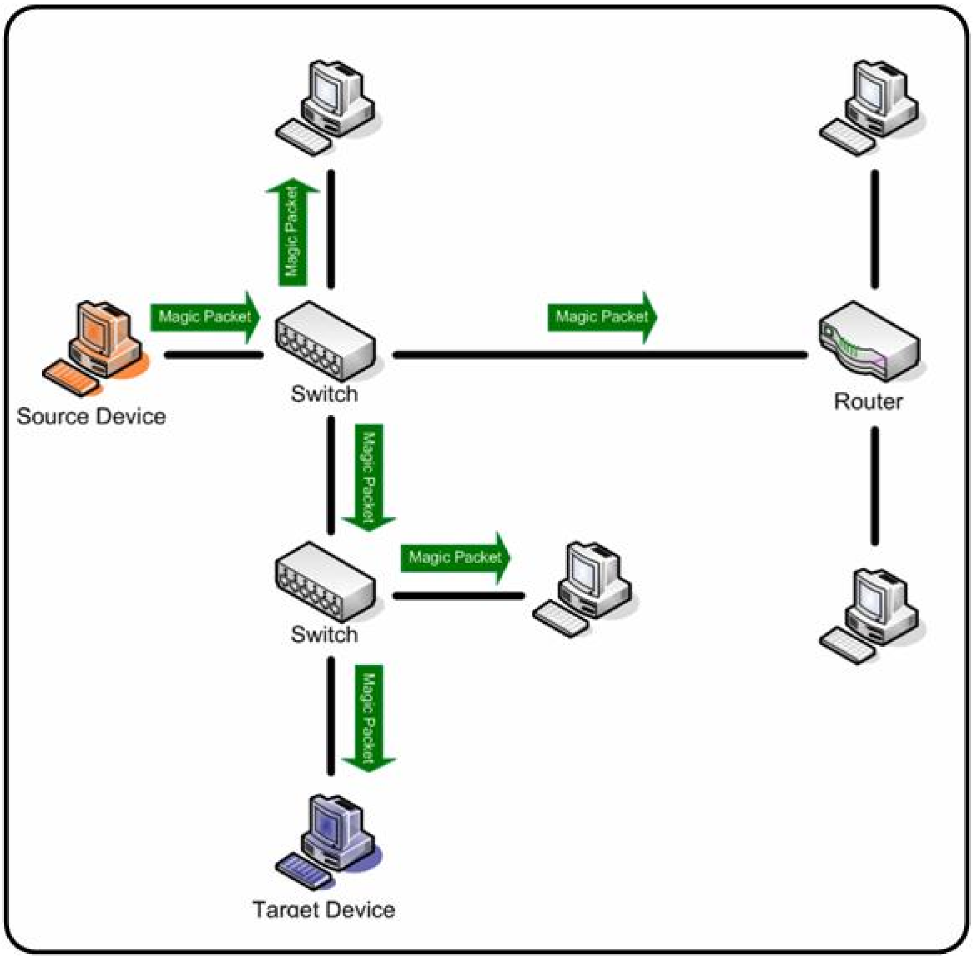


Figure 3.2 – ‘Magic packet’ transmission
(Source: Newman, 2008)

As illustrated in figure 3.2 above, the ‘magic packet’ is broadcast over the LAN causing it to reach all devices within the LAN. To do this, the ‘magic packet’ is captured in a User Datagram Protocol (UDP) broadcast packet, set for a network address of 255.255.255. The advantage of using UDP is that, as a connectionless protocol it does not require any handshaking in order to open a connection (Neumann 2008).

3.4.2 Wake on LAN Setting Considerations

As WoL performance is largely dependant on the network device, in order for it to operate, it requires that correct settings be established from within BIOS and also the OS. This is largely due to the fact that while WoL can be initially selected within BIOS, modern operating systems making use of ACPI features allow for the configuration of individual components (Intel 2010).

Experimentation with the WoL functionality has demonstrated that the following components must be checked and/or selected in order for WoL to function correctly:

- The device network card must support WoL.
- The networked device power supply must support WoL.
- WoL must be enabled in BIOS.
- The operating system must be configured to enable WoL.

Furthermore, trial and error throughout this project has shown that in order for WoL to have a consistent capability, then the BIOS settings must ensure that after a mains power loss, the system returns to the 'last state'. Expanding on this, a given BIOS will generally allow for three settings with respect to power management, after an AC mains power outage including:

- Power off,
- Last state, and
- Power on.

If the BIOS setting is configured for power off, the device will be in a 'hard off' state and therefore unlikely to respond to a WoL request. However, while the selection of the 'last state' increases the reliability of a WoL post a power outage, different PCs have demonstrated that operation post a power outage, or more specifically a return to an ACPI sleeping state of S1-S5 after a power outage, is largely unpredictable. In this case, the only way to guarantee WoL will operate correctly is to ensure that no power outages or 'hard offs' have taken place post a user or system initiated shutdown.

Chapter 4. Project objectives and scope analysis

4.1 Introduction

Typical software design generally progresses from a user requirements phase through to a formal specification phase and then onto a design phase (Somerville 2007). However, this project in effectively commencing from a formal specification phase, makes user requirements somewhat redundant. When considering that user requirements form a significant consideration in software estimation and planning, and hence overall user satisfaction (Galorath and Evans 2006), detailed user requirements are an extremely valuable component for any software project.

In analysing the project objectives or specification which is included at Appendix A, we are able to broadly capture the intent and breadth of both user functional and non-functional requirements, and from this detail some of the envisaged benefits or outcomes from the project. Furthermore, as Wysocki, Beck & Crane (2000) suggest that project scope is significantly more than just requirements, the use of additional information within the project scope will lead to a more robust scope and hence maximise the chances of project success. Accordingly, the following paragraphs examine the objectives for the project in detail, and subsequently provide some discussion on the rationale behind the objectives.

4.2 Project objectives and scoping

Specification Item 1 *- Research and evaluate the average power use, idle time and other statistics of a PC that relate to power consumption in a standard office environment.*

Through the examination of the PC idle time statistic, an insight into how PCs are managed can be gained, that is, how the PC start-up and shutdown functions are utilised in relation to both the duration and frequency of user driven or automated tasks performed on the PC. Consequently, this then allows the examination of whether an automated approach to start-up and shutdown will provide any advantages in relation to the ratio between the idle and active times of the PC. Using these idle time figures, in addition to the statistics on average PC power use, the generation of a number of statistics detailing actual amounts of power used and wasted by an average PC in a standard office environment can be calculated.

Specification Item 2 *- Research the potential for, and severity of power spikes, network traffic floods and other issues in a standard office environment due to simultaneous PC start-up.*

Where PCs in an office environment are started at the same time, for example, at the commencement of a working day, a significant load may be placed on the local power grid, potentially causing power spikes. Through more controlled management of the start-up sequence computers on the same power circuit, the potential for and severity of power spikes may be reduced. Additionally, within a networked PC environment, issues may be caused where PCs attempt to start communicating on the network at the same time, possibly resulting in a network traffic flood. Through the use of a phased start-up, possibly together with some sort of primitive flagging system, the number of PCs attempting to establish communication with the network simultaneously may be able to be better managed, and thus reduce network traffic flood problems.

Specification Item 3 *- Develop a software tool that controls start-up and shutdown of a PC either via scheduling or manual control.*

Given the issues raised in sections items 1 and 2, it can be understood that the random manual switching of a networked PC potentially raises a number of power efficiency issues, in addition to possibly creating network traffic issues. Through the provision of a software tool that controls PC shutdown in accordance with user or network defined conditions, then a more balanced network traffic load is possible together with more efficient power management within a network.

Specification Item 4 *- In conjunction with the local software tool described in component 3, develop a software tool that remotely controls the start-up and shutdown of either singular or grouped PCs (within a designated domain) over a Local Area Network (LAN).*

Through building upon the functionality of the local start-up and shutdown tool described above via the inclusion of a networked control capability, the management of PC start-up and shutdown can be removed from the user, and therefore reside with a nominated administrator. In addition to freeing the user from managing this task, the use of a centralised start-up capability could potentially provide some benefits due to the ability of the administrator to recognise usage patterns over the entire network, and thus implement a start-up and shutdown schedule that is the most beneficial for the entire network.

Specification Item 5 *- In addition to specifications outlined in components 3 and 4 above, the software tool shall:*

a. Implement a phased start-up (to prevent power spikes) where multiple PCs on a single LAN may be scheduled to start-up.

b. Presentation of software tool options, controls, and critical usage statistics within a user friendly Graphical User Interface (GUI).

A centralised phased start-up will reduce peak demand on shared power sources, and also aid in distributing network traffic to prevent traffic floods. The inclusion of a GUI in addition to an intuitive based format will provide ease of use for the PC user or network administrator. Additionally, the publication of usage trends and statistics may provide the user with some incentive or motivation towards managing their PC usage more efficiently.

Specification Item 6 *- Evaluate and test the tool in a small office/network environment*

This testing phase will be incorporated within the development phases under an agile development methodology, which according to Somerville (2004) relies on an iterative approach to software development. This will effectively allow the trial of features etc. as they are implemented and added rather than at the end of development. However, it is most likely that the small office testing will be conducted in two major stages: Firstly, the testing at the completion of local application; and Secondly, the second stage of testing occurring at the end of the development of the network component of the tool.

Specification Item 7 *- Evaluate and test the tool in a large office/network environment.*

This will provide a test environment that replicates the intended usage environment, and thus effectively indicate how well the tool performs in its intended role. As this is on an 'as time permits' basis, and not a critical component of this project, planning for and testing within this environment will only occur (if at all) during the final stages of the small office testing for the network component of the tool.

Specification Item 8 *- Translate 'wasted power measurements', and also saved power with the use of the tool into figures relating to a 'carbon footprint' and greenhouse gas measurement.*

By translating the project technical findings and certain features of the projects functionality into terminology that is presently highly topical and also more common, the benefits of the tool and dissertation are likely to reach a wider audience. That is, it is hoped that the benefits of a tool of this type will disseminate well beyond the technical environment, and therefore hopefully reach PC users in all environments.

Chapter 5. Project statistics and outcomes

5.1 Introduction

According to Bigelow (2003), PC power as a topic, is not simply an issue about the connection of a small silver box. With global concerns about limited resources and greenhouse gasses, there is a growing focus on the power that millions of PCs world wide consume, and perhaps more significantly waste, while not being actively used.

Before looking at general PC power use statistics it is worthwhile understanding the different types of power supplies that power modern PCs and also the improvement in these systems over recent years. There are two primary categories of PC power supplies:

- Linear power supplies, and
- Switched regulator supplies.

Linear power supplies can often waste half of the input power through heat dissipation, thus making them tremendously inefficient. While switched power supplies through the use of feedback loop effectively switch the input voltage off and on to maintain a steady output, and thus minimise power wasted as heat. (Bigelow 2003).

In any case, whether utilising efficient or inefficient power supplies, the global focus on greenhouse gas emissions is placing pressure on the corporate environment to ensure that power wastage is significantly reduced through the ‘switching off’ of devices when not in use.

5.2 Computer usage statistics

5.2.1 Computer power status

Figure 5.1 below illustrates the observed power status for a number of common office machines including PCs during a survey conducted on an average office environment, during working hours. Importantly, it can be seen that approximately 55% of computers were in an 'on' status during the survey.

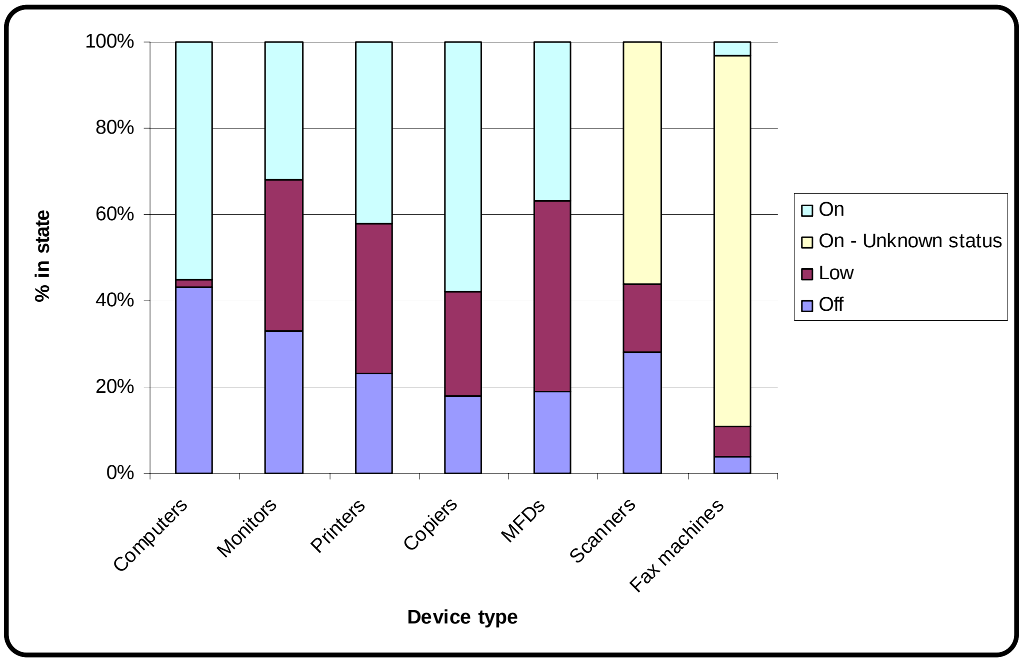


Figure 5.1 – Power status by equipment type

(Source: Webber et al. 2001)

Building on above, table 5.1 details the results from a number of surveys looking at PC power status for an out-of-hours period. In light of these results together with previously detailed PC power usage statistics, the ability to automate a switch to a PC power state that has a reduced power draw, has significant merit.

Study	Sample Size	On	Low Power	Off
Syzdlowski & Chvala (1994)	182	18%	--	82%
Tille & Newsham (1993)	94	20%	--	80%
IHEM (1994)	30	47%		53%
Nordman et al. (1996)	70	26%	6%	69%
LBNL-Forrestal	~200	10%		90%
CADDET (1999)	307	>99%	<1%	0%
Arney & Frey	20	25%	--	75%
Nielson (1998)	373	11%		89%
Picklum et al. (1999)	904	17%	9%	73%
Nordman (2000)	154	9%	0%	91%

Table 5.1 – PC Night Status

(Source: Adapted from Webber et al. 2001)

5.2.2 Energy financial cost estimates

Table 5.2 below details the charges for a number of Ergon Energy advertised energy tariffs. Although the tariffs are quite detailed, they are included to illustrate the approximate financial cost of energy, in addition to the variety of pricing schemes available. Based on this, an arbitrary cost of 17 cents per kilo-Watt-hour (kWh) is used for the financial cost calculations within the project.

Tariff	Condition	Cost (cents per kWh)
20 – General supply	All consumption	23.925
21 – General supply	First 100 kWh per month	29.722
	Next 9,900 kWh per month	27.918
	Remaining usage	21.252
22 – General supply (time of use)	Low rate	10.241
	All other consumption High Rate	29.073
31 – Night rate (super economy)	All consumption	8.712
33 – Controlled supply (economy)	All consumption	12.826
37 - Non-Domestic Heating Time of Use (Obsolescent)	From 10.30pm to 4.30pm (18 hours)	12.727
	From 4.30pm to 10.30pm (6 hours)	31.823
41 - Low Voltage General Supply Demand	All consumption	7.414
43 - General Supply Demand Time of Use	From 7.00am - 11.00pm Mon - Fri	15.08
	Other Times	6.028

Table 5.2 – Common energy tariffs and costs
(Source: Ergon Energy, 2010)

5.2.3 Energy environmental cost estimates

Table 5.3 below details the estimated carbon dioxide ‘costs’ for coal produced energy. As with the above energy financial costs, they are included to illustrate the estimated environmental cost of energy, in addition to the variety of estimates available for carbon dioxide ‘costs’. Based on this, an arbitrary cost of carbon at 0.956 kg per kWh is used for the energy production environment cost calculations within the project.

Source	Cost (carbon (kg) per kWh)
My Clean Sky (2010)	1
Wiki (2010)	0.963
Department of Energy (2000)	0.960
Parliamentary Office of Science and Technology (2006)	>1
Carbon Dioxide (2010)	0.966

Table 5.3 – Estimates for ‘carbon cost’ of coal fired energy

5.2.4 Idle time statistics

Idle time is the time that a PC spends with no direct user interaction. According to a study by Kawamoto et al. (2004), the average computer idle time in any given day is 3.9 hours. By specifying a variable delay, after which computers enter a low power state, the idle time in a high power state can be reduced.

Table 5.4 below details study related findings that show the time that PCs spend in various power states without user interaction. Additionally, the table illustrates the effects when an idle time delay is utilised in order to switch the PC to a power state with a reduced power draw. Furthermore figure 5.2 below illustrates that, the greater the time selected for ‘idle time delay’, then the greater the time the PC will sit idle in an ‘active’ or high power mode.

Idle Time Delay (minutes)	Idle Time in Active Mode (minutes)	Idle Time in Low Power Mode (minutes)	Time is Low power Mode (minutes)
5	0.9	3	76
15	1.9	2	51
30	2.6	1.3	34
60	3.1	0.8	20

Table 5.4 – Idle time delay and effect on PC power state

(Source: Kawamoto et al, 2004)

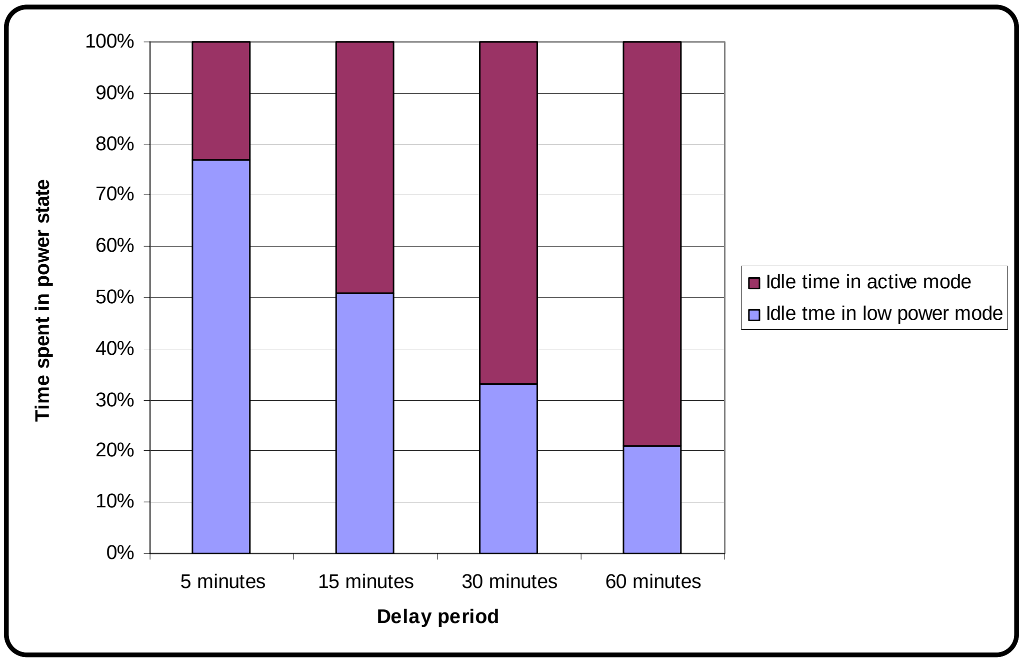


Figure 5.2 – Idle time delay effect on PC power state

(Source: Kawamoto et al, 2004)

5.2.5 Network traffic floods

A network traffic flood is the saturation of the network with too much traffic, which has the effect of slowing down transmission speeds over the network. Traffic shaping is typically utilised to control the volume of traffic on the network through the imposition of delays on selected packets (Wikipedia 2010). Accordingly, the controlled phased switching of network devices could be considered a primitive form of traffic shaping.

It is important to note that network shaping is significantly different to traffic policing. Traffic policing is the dropping or remarking of selected packets during peak traffic load times (Cisco 2010). One of the significant advantages of traffic shaping, or in fact any sort of network traffic load control, is the even distribution of network traffic over time, rather than a series of extreme peaks and troughs.

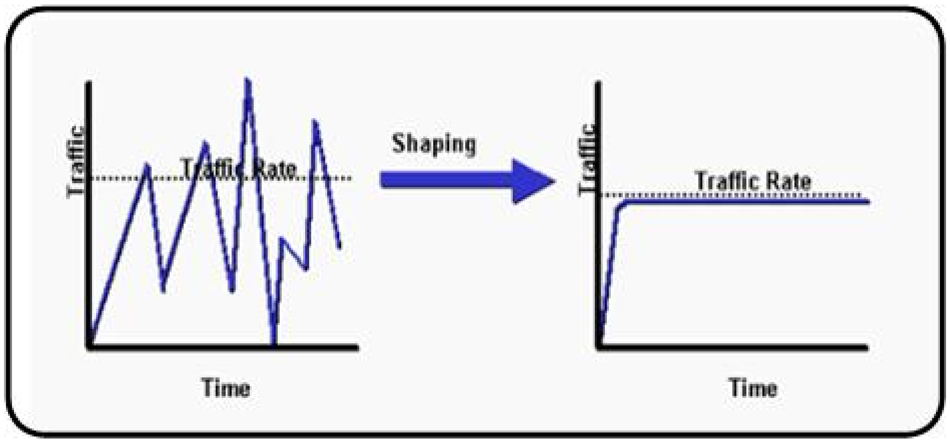


Figure 5.3 – Effects of network traffic shaping
(Source: Cisco, 2010)

5.2.6 Power spikes

As explained previously, power spikes are generally caused by environmental conditions such as lightning, or are due to the switching of large electrical loads. According to Interstate Assessment Technologies (2010) most semi-conductor devices are intolerant of voltage transients in excess of their voltage rating.

Large scale power spikes can cause instantaneous damage such as short circuits, breakdown or melting of insulation, and possibly fire. Looking past immediate physical effects, both large and small scall transients can have the following effects:

- Causing semi-conductor devices to fail, and
- Degradation of semi-conductor devices leading to a shorter life.

Furthermore, transient impulses can impact upon memory. This can then lead to Random Access Memory (RAM) alteration and failure, in addition to pitting and metal transfer between switch contacts on disc-drives (Interstate Assessment Technologies). While the predominate measure to guard against power transients is the use of power filtering or surge protection, this is very much a reactive measure. Similar to the rationale behind the controlled phased switching of network devices mentioned above, controlled and automated PC switching could be considered more of a proactive or initial measure against voltage transients.

5.3 Environmental and financial impacts

As Rowsell-Jones (2007) suggests the value of an ICT investment is measured in purely in terms of business performance only, it can be concluded that a congested or unavailable network will not provide an optimal return for the business. Therefore the following measures may have significant impact on overall ICT availability, and hence provide a greater financial return for the business:

- The efficient management of network traffic to minimise congestion and hence ensure that network traffic throughput is efficient as possible.

- The reduction of transient voltages applied to a network to ensure a greater availability of network devices, and also a greater Return On Investment (ROI), through an increased lifespan of network hardware.

It is also hoped that by translating the project energy consumption and production findings into terminology that is presently highly topical and also more common, the benefits of the tool and dissertation are likely to reach a wider audience. Given that engineers are often perceived to reside between pure science and wider society, then a translation of data into a more practical context seems worthwhile.

As discussed above, given the topical nature of energy production, projects that can potentially reduce energy wastage, and thus effectively contribute to reducing carbon emissions are likely to incur significant attention and hopefully further development. It is important to note however, that while this project improves PC power efficiency, it does not alter or improve a PCs power efficiency while operating, and thus the overall energy efficiency is still dependent on the actual energy consumption rate of the device being used.

I have included the above, not so much to highlight a limitation of the project, but rather highlight the pragmatic nature of the project, in that its main function is actually to reduce PC power wastage, not make an operating device more efficient.

Chapter 6. Project management

6.1 Introduction

While this dissertation is primarily intended to identify and detail a solution for a proposed problem or issue within an ‘academic’ environment, the methodology utilised to manage the solution development in this environment is just as critical in ensuring project success, as it is in the commercial world.

According to Kerzner (2003) a project can be likened to a series of activities that are subject to some or all of the following criteria:

- Defined objectives to be completed in accordance with specifications.
- Defined commencement and cessation dates.
- Have financial constraints.
- Consumption of resources.

With the above in mind, this project aligns with the contemporary definition of a formal project. Building on this, the appropriate management of the project ‘triple constraint’ of scope, time, and cost (Schwalbe 2006) is pivotal in ensuring project success. The relationship between the ‘triple constraint’ components is illustrated in Figure 6.1 below.

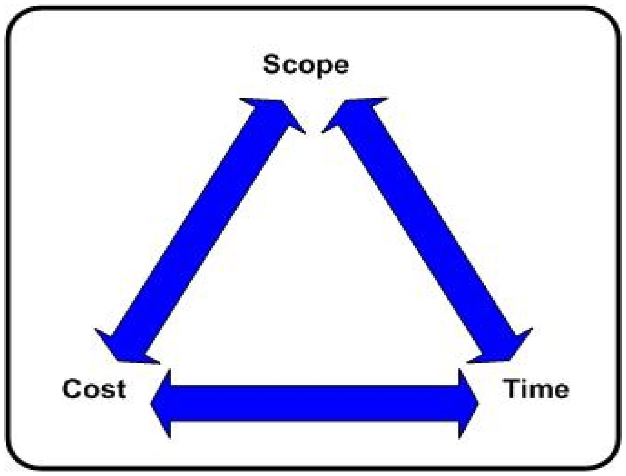


Figure 6.1 – Project ‘triple constraint’

6.2 Project methodology and lifecycle

As there are a number of project lifecycle methodologies, it is important to choose a lifecycle approach that aligns with the project’s scope, time and cost requirements, and in the case of software development, aligns with the intended development methodology. For this project, a relatively simple lifecycle approach, which is illustrated in Figure 6.2, was selected. Importantly, this project lifecycle contains clear and discrete phases. Furthermore, the project is deemed to be completed at the delivery point for the concept, which differs with some other long term lifecycle methodologies.

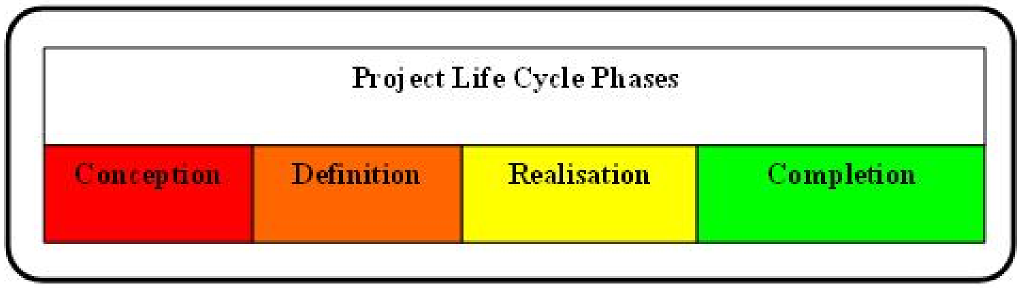


Figure 6.2 – Project lifecycle phases

The following paragraphs are, according to Dobie (2007), some of the essential components that are required to be included in a Project Management Plan (PMP):

- Project background.
- Project objective(s).
- Project scope.
- Project risks.
- Project exclusions, constraints, and assumptions.

Furthermore, the PMI Standards Committee (2000) suggests that project plans should also include a robust risk management component, detailed activity listing, and a corresponding schedule. While we have previously discussed the project background, scope and objectives, the following paragraphs cover the project risk, activities and schedule.

6.3 Project Scope

As discussed previously, the project scope is much more than the requirements or specification, and if not managed correctly can lead to an unsuccessful project. Broadly, project scope is managed under the following categories (PMI Standards Committee 2000):

- Scope planning,
- Scope definition,
- Scope verification, and
- Scope control.

While the essence of the first three components has effectively been detailed previously, scope control has not been addressed. Scope control looks at influencing and controlling the factors that may impact upon the agreed project scope. Appropriate scope control is therefore critical in ensuring that only approved changes are adopted, and are done so in a controlled manner. Changes which are outside this control are often referred to as scope creep and present a major risk to the success of this project.

6.4 Project risk

According to Gardiner (2005), risk management consists of risk assessment and risk control, both of which are effectively governed under the higher level category of risk planning. Given that that the task at hand is in fact a project by definition, the following project management risk types may also be applicable (MGT8022 2009):

- Technical risk – the possibility that the original requirements may be not achievable within the given schedule.

- Schedule risk – the possibility that the schedule will not conform to the management plan.

- Operational risk – the possibility that the equipment will not meet all of the user requirements.

From the above risk categories, technical risk and operational risk are the main risk types that may impact upon this or like projects. It is also important to highlight that while ENG4111/4112 (2009) details that risk implies a threat to personal or public safety, as can be seen from the project risk factors considered above, a holistic perspective to project risk considers much more than purely physical safety issues.

Lanza (2002) suggests that risk management essentially follows a process of: risk identification, risk assessment, response to risk, and documentation, with a resolution process structured around prevention, mitigation, or acceptance. Accordingly, Table 6.1 below is a brief risk management plan for this project and considers the relevant risks for this project.

Issue	Effect	Likelihood	Mitigation
Project scope creep cause by desire to increase application functionality or user interface	Project may differ from original requirements or specification	Possible	• Maintain strict adherence to specification. • Consider the need to balance changes against overall intent of project. • If changes are made to one area then consider how this impacts other areas given the fixed time constraint.
Project related detrimental effects on home network (Data loss)	Corruption or loss of data due to project related task	Possible	• Backup data regularly • Ensure development environment remains stable
Project related detrimental effects on home network (hardware damage)	Damage to computer hardware	Unlikely	• Use approved methods for hardware removal
Project related detrimental effects on small office network (Data loss)	Corruption or loss of data due to project related task	Possible	• Backup data regularly • Test in non peak times • Use guidance of system technical staff
Project related detrimental effects on home network (hardware damage)	Damage to computer hardware	Unlikely	• Use technical staff to change hardware if required.

Table 6.1 – Project risk management plan

6.5 Project activities and schedule

In looking at the project activities and schedule, one of the most obvious differences when compared with industry project management and scheduling, is the fact that the project specification has largely been defined to match the available timeframe. This approach differs with the common approach of using a project specification or requirements to drive a Work Breakdown Structure (WBS), and hence project schedule. This is not a problem, but more an observation of how time constraints can effectively curtail a project from inception.

According to the PMI (2006), planning is pivotal in successful project management, whereby the Work Breakdown Structure (WBS) is used to provide a foundation for defining the project work, and also the framework for managing the work. The planned schedule for the entire project is included in Appendix B, and is derived from the activity table which forms the basis of a WBS, in table 6.2 below.

Various Level Activity Listing		
1 Project	1.1 Project Phase 1 (ENG4111)	1.1.1 Topic Investigation/Negotiation
		1.1.2 Project Specification Negotiation
		1.1.3 Implementation Investigation
		1.1.4 Environmental Factors Research
		1.1.5 Requirements Preliminary Research
		1.1.6 Project Coding (Local app)
		1.1.7 Project Appreciation
	1.2 Project Phase 2 (ENG4112)	1.2.1 Write Draft Dissertation
		1.2.2 Project Coding (Networked app)
		1.2.3 Small office testing
		1.2.4 Partial Draft Dissertation
		1.2.5 Write Final Dissertation

Table 6.2 – Project Activity Table

6.6 Software Project Management

While the above paragraphs detail the level of generic project planning that has been applied to this project, it is important to note that software development, which is often perceived as intangible, differs from some project management approaches. Phillips (1998) is brutally honest with his depiction of 'typical software project management' illustrated in Figure 6.3 below. Additionally, Phillips (1998) takes a differing and perhaps more pragmatic approach to the project triple constraint proposed above, when he suggests that software project management could be done much better through a focus on the following areas:

- People perspective,
- Business perspective, and
- Process perspective.

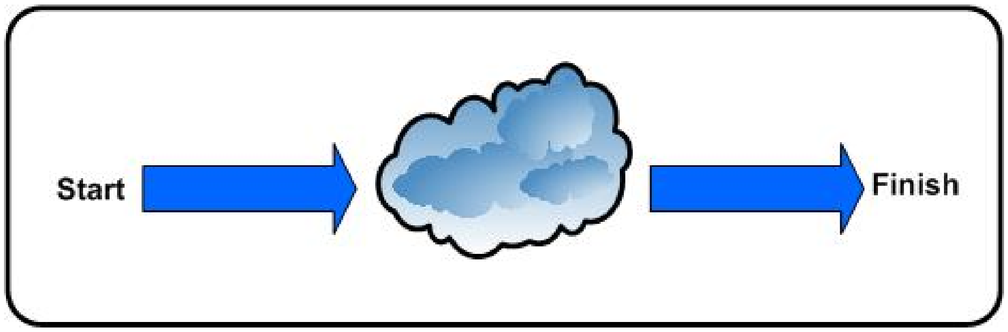


Figure 6.3 – Typical software project management
(Source: Phillips, 1998)

In defining how to maintain a focus on the above areas, Phillips (1998) has some very sound advice in building only the software that people want, and the business needs, in addition to using processes that are proven to work. In contrasting this approach with my project, it is worth highlighting that this application is very much an application that I wanted to build, hence it's functionality may or may not have a useful purpose in the 'outside world'.

Chapter 7. Software development

7.1 Introduction

The following paragraphs detail the software development environment, development methodology, design methodology and security considerations that combined to provide the overall development framework for the application.

7.2 Development and implementation environment

According to Eeles and Cripps (2010) the software development environment will include the infrastructure to support the method and tools of the developer(s). In looking at the methods and tools for this project, the primary components are:

- Implementation language, and
- Software development environment.

The identification and use of suitable Integrated Development Environment (IDE) which is an all one tool for writing, editing, compiling and running a computer program is a critical consideration in software projects. According to Qin, Xing and Zheng (2008) an IDE presents the following advantages from the perspective of a developer:

- Reduced requirement for programmers to remember grammar and syntax.

- Integrated management of development file systems.

- Holds an aggregation of tools which can cover design through to testing.

- Allows for the inclusion of developer defined tools.

While the above advantages were considered during the early stages of the definition phase of the project lifecycle, the primary consideration for the development environment was programmer skills and experience. Given the developers experience with the C and C++ languages, the initial environment was going to be built around a C++ implementation together a HTML derived user interface. However, through trial and error, a number of limitations with this approach were discovered and a decision was made to consider an alternative development environment.

The chosen development environment for the project was the Microsoft Visual Studio development environment. Microsoft Visual Studio development environment is one of the latest steps in software development environments and is a large and comprehensive compilation of development tools that can be accessed through a windowed interface (Sebasta 2005).

Highly related to the IDE outlined above was the selection of a suitable programming language. While the Microsoft Visual Studio development environment offers support for a number of different languages, the C# programming language was selected. Advantages of both the C# language and the chosen development environment are the relative ease of incorporating such things as Graphical User Interfaces (GUIs), and network capable applications. Additionally, the use of Visual Studio offers a number of advantages in terms of its compatibility with the Windows operating environment, which is the predominate networked environment that the application will be tested on.

Finally, according to Troelson (2003), C# provides almost as much flexibility and power as C++, and offers the following features in comparison:

- No pointers required, as C# programs do not generally have a need for direct pointers.

- Automatic memory management, which precludes the need for ‘delete’ keyword.

- Formal constructs for enumerations, structures and class properties.

7.3 Development methodology

The trial and error approach to the identification and selection of an appropriate development environment and language is largely similar to the prototyping software development methodology. Given this prototyping development methodology has the advantage of allowing the early detection of errors (Vliet 2001), it is perhaps unsurprising that a similar approach for the actual software development was undertaken.

The intended development methodology for this project was the waterfall software methodology. The decision to use the waterfall methodology which cascades from one phase to the next (Somerville 2007) and is illustrated in Figure 7.1 below, was based upon the following rationale:

- The project was defined by a robust and detailed specification, and
- The project included the project specification in addition to a comprehensive project scoping analysis.

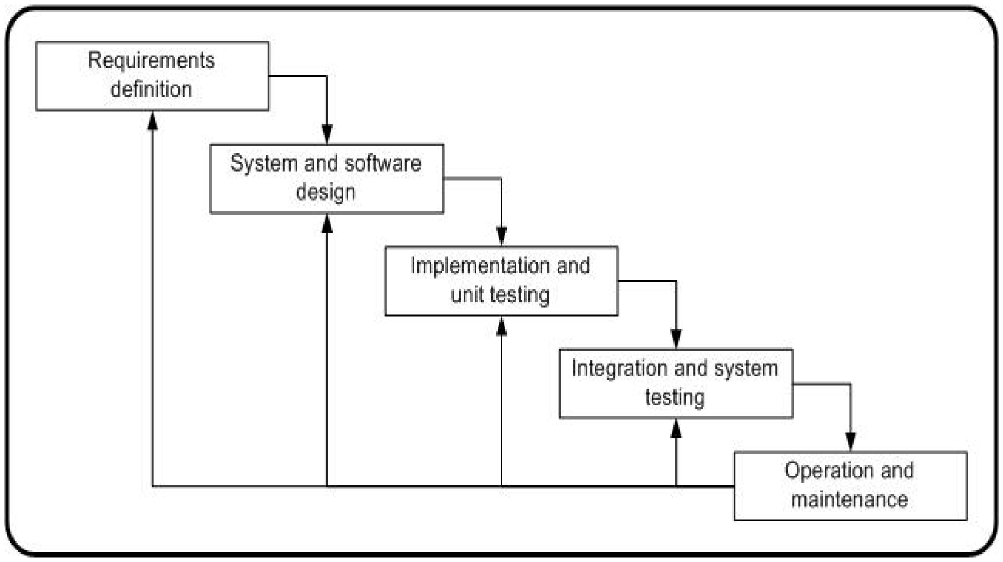

Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Figure 7.1 – ‘Waterfall’ software development lifecycle.

However, as can be seen in the above illustration the waterfall methodology requires a robust system and software design phase in order to produce a detailed system design. As a decision was made early in the development phase to move further into implementation at the expense of detailed design, the waterfall methodology in many ways became redundant.

The driver behind the decision to adopt a more agile development approach was based around the desire to complete the majority of the ‘client’ application before commencing on the ‘server’ application, which provided the network functionality of the application. The development of the application in this way is highly similar to the iterative/incremental software development life cycle, where progression towards the user requirements is achieved via delivery in a number of small steps (Schach 1990). The incremental development methodology is illustrated in Figure 7.2 below.

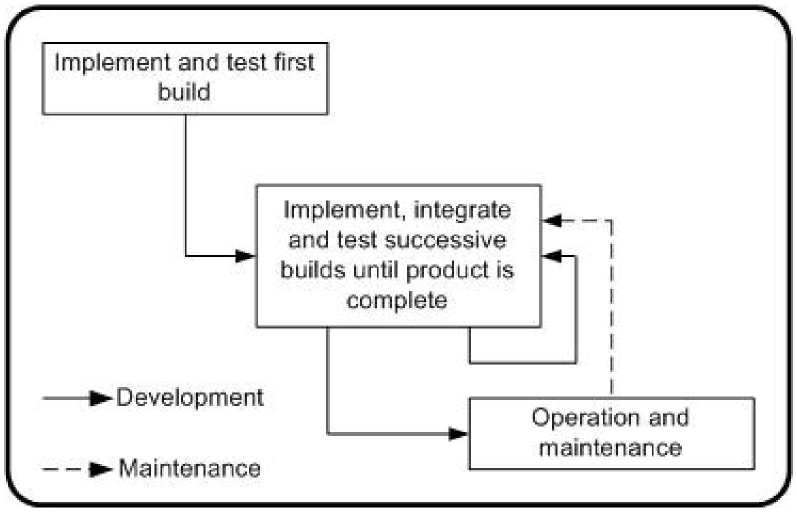

Implement and test first build
Implement, integrate and test successive builds until product is complete
Operation and maintenance
Development
Maintenance

Figure 7.2 – ‘Incremental’ software development lifecycle
(Source: Schach, 1990)

In the case of this project, the incremental development methodology offered the advantages of a rapid and early development, without the need for a rigid initial design. However, without a detailed design the risk of re-work was increased, that is, if either the development environment or implementation could not provide desired level of functionality, then significant re-work and possibly redesign, may have been required.

In looking at the advantages of the above approach it is worthwhile looking at the wider benefits of such an approach. From a practical perspective, Schwalbe (2006) suggests this type of adaptive approach seems to be more suitable for smaller scale Information Technology (IT) or software projects as it allows requirements to be managed within an iterative framework, when compared to a phased life cycle approach such as the waterfall methodology. Furthermore, an approach similar to above would present the following advantages:

- More flexible responding to changes associated with technology or user requirements.

- Achievement of milestones early in the project lifecycle and thus demonstrate earned value and other commercial benefits.

In combining the best of both of the above methodologies, a hybrid approach which is illustrated in Figure 7.3 was used as the development methodology for this project.

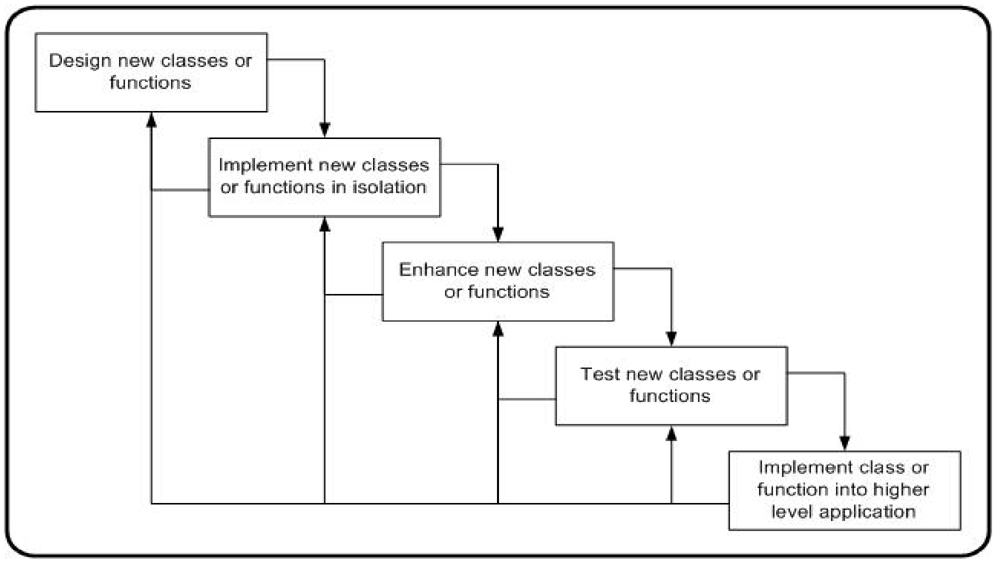

Design new classes or functions
Implement new classes or functions in isolation
Enhance new classes or functions
Test new classes or functions
Implement class or function into higher level application

Figure 7.3 – Hybrid software development lifecycle

7.4 Design methodology

7.4.1 Object-oriented approach

The chosen software design methodology for the project was an object-oriented approach. While the C# language implies an object-orientated approach, there still must be a desire by the programmer to maintain the principles of the object-oriented methodology irrespective of the chosen language.

Object-oriented programming presents a significant advantage in terms of dealing with one of the major problems associated with procedural programming. This problem in procedural programming occurs when changes made to parts of a program cause a cascading ripple effect throughout other parts of a program (Johnsonbaugh and Kalin 2000). Given that an incremental software development methodology was utilised for this project and therefore that major changes in software are likely to consistently occur throughout the software development process, the elimination of this issue through the adoption of an object-oriented approach is a significant advantage.

The chosen language, C#, supports all the concepts expected within a modern object-oriented language such as inheritance, encapsulation, polymorphism and interface-based programming (Dreyton, Albahari and Neward 2002).

7.4.2 Human-computer interaction

In looking at the user interface or human design requirements of the project, Dix, Finally, Abowd and Beale (1998) provide some valuable guidance in suggesting that software and systems should be designed in such a way that people with specific tasks will want to use systems in a way that is seamless with respect to their everyday role. Undoubtedly, one of the major challenges in this project was around the need to include significant user driven functionality, without creating an environment that was too complex or daunting for either new or experienced users.

Related to above, Mandel (1997) in suggesting that there will never be one perfect tool, and that visually sophisticated programs can get in the way, and hence distract users, further highlights the difficulty in creating a user interface that is suitable for a broad audience. Figure 7.4 below provides an illustration of the major categories that should be considered when designing a user interface.

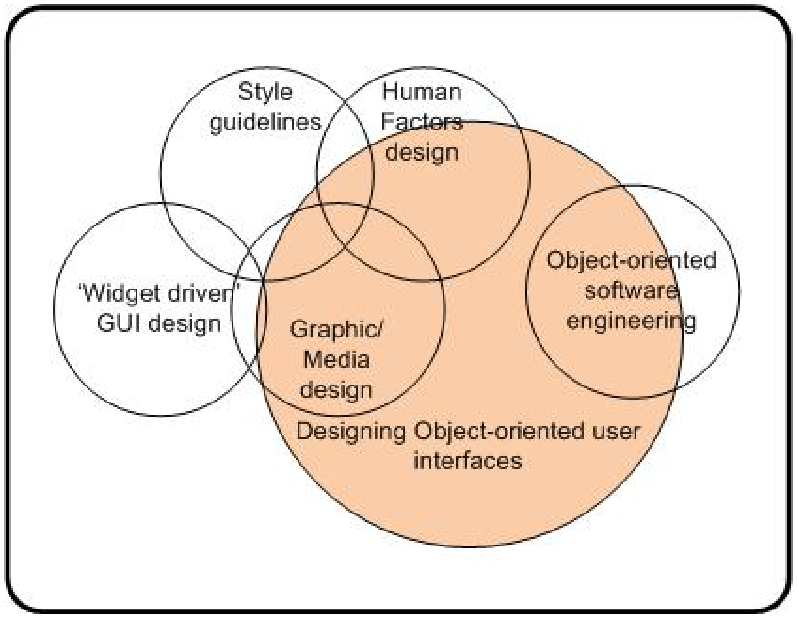

Style guidelines
Human Factors design
'Widget driven' GUI design
Graphic/ Media design
Object-oriented software engineering
Designing Object-oriented user interfaces

Figure 7.4 – User interface design approaches
(Source: Mandel, 1997)

While a number of the above attributes were considered, the primary considerations for the application developed were as follows:

- The available timeframe,
- The greater importance of functionality over user interface requirements,
- The assumption that the user of the ‘client’ application would be relatively inexperienced.
- The assumption that the primary application administrator or network controller would be moderately experienced in a technical sense.

7.5 Security

While the primary aim of this project is meet academic requirements, the consideration given to real world issues such as security and protection is still worthwhile. Additionally, when considering that variants of this application could potentially be deployed over Wide Area Networks (WANs), the need to consider security requirements early in the design phase is essential.

From a business and/or commercial perspective, Vachon and Graziani (2008) suggest that within any system there are three categories of security related vulnerabilities that will exist, namely: technological, configuration, and security policy weaknesses. In developing and deploying an application of this sort, the technological and configuration are the primary security considerations. Vachon and Graziani (2008) also suggest that the following items should be considered when looking at network security across an enterprise:

- Device hardening,
- Antivirus software,
- Personal firewalls,
- Software patches,
- Public/Private key encryption,
- Authentication,
- Virtual Private Networks (VPN).

Critical to the above is the implicit assumption is that either developed or off the shelf applications that are rolled out across an enterprise will effectively be self sufficient withy respect to security. That is, these applications will neither impact the security of the network, nor the data carried or accessed from within it. Furthermore, when considering that security is the property of a system to protect resources against accidental or malicious unauthorised use (Bruege & Dutoit 2004), then the need to consider security in any software design is further highlighted.

Security as an additional requirement for the developed application was not considered within this project. However, when considering Pressman's (2010) view that software security relates directly to software quality, then it can be assumed that utilising software that has been developed within a robust and appropriate methodology, will provide a degree of security. Furthermore, Pressman (2010) suggests that in order to build a secure system, a focus must be on quality, and therefore that a focus on eliminating architectural flaws rather than simply eliminating bugs must be pursued in development.

Building on the above, the security of the designed system is built on the following assumptions and considerations:

- The network on which the application resides will currently meet enterprise security requirements.

- The application has been developed within a robust development methodology.

- The incremental development methodology allowed for the early detection and correction of flaws, rather than a ‘debug’ at the end of the development.

- Significant effort has gone into ensuring that user menus are intuitive, and can therefore only provide an approved range of choices to a user.

- Where a PC enters hibernation, the OS will require user authentication a ‘wake’ time.

- Where a PC restarts, the OS will require user authentication at ‘boot’ time.

7.6 Implementation considerations and resources

As mentioned previously, the chosen development environment for is Microsoft’s Visual Studio. Although, typically an executable file developed from within this environment is fully portable and thus able to operate on any Windows based System, in the case of this project there are a number of other considerations.

The .NET framework contains the Common Language Runtime (CLR) and Base Class Libraries (BCL) that provide the majority of functionality and is required to be deployed to PCs that will operate the developed application. Furthermore, this framework should be a version that is compatible with the development environment.

The following paragraphs detail the other considerations and resources required to deploy and run the developed application:

- Microsoft .NET Framework 3.5 – The development environment is Visual Studio 2008, which is aligned with the .NET 3.5 framework.

- Dynamic Link Library (DLL) file(s) – These are the specific library files developed for the shared classes in the developed application.

- AUTOEXEC.BAT file – This is the plain text batch file that contains the run instructions for the developed application.

- Executable file(s) – The actual client or server application file(s).

- Printed files – The binary or text files that provide the permanent storage for the application.

Chapter 8. Program outline

8.1 Introduction

The following sections within this chapter describe the key features, functions and operation of the PowerMan client and server applications. As outlined previously, the core function of the applications is to manage the transitions between the S0 to S5 states to reduce the power wastage of a PC. Figure 8.1 provides a high level illustration of the relationship between these states.

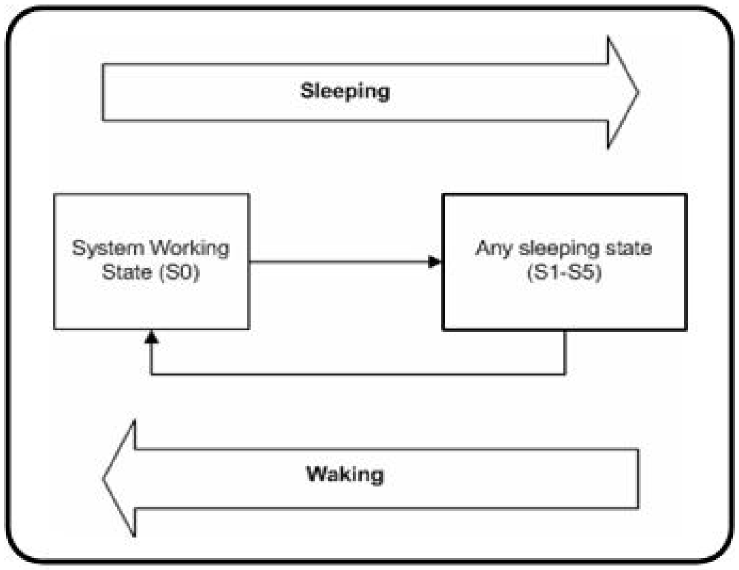


Figure 8.1 – Transitions between sleeping and waking states

A critical consideration in the design of this application was the selection of an appropriate sleeping state. As discussed above, the ACPI provides numerous ‘sleeping’ states, each with their own advantages and disadvantages in terms of responsiveness, power usage characteristics, and storage persistence. In selecting the appropriate standard ‘sleeping’ state for the application the following issues and features were given consideration:

- Responsiveness – The time taken to return to normal operation from a sleeping state.
- Power Usage – The relative power usage between the sleeping states.
- Data Assurance – The ability of the state to protect system data should an unexpected event like a power failure occur.
- Sleep Functionality – The ability to be able to utilise wake timers or like in the sleep state.
- Wake-on-LAN Compatibility – The ability to be able to respond to a WoL magic packet in the sleep state.

Table 8.1 details the results of a comparative assessment used to determine the appropriate sleeping state for the application. The results were drawn from testing on a number of ACPI capable x86 Intel PCs.

Feature	**Sleep State**		
	S3 (Standby)	**S4 (Hibernate)**	**S5 (Full off)**
Responsiveness	fast	medium	Slow
Power Usage	medium	low	Low
Data Assurance	no	full	full
Sleep Functionality	yes	yes	Some PCs
WoL Compatibility	Yes	yes	Some PCs

Table 8.1 – Performance of sleep state against desired criteria

Based on the results of the above assessment, the selected sleep state for the application was the ‘hibernate’ or S4 state which provided medium level responsiveness, low power usage, data assurance of ‘open’ data, the ability to be able to respond to wake timers, and finally the ability to be able to respond to a WoL command.

8.2 PowerMan general outline

The PowerMan applications are essentially two separate applications operating independently but utilising a client-server communications link to provide both a localised and remotely controlled PC power management capability. This primitive client server relationship is illustrated in Figure 8.2 below.

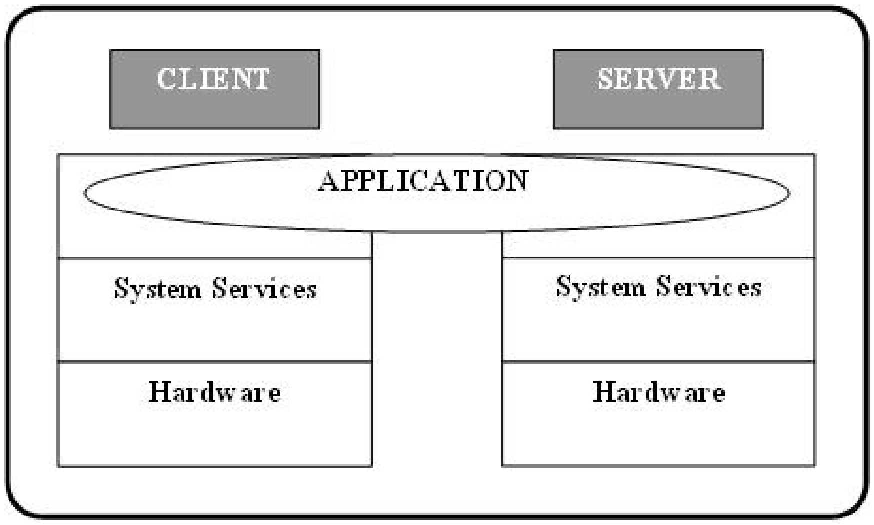


Figure 8.2 – Basic client server diagram

The design of the application in this way allows the local application to maintain control of the PCs scheduling and efficiency settings, that is, operate autonomously under the direction of the user. However, the local application also has with the ability to be able to provide information to the server application, or alternatively respond to server wake or sleep commands.

Figure 8.3 below provides a high level illustration of the overall application model. From this it can be seen that the server hosts client(s) in a one to many relationship, while both applications share a number of purpose built classes held within the class library.

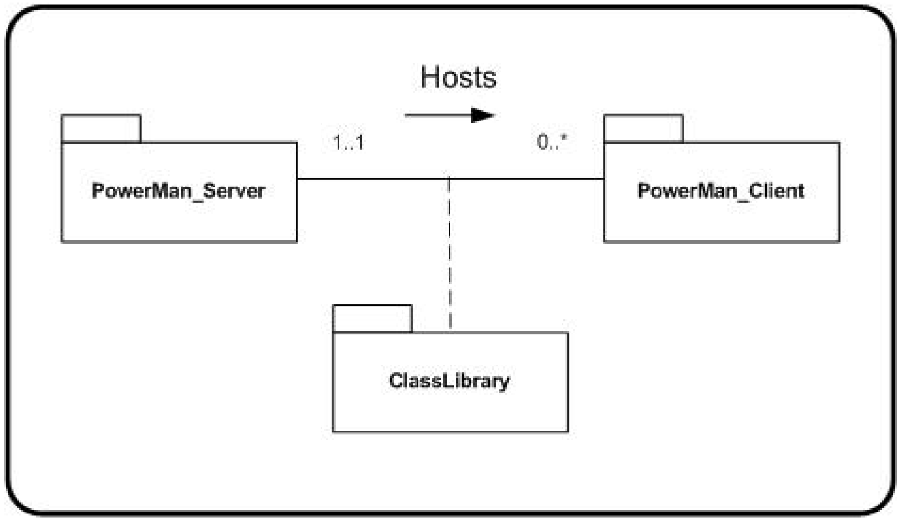


Figure 8.3 – High level application model

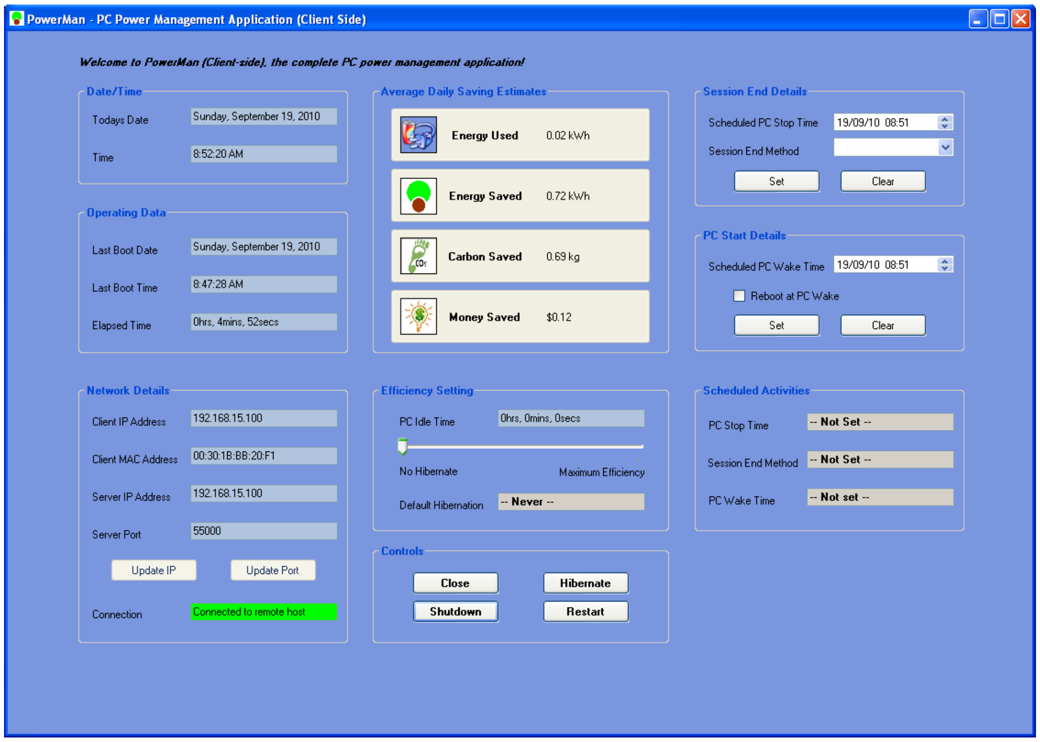
The general operation of the overall application sees the server application initialising first and from here ‘listening’ for clients. From here client applications are initialised and attempt to form a connection with the pre designated server IP address. Once running the client application will display a suite of system operating and efficiency data. The user is then able to schedule a future sleep event together with a follow up wake event, or alternatively adjust the efficiency setting that will result in the PC hibernating after the set amount of system idle time.

The server application will then display the identification of ‘connected’ clients, with the option of being able to view individual PC energy measures. The server can then shutdown or hibernate a client PC if desired, or close the application on an individual PC. Finally the server can set a wake schedule to mange a phased PC start-up at a regular time each day, and thus minimise traffic flooding and power spikes.

8.3 PowerMan client operation

As discussed above, the PowerMan client application can operate in conjunction with, or independently of, the PowerMan server application. A screenshot of the PowerMan client application is included in figure 8.4 below. The core functional areas of the PowerMan client application are as follows:

- System details.
- Server connection.
- Efficiency.
- Waking.
- Sleeping.
- Storage.



PowerMan - PC Power Management Application (Client Side)

Welcome to PowerMan (Client-side), the complete PC power management application!

Date/Time
Todays Date: Sunday, September 19, 2010
Time: 8:52:20 AM

Operating Data
Last Boot Date: Sunday, September 19, 2010
Last Boot Time: 8:47:28 AM
Elapsed Time: 0hrs, 4mins, 52secs

Network Details
Client IP Address: 192.168.15.100
Client MAC Address: 00:30:1B:BB:20:F1
Server IP Address: 192.168.15.100
Server Port: 55000
Connection: Connected to remote host

Average Daily Saving Estimates
Energy Used: 0.02 kWh
Energy Saved: 0.72 kWh
Carbon Saved: 0.69 kg
Money Saved: $0.12

Efficiency Setting
PC Idle Time: 0hrs, 0mins, 0secs
Default Hibernation: -- Never --

Session End Details
Scheduled PC Stop Time: 19/09/10 08:51
Session End Method:

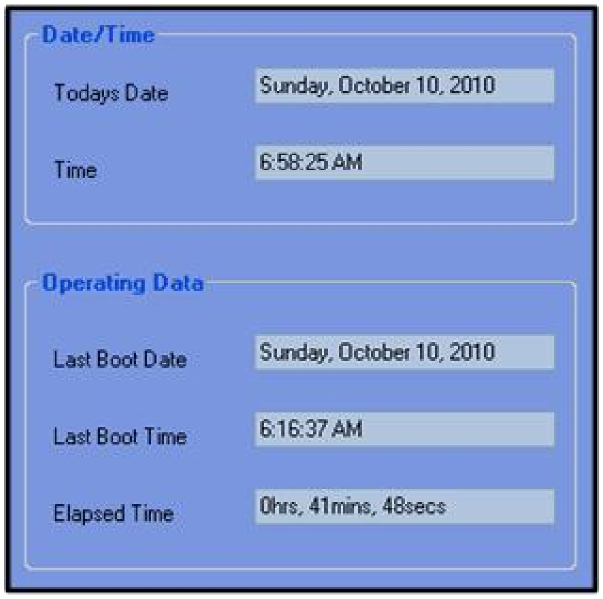
PC Start Details
Scheduled PC Wake Time: 19/09/10 08:51
Reboot at PC Wake: ☐

Scheduled Activities
PC Stop Time: -- Not Set --
Session End Method: -- Not Set --
PC Wake Time: -- Not set --

Figure 8.4 – 'PowerMan' client side screenshot

8.3.1 System details

The system details provided for viewing are the operating data which includes the last boot time and elapsed time since last boot, and also the current date and time. The last boot time data is gathered through the use of Windows API functions, while the current date and time make use of the system 'DateTime' class.


Date/Time

Todays Date: Sunday, October 10, 2010

Time: 6:58:25 AM

Operating Data

Last Boot Date: Sunday, October 10, 2010

Last Boot Time: 6:16:37 AM

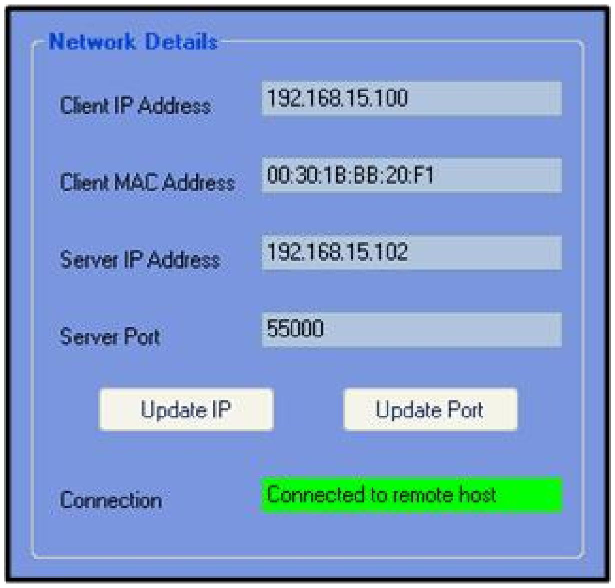
Elapsed Time: 0hrs, 41mins, 48secs

Figure 8.5 – Client system details

8.3.2 Server connection

The client IP and MAC address details are also gathered through the use of Windows API functions, while the server port and IP details are set as defaults within the implementation code. The server port and IP address can be modified within a small time window prior to the client attempting connection to the server.

The connection between the client and server is built around a Transmission Control Protocol (TCP) socket between the applications. TCP has the advantage of being able to guarantee delivery, and thus ensure that critical system messages are passed between the applications.


Network Details

Client IP Address 192.168.15.100

Client MAC Address 00:30:1B:BB:20:F1

Server IP Address 192.168.15.102

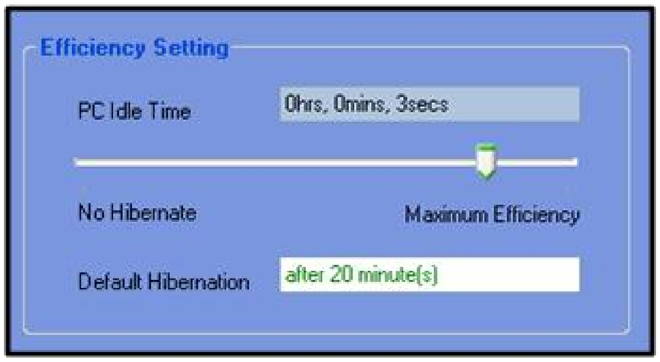
Server Port 55000

Connection Connected to remote host

Figure 8.6 – Client network details

8.3.3 Efficiency

The ‘efficiency’ of the client PC is set through the use of a selectable track-bar. This track-bar details a range of time settings which relate to the time before a sleep event will be initiated if the system idle time exceeds the set value. As in the case for a scheduled sleep event, if the idle initiates a sleep event and a wake time has been selected, the waitable timer will be invoked prior to sleeping.


Efficiency Setting

PC Idle Time 0hrs, 0mins, 3secs

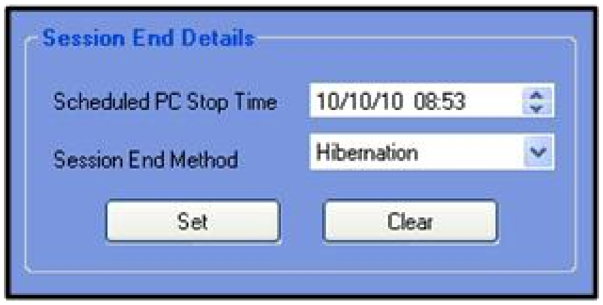
Default Hibernation after 20 minute(s)

Figure 8.7 – Client efficiency settings

8.3.4 Sleeping

A transition to a sleep state can be scheduled by a user for a set date and time. These transitions at the set times can either be to the hibernate or 'full off' states, or alternatively to schedule a complete system restart.

Importantly, a corresponding scheduled wake event can only occur if the PC is scheduled to sleep in the hibernate state. In this case, prior to sleeping, the internal waitable timer is initiated to ensure a wake event at the user selected time.

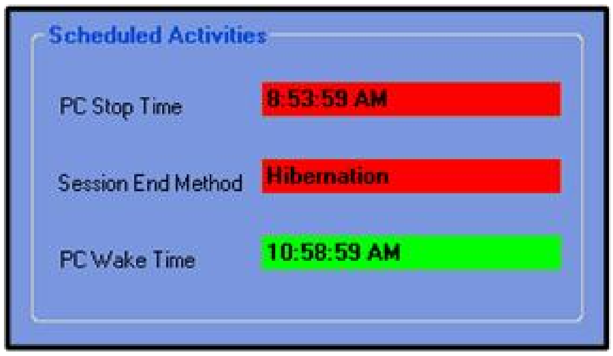


Session End Details

Scheduled PC Stop Time: 10/10/10 08:53

Session End Method: Hibernation

Figure 8.8 – Client session end settings



Scheduled Activities

PC Stop Time: 8:53:59 AM

Session End Method: Hibernation

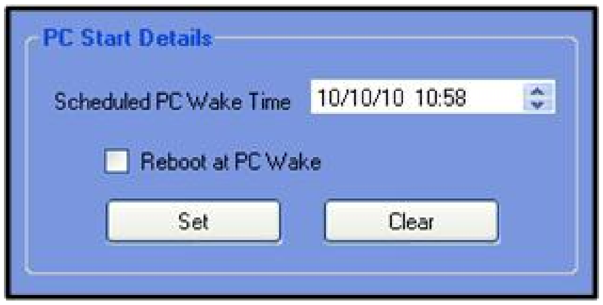
PC Wake Time: 10:58:59 AM

Figure 8.9 – Client session end and start details

8.3.5 Waking

As outlined above, scheduled waking occurs from the use of the internal waitable timer. The SetWaitableTimer function continues to operate in the low power S4 state, and when the timer is signalled, the system thread that initiated the timer calls the desired completion routine, that is, a system wake event.

The alternate method of waking, that is, using a server generated WoL command is detailed in the PowerMan server application outline.



Scheduled PC Wake Time 10/10/10 10:58

Reboot at PC Wake [ ]

Figure 8.10 – Client session wake settings

8.3.6 Storage

Storage for the client application is provided by the use of a binary data file. During initialisation of the client application or upon resumption from a sleep state, the client details file (if created) is read in, with the stored data being contained within an object of the purpose built Client class. The data stored is the client energy usage measures in addition to the server port and IP details.

Prior to closing the client application or transitioning the system to a sleep state, the Client object is written to the binary data file to provide persistent storage.

8.3.7 PowerMan client functions

The following paragraphs provide a brief description of the functions contained within the PowerMan client application.

- **SetHibernate** – Calls the system hibernate function for an enduring hibernation or alternatively calls the waitable timer function if a wake time has been set.

- **SetShutdown** – Initiates the formal shutdown process together with generating an appropriate system message.

- **SetRestart -** Initiates the formal shutdown (restart) process together with generating an appropriate system message.

- **CloseApplication** – Formal method to gracefully close the PowerMan application.

- **IsValidIP** – This method uses regular expression testing and returns a true value if the entered IP address is in the correct format, else returns false.

- **IsValidPort** – This method returns a true value if the entered port is in the correct range, else returns false.

- **FormatMAC** – This method takes MAC addresses with either the ':', '-', or ' ' separator and returns a standard MAC address format using the ':' separator, if a valid address.

- **CloseSocket** – This method closes open sockets and ensures socket references contain a null value.

- **SocketTest** – Tests client socket and Returns a true value if connected or false if not connected.

- **SaveToFile -** Specifies file to save client object to.

- **SetWaitForWakeUpTime** – Creates the waitable timer handle and suspends the computer before canceling the handle after waking. Will also force the PC into ‘display’ mode after waking.

- **ConnectToServer** – Creates the client socket, that is, connection to the server and gets the details of the connection from the socket reference.

- **WaitForData -** This method is called after the socket is established and waits for data to be received from server.

- **OnDataReceived** – Captures the data when it is read in from the socket to the server. Features a switch statement to take appropriate action based upon data received from server.

- **SendClientDetails** – Sends the client details object to the server.

- **ByteArrayToObject** – This method converts the client byte array to an object after it has been received over the socket.

- **UpdateControls** – This method provides a statement (and colour) indicating the current status of the client socket.

- **ObjectToFile** – This method converts the client object to a byte array so that it can be written to a binary file.

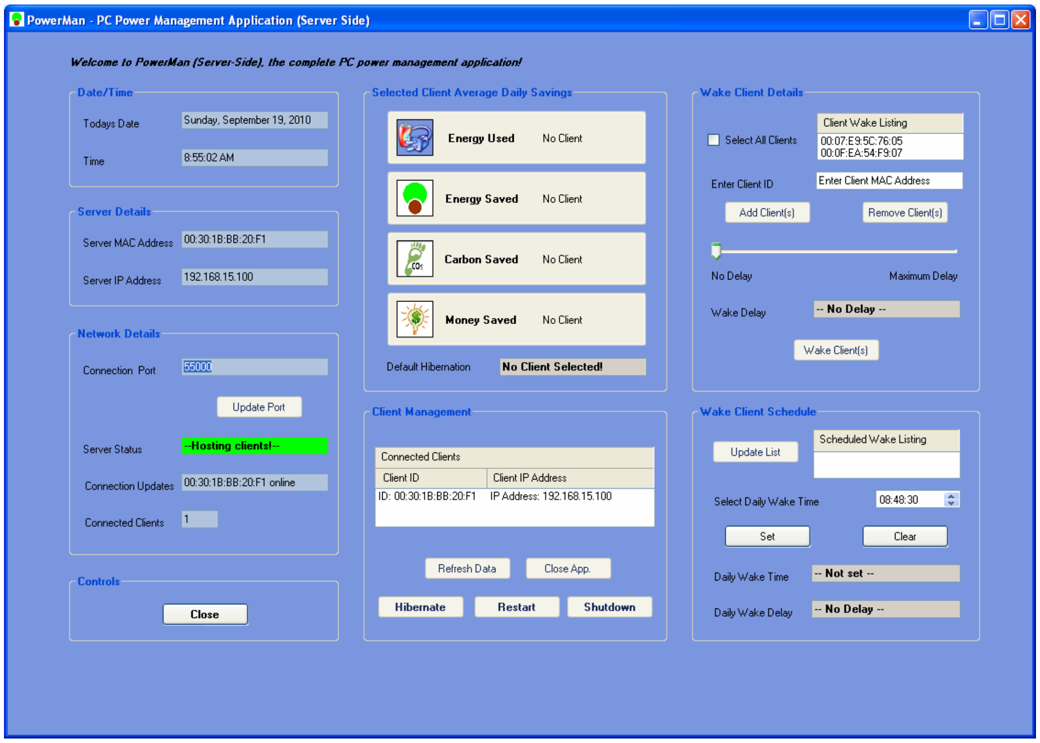
- **FileToObject -** This method converts a read in binary file to a client object.

- **ObjectToByteArray** - This method converts a client object to a byte array so that it can be transmitted over the socket.

8.4 PowerMan server operation

The PowerMan server application operates as a central hub for the PowerMan client application(s). A screenshot of the PowerMan server application is included in Figure 8.5 below. The core functional areas of the PowerMan server application are as follows:

- System details.
- Network details.
- Client management.
- Wake list management.
- Wake scheduling.
- Storage.


PowerMan - PC Power Management Application (Server Side)

Welcome to PowerMan (Server-Side), the complete PC power management application!

Date/Time

Todays Date: Sunday, September 19, 2010
Time: 8:55:02 AM

Server Details

Server MAC Address: 00:30:1B:BB:20:F1
Server IP Address: 192.168.15.100

Network Details

Connection Port: 55000
Server Status: --Hosting clients!--
Connection Updates: 00:30:1B:BB:20:F1 online
Connected Clients: 1

Selected Client Average Daily Savings

Energy Used: No Client
Energy Saved: No Client
Carbon Saved: No Client
Money Saved: No Client
Default Hibernation: No Client Selected!

Client Management

Client ID	Client IP Address
ID: 00:30:1B:BB:20:F1	IP Address: 192.168.15.100

Wake Client Details

Client Wake Listing:
00:07:E9:5C:76:05
00:0F:EA:54:F9:07
Enter Client ID: Enter Client MAC Address
Wake Delay: -- No Delay --

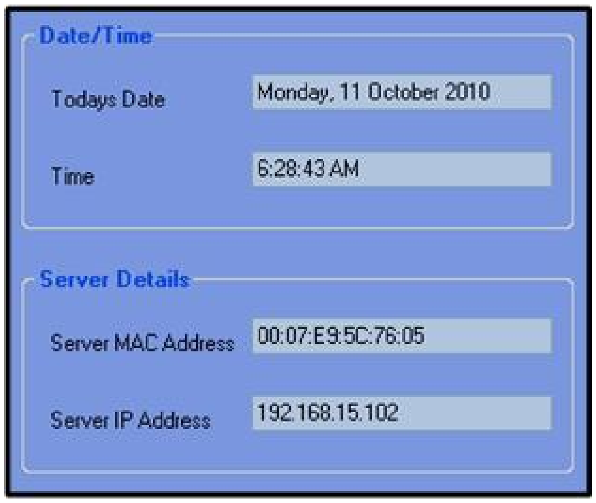
Wake Client Schedule

Scheduled Wake Listing:
Select Daily Wake Time: 08:48:30
Daily Wake Time: -- Not set --
Daily Wake Delay: -- No Delay --

Figure 8.11 – ‘PowerMan’ server side screenshot

8.4.1 System details

As with the client application, the server system details provided for viewing are the operating data which are the current date and time, and also the server IP and MAC address details. The current date and time make use of the system 'DateTime' class, while the server IP and MAC address details are accessed through the use of Windows API functions.



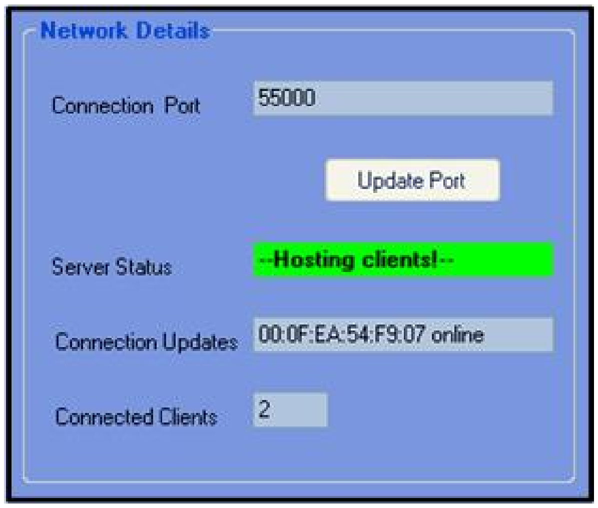
Todays Date: Monday, 11 October 2010
Time: 6:28:43 AM
Server MAC Address: 00:07:E9:5C:76:05
Server IP Address: 192.168.15.102

Figure 8.12 – Server system details

8.4.2 Network details

The server port details are set as defaults within the implementation code, which can be modified within a small time window prior to the server beginning to establishing the TCP listener. When the server commences 'listening', a status update indicating that the server is ready to host is displayed on the application.

Up to the second details of client connection status changes are provided through a poll of the listed open sockets. The details displayed are the most recent connection change, which includes the client details, and also the current number of connected clients.



Connection Port: 55000
Server Status: --Hosting clients!--
Connection Updates: 00:0F:EA:54:F9:07 online
Connected Clients: 2

Figure 8.13 – Server network details

8.4.3 Client management

The client management section lists the MAC and IP address details for all currently connected clients. As with the network details section, updates to this list are made every second through a status poll on the currently connected sockets. Importantly, this area also provides the 'controls' for which the server can direct the client application. These controls are only enabled when a client is selected within the list and provide the following functionality:

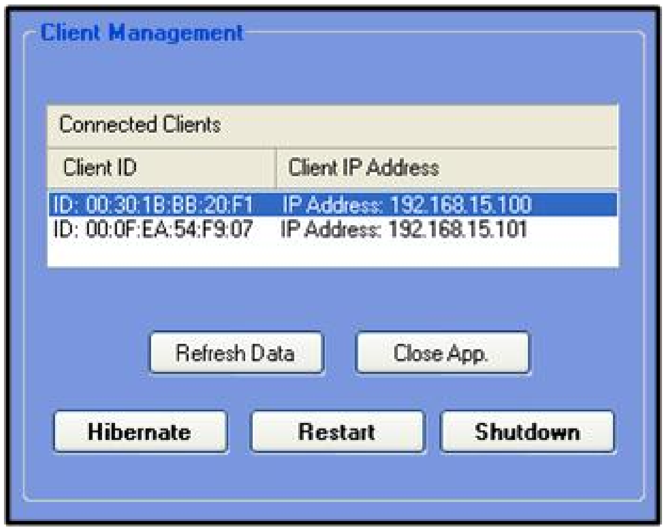
- **Refresh data** – This function allows the client energy data to be displayed which includes: Energy used, Energy saved, Carbon saved, and Money saved.

- **Close App.** – This function sends a string object containing a direction to close to the client application through the socket. The client then closes the application.

- **Hibernate** – This function sends a string object containing a direction to hibernate through the socket. The client application then immediately forces the client PC to enter hibernation.

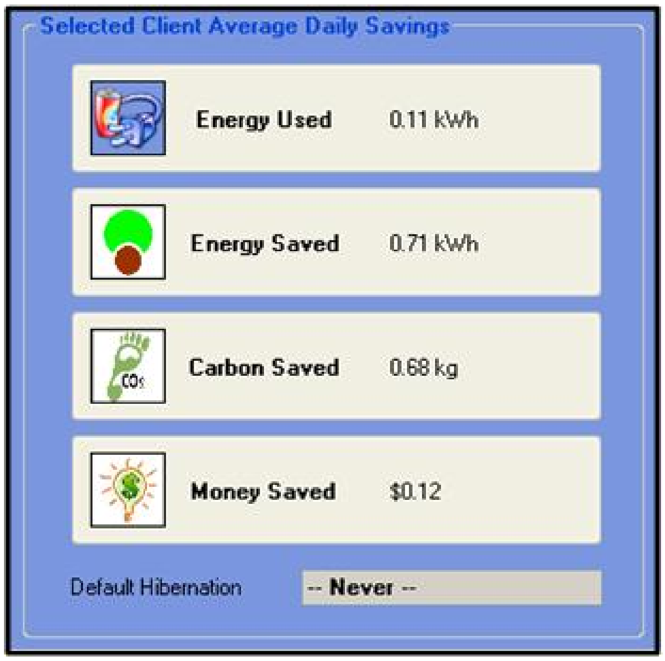
- **Restart** – This function sends a string object containing a direction to restart through the socket. The client application then immediately initiates a restart system warning message and time delayed restart on the client PC.

- **Shutdown** – This function sends a string object containing a direction to shutdown through the socket. The client application then immediately initiates a shutdown system warning message and time delayed restart on the client PC.



Client ID	Client IP Address
ID: 00:30:1B:BB:20:F1	IP Address: 192.168.15.100
ID: 00:0F:EA:54:F9:07	IP Address: 192.168.15.101

Figure 8.14 – Server client management details



Selected Client Average Daily Savings

Energy Used	0.11 kWh
Energy Saved	0.71 kWh
Carbon Saved	0.68 kg
Money Saved	$0.12

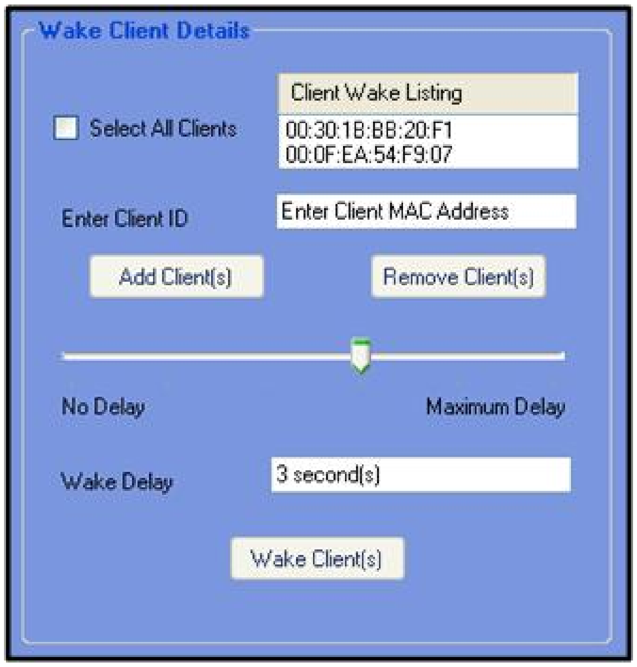
Default Hibernation: -- Never --

Figure 8.15 – Selected client energy details

8.4.4 Wake list management

The wake list management area is an updateable list of built on the clients that connect to the server. The listing is initially read in from a file during commencement of the server application, and then written to file when closing the application.

In addition to the automated updating of clients, the user can manually enter a client PCs MAC address for storing on the wake list. After selecting one or more client MAC addresses, the user can then activate the ‘wake’ button which will send a magic packet to the selected PCs, and hence wake them. The delay between wake magic packet transmissions to individual client PCs can be varied between zero and five seconds, and thus force a phased wake of PCs across a network.



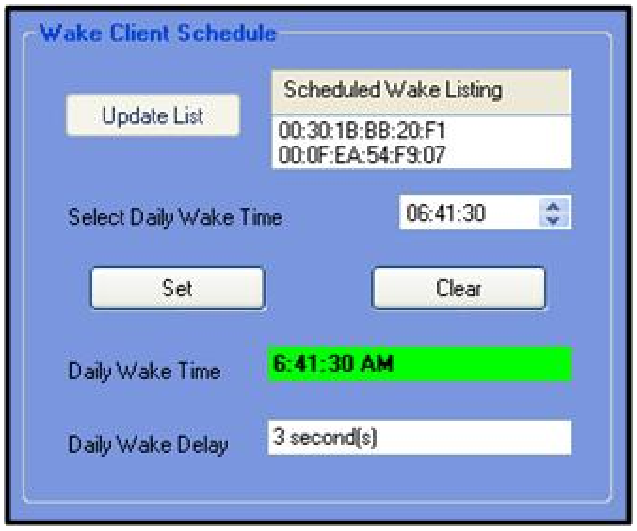
00:30:1B:BB:20:F1
00:0F:EA:54:F9:07

3 second(s)

Figure 8.16 – Server wake list details

8.4.5 Wake scheduling

The wake scheduling area allows for the setting of a periodical networked PC wake event. The scheduled wake list is updateable from the client wake listing and will indicate the client PCs that will be selected for the scheduled wake event. The scheduled wake event is then set for a daily occurrence, and as with the manual wake event, can operate with a delay ranging between zero and five seconds.



Scheduled Wake Listing
00:30:1B:BB:20:F1
00:0F:EA:54:F9:07

Select Daily Wake Time 06:41:30

Daily Wake Time 6:41:30 AM

Daily Wake Delay 3 second(s)

Figure 8.17 – Server scheduled wake list details

8.4.6 Storage

The storage for the server application is provided by the use of a text file. During initialisation of the server application, the text file (if created) is read in, with the stored data being contained within a string object. The data stored is the client MAC address details for use in wake management areas of the server application. Prior to closing the server application, the client wake list details are written to the text file.

8.4.7 PowerMan server functions

- **IsValidPort** – This method returns a true value if the entered port is in the correct range, else returns false.

- **IsValidMAC** - This method uses regular expression testing and returns a an upper case MAC address string if the entered MAC address is in the correct format, else returns " ".

- **IpToMacAddress** - This method uses an Address Resolution Protocol request to get a MAC address from a given IP address.

- **FormatMAC** – This method takes MAC addresses with either the ':', '-', or ' ' separator and returns a standard format using a ':' separator if a valid address.

- **ListenForClients** – This method creates a beginning primary socket and then after listening for clients begins to accept new client connections.

- **OnClientConnect** - This is the call back method which is invoked when each new client is connected.

- **SendServerRequest** - Sends the server request object to the server.

- **ObjectToByteArray** - This method converts a server object to a byte array so that it can be transmitted over the socket.

- **CloseSockets** - This method closes open sockets and ensures socket references contain a null value.

- **UpdateClientSockets** - This method tests all sockets and then updates the lists that containreferences the sockets to ensure that only client details for connected sockest are displayed.

- **SocketTest** – This method polls an individual socket and returns true if socket still open, else returns false.

- **WaitForData -** This method is called after socket is established and waits for data to be received from the client.

- **OnDataReceived -** Captures the data when it is read in from the socket to the client.

- **ByteArrayToObject -** This method converts the server byte array to an object after it has been received over the socket.

- **WakeUp** – Creates a list of clients to be woken using a magic packet. This wake event is flagged so that the wake is activated during the periodic timer operation.

- **WakeUpClient -** This method sends a Wake On Lan packet to the specified MAC address using UDP.

- **MACStringToBytes -** Converts a valid MAC address string to a byte array for use in the WakeUpClient function.

- **CountClients** – This method is called periodically to provide an up to date count of the actual number of connected clients.

- **WriteWakeList** – This method writes the wake list to a text file.

- **ReadWakeList** – This method reads in the wake list from the specified file.

- **UpdateClientControls** – This method will activate certain controls after a pre designated user prompt or input.

Chapter 9. Testing and evaluation

9.1 Introduction

Software development presents some unique challenges from an engineering perspective. Firstly, it is extremely difficult to test given the large number of variables that are associated with its operation. Secondly, the accurate measurement of software reliability, when compared to hardware is difficult to measure (Faulconbridge & Ryan 2003)

9.2 Testing methodology

In line with the previously described development methodology, the testing has been incorporated within the development phases using a form of development methodology which according to Somerville (2004) relies on an incremental approach to software development. This has effectively allowed the trial of features etc. as they are implemented and added rather than at the end of development. Figure 9.1 below provides a simple illustration showing the chosen testing approach.

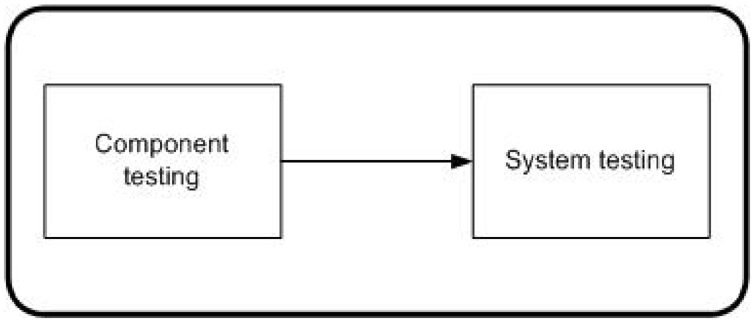


Figure 9.1 – Testing phases

A more detailed view of the process behind the testing at each of the above phases of the software development lifecycle is illustrated in Figure 9.2 below.

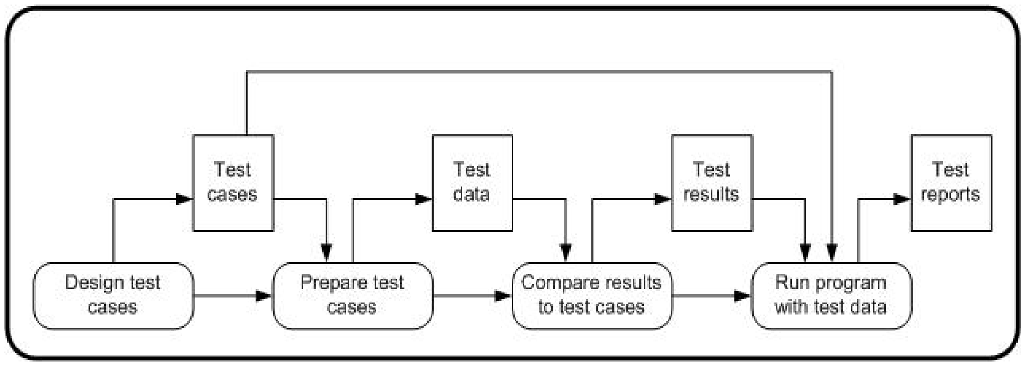


Figure 9.2 – Example of software testing process model
(Source: Somerville, 2004)

Although the test model used for this project was not as rigid or methodical as the process detailed above, a conscious effort was made to ensure that testing aligned with the specification and objectives of the project, rather than simply using testing as an advanced form of debugging.

9.3 Test environment

Given the specifications, the actual application, and finally the testing regime have been developed and implemented by the same person, a typical testing regime which separates functional, user, and possibly compliance testing has not been selected.

As explained previously, the majority of the application testing has occurred during the incremental development of both the client and the server application(s). However, as there is a need to understand the performance of the applications in relation to the original requirements, detailed functional testing has taken place.

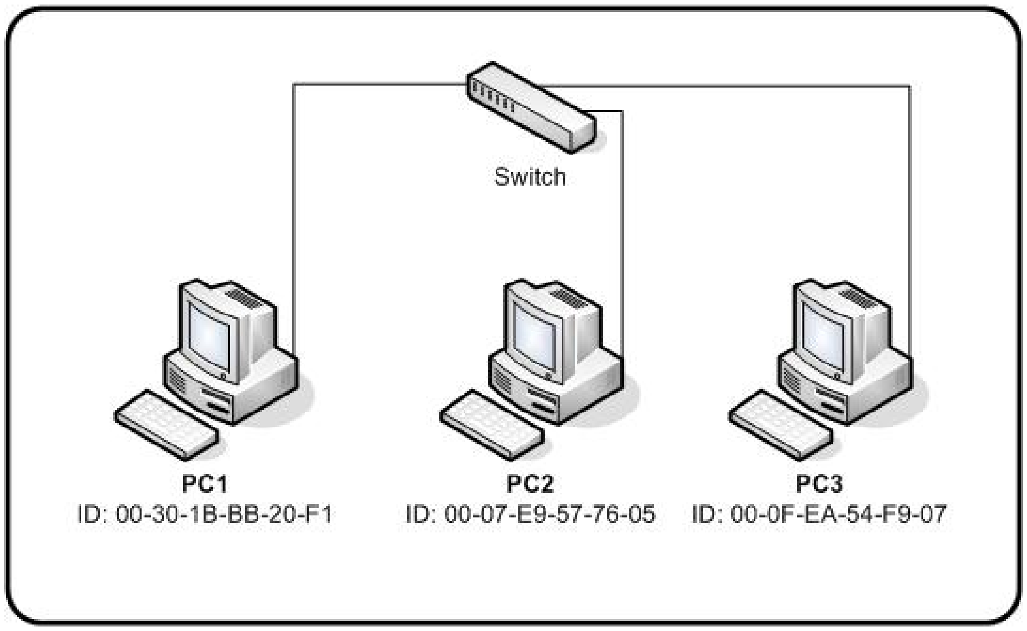


Figure 9.3 – Application development and testing network environment

The development and testing environment for the PowerMan application(s) is illustrated in Figure 9.3 above. While this environment does not contain an overly large number of networked PCs, it does provide an adequate network for the purpose of establishing multiple client communication with a server, and thus evaluating the performance of the application(s).

9.4 Test plan and results

The aim of this test plan and subsequent test cases is to detail the functional performance of the applications.

9.4.1 Client specific testing

a) Successful start-up (without storage file)

Comments: Client application commences with no initial file.

b) Successful start-up (with storage file)

Comments: Client application commences reading in storage file.

c) Loaded data integrity

Comments**:** Program displays the same data as the last time the application was run.

d) System data display

Comments: Program display correct date and time.

e) Operating data display

Comments: Program display correct last boot time and provides correct time since last boot event.

f) Client network details display

Comments: Program displays correct client MAC and IP address

g) Server connection details display

Comments: Program displays server IP address and port details and can be modified in the initial open time only.

h) Connection status display

Comments: Connection status is displayed and counts down until connection initiation, and then displays appropriate connection status.

i) Energy data display

Comments: Current energy data is correctly displayed for client.

j) Efficiency setting idle time display

Comments: Idle time is displayed and is correctly reset by user interaction.

k) Efficiency setting operation

Comment: User efficiency track-bar operates and correct value is displayed in text box. Client PC hibernates according to selected efficiency time.

l) Client control operation

Comments: All client controls (close, hibernate, shutdown and restart) gracefully close the application and provide the desired PC/system response.

m) Session end operation

Comments: Valid session end details can be entered, and are displayed. Application ends at desired time and by desired method, and hence forces corresponding action upon PC.

n) Session start operation

Comments: Valid session start details can be entered, and are displayed. PC and application commences at desired time if last end method was by hibernation.

9.4.2 Server specific testing

a) Successful start-up (without storage file)

Comments: Server application commences with no initial file.

b) Successful start-up (with storage file)

Comments: Server application commences reading in storage file.

c) Loaded data integrity

Comments**:** Program displays the same data as the last time the application was run.

d) System data display

Comments: Program display correct date and time.

e) Client network details display

Comments: Program displays correct server MAC and IP address

f) Network connection details display

Comments: Program displays correct port details and can be modified in the initial open time only.

g) Network connection status display

Comments: Server hosting status is displayed and counts down until connection initiation, and then displays appropriate listening status. Connection updates and number of connected clients are displayed and automatically updated.

h) Server control operation

Comments: Server controls (close) gracefully close the application.

i) Energy data display

Comments: Current energy data is correctly displayed for selected client.

j) Efficiency setting display

Comments: Efficiency setting for selected client is displayed.

k) Client management display

Comment: Connected client details are correctly displayed.

l) Wake list operation (Manual)

Comments: Correct wake list (from file) is displayed. Only valid MAC addresses can be written to the list. Wake delay track-bar operates with the chosen wake delay value stored in the text box.

m) Wake list operation (Automated)

Comments: Scheduled wake list can be updated from 'client wake listing'. Daily wake time can be modified and displays correctly.

n) Session start operation

Comments: Valid session start details can be entered, and are displayed. PC and application commences at desired time if last end method was by hibernation.

9.4.3 Joint server and client testing

a) New connection status display

Comments: Client application displays correct status when client connects to server. Server application displays correct status when client connects to server.

b) Closed connection status display

Comments: Client application displays correct status when client is disconnected from server. Server application updates connection information when client application closes or hibernates.

c) Server initiates client 'Refresh Data.'

Comments: Server forces client application refresh energy data.

d) Server initiates client 'Close App.'

Comments: Server forces client application to close gracefully.

e) Server initiates client 'Hibernate'.

Comments: Server forces client application to hibernate.

f) Server initiates client 'Restart'.

Comments: Server forces client application to restart.

g) Server initiates client 'Shutdown'.

Comments: Server forces client application to shutdown.

h) Server initiates manual wake event.

Comments: Selected client PCs are successfully woken with the selected time delay between PC wake events.

i) Server sets automated wake event.

Comments: Selected client PCs are successfully woken at the entered time with the selected time delay between PC wake events. Additionally, selected client PCs are successfully woken at the entered time over numerous days (post the initial wake event), with the selected time delay between PC wake events.

9.5 Testing summary

Overall, the testing satisfied all the requirements of the test plan, and hence indicates that the PowerMan applications meet and exceed the items detailed in the PowerMan specification.

Chapter 10. Conclusions and Future developments

While the core purpose of this project was to develop a solution that met the requirements of the specification, there are other significant other success factors which should be considered when assessing the benefits of the project. The overall successes of the project are:

- The development of the 'PowerMan' application(s) which meet the requirements of the specification.

- The adoption of a hybrid development lifecycle tailored to meet the requirements of this project.

- The ability to develop the project successfully while adhering to project management methodologies, and thus satisfy the scope, time and cost estimates for the project.

10.1 Achievement of project objectives

As stated above, the project research combined with the developed applications have met all of the project objectives. While neither the time nor the resources were available to allow for the testing of the applications in a larger office environment, the testing regime and testing environment provided a solid functional representation of an office LAN environment. Therefore, it is fair to suggest that this testing can be deemed to satisfy the project specification *desirable* item of 'large office testing'.

10.2 Further work

Ideally, the ‘PowerMan’ applications could be used in conjunction with more energy efficient PCs and peripherals and therefore provide an overall package that is power efficient whatever its state of operation, with little control or interaction from the user.

Other suggestions for further development to either provide a better tool in terms of function or power saving would be:

- Intuitive scheduling – Further development of the ‘PowerMan’ applications could provide greater program functionality such as functionality that can examine current and planned system processes and look at scheduling shutdown and wakeup operations around these to minimize power use without relying on user driven scheduling.

- Aggregation of client statistics – This would allow for the server application to present an aggregated network energy usage and savings figures.

- Business hours related energy statistics – This would allow energy used and saved to reflect the standard hours of operation for a business, and therefore account for business hours, public holidays, and weekends in energy savings calculations.

Overall, the opportunity to research, develop and test this application has provided a valuable and enjoyable learning experience. Furthermore, the ability to be able to work on a project where the environment may be a long term benefactor has also been extremely satisfying.

List of References

Addie, R 2007, *CSC3413 Network design and analysis: study book*, University of Southern Queensland, Toowoomba.

Bigelow, S 2003, *Bigelow's PC Hardware Desk Reference*, McGraw-Hill, New York.

Bruege, B & Dutoit, A 2004, *Object-Oriented Software Engineering Using UML, Patterns and Java*, 2nd edn, Pearson Education Inc., Upper Saddle River, NJ.

Carbon Dioxide 2010, *Renewable Energy and Carbon Dioxide*, web article, viewed 10 June 2010, <http://www.esru.strath.ac.uk/EandE/Web_sites/01-02/RE_info/C02.htm>

Cisco 2010, Comparing Traffic Policing and Traffic Shaping for Bandwidth Limiting, Web article, viewed 10 June 2010, <http://www.cisco.com/en/US/tech/tk543/tk545/technologies_tech_note09186a00800a3a25.shtml>

Department of Energy 2000, *Carbon Dioxide Emissions from the Generation of Electric Power in the United States*, Web Report, viewed 10 June 2010, http://www.eia.doe.gov/electricity/page/co2_report/co2report.html#electric

Dix, A, Finlay, J, Abowd, G & Beale, R 1998, *Human Computer Interaction*, 2nd edn, Prentice Hall, Harlow, Essex.

Dobie, C 2007, *A Handbook of Project Management: A Complete Guide For Beginners to Professionals*, Allen & Unwin, Crows Nest, NSW.

Dreyton, P, Albahari, B & Neward, T 2002, *C# in a Nutshell*, 2nd edn, O'Reilly and Associates, Sebastopol, CA.

Eeles, P & Cripps, P 2010, *The Process of Software Architecture*, Pearson Education Inc., Boston, MA.

ENG4111/4112 NIA Research Project: study book 2010, University of Southern Queensland, Toowoomba.

Ergon Energy, 2010, *Electricity Prices*, Energy Cost Schedule, viewed 10 June 2010, <http://www.ergon.com.au/your-business/accounts--and--billing/electricity-prices>

Faulconbridge, R & Ryan, M 2003, *Managing Complex Technical Projects: A Systems Engineering Approach*, Artech House Inc., Norwood, MA.

Galorath, D and Evans, M 2006, *Software Sizing, Estimation, and Risk Management*, Aurbach Publications, Boca Raton, FL.

Gardiner, P 2005, *Project Management: A Strategic Planning Approach*, Palgrave Macmillan, Hampshire.

Hancock, B 1990, *Issues and Problems in Computer Networking*, Amacom, United States.

Held, G 1996, *Understanding Data Communications*, 5th edn, Sams Publishing, Indianapolis, IN.

Hewlett-Packard, Intel, Microsoft, Phoenix & Toshiba 2010, *Advanced Configuration and Power Interface Specification, Revision 4.0*, specification, 5 April 2010, viewed 10 June 2010, <http://www.acpi.info/spec.htm>

Intel 2010, *Network Connectivity*, web article, viewed 10 September 2010, <http://www.intel.com/support/network/sb/cs-008459.htm>

Interstate Assessment Technologies (IAT) 2010, *Transient Voltage*, Web article, viewed 10 June 2010, <http://iat-eztyme.com/transients.pdf>

Johnsonbaugh. R & Kalin, M 2000, *Object-Oriented Programming in C++*, 2nd edn, Prentice Hall Inc., Upper Saddle River, New Jersey.

Kawamoto, K., Shimoda, Y. and Mizuno, M. (2004), *Energy Saving Potential of Office Equipment Power Management*, Energy and Buildings, Volume 36, Issue 9, pp915-923.

Kerzner, H 2003, *Project Management: A Systems Approach to Planning, Scheduling and Controlling*, 8th edn, John Wiley & Sons Inc., Hoboken, New Jersey.

Kist, A 2009, *09-070 Development and Testing of a Networked PC Power Management Tool*, Project Topic Description, Faculty of Engineering and Surveying, University of Southern Queensland, viewed 16 May 2009, <http://www.usq.edu.au/engsurv/students/enrolment/project/b-topicoffer/topicoffer09/topics61-121/09070.htm>

Lanza, R 2002 'Does Your Project Risk Management System Do the Job?', Auerbach Publications, New York

Leonhard, W & Murray, K 2009, *Green Home Computing for Dummies*, Wiley Publishing, Inc., Indianapolis, Indiana.

Lewis, W 2005, *LAN Switching and Wireless: CCNA Exploration Companion Guide*, Cisco Press, Indianapolis, Indiana.

Mandel, T 1997, *The Elements of Unser Interface Design*, John Wiley & Sons Inc., United States.

MGT8022 NIA Project management framework: study book 2009, University of Southern Queensland, Toowoomba.

MSDN 2010, *Overview of the Windows API*, web article, viewed 10 June 2010, <http://msdn.microsoft.com/en-us/library/aa383723.aspx>

MSDN 2010, *System Power States*, web article, viewed 10 June 2010, <http://msdn.microsoft.com/en-us/library/aa373229%28VS.85%29.aspx>

MSDN 2010, *Windows Management Instrumentation*, web article, viewed 10 June 2010, http://msdn.microsoft.com/en-us/library/aa394582%28VS.85%29.aspx

My Clean Sky, 2010, *A ton of greenhouse emissions produced can be balanced by a ton of emissions reduced*, web article, viewed 10 June 2010, <http://www.mycleansky.com/?a=efficiency>

Neumann, L 2008, *Wake-On-LAN in C#,* Memos Team Blog, Blog entry, viewed 10 June 2010, <http://blog.memos.cz/index.php/team/2008/06/12/wake-on-lan-in-csharp>

Phillipson, G 2008, *The 2008 Digital Atlas of Australia,* Web article, Connected Home, viewed 16 May 2010, <http://www.connectedhome.com.au/article/digital-atlas-australia>

Phillips, D 1998, The Software Project Manager's Handbook – Principles That Work At Work, IEEE Computer Society Press, Los Alamitos, CA

Parliamentary Office of Science and Technology 2006, Postnote, web article, viewed 10 June 2010, http://www.parliament.uk/documents/post/postpn268.pdf

Pressman, R 2010, Software Engineering – A Practitioners Approach, McGraw-Hill, New York, NY.

Project Management Institute (PMI) Standards Committee 2000, *A guide to the project management book of knowledge*, 3rd edn, PMI, Newtown Square, Pennsylvania.

Project Management Institute (PMI) 2006, *Practice Standard for Work Breakdown Structures*, 2nd edn, PMI, Newtown Square, Pennsylvania.

Qin, Z, Xing, J & Zheng, X 2008, *Software Architecture*, Zhejiang University press, Hangzhou.

Rowsell-Jones, A 2007, 'Performance Artist', *CIO*, December 2007/January 2008, p. 8.

Scach, S 1990, *Software Engineering*, Asken Associates Inc., United States.

Schwalbe, K 2006, *Information Technology Project Management,* 4th edn, Thompson Course Technology, Boston.

Sebasta, R 2005, *Concepts of Programming Languages*, 7th edn, Pearson Education Inc., Boston, MA.

Silberschatz, A, Galvin, P & Gagne, G 2005, *Operating System Concepts*, 7th edn, John Wiley & Sons, Inc., Danvers, MA,

Silver, J 2008, *Global warming & climate change: Demystified*, McGraw-Hill, New York.

Slone, J 2000, Local Area Network Handbook, 6th edn, Aurbach Publications, Boca Raton, FL.

Smith, J 2010, *Computer Basics – 10 What You See: Power Protection,* Web Article, Jans Computer Basics, viewed 10 June 2010, <http://www.jegsworks.com/lessons/lesson10/lesson10-5.htm>

Somerville, I 2004, *Software Engineering*, 7th edn, Pearson Education Limited, Harlow, Essex.

Somerville, I 2007, *Software Engineering*, 8th edn, Pearson Education Limited, Harlow, Essex.

ThomasNet, 2010, Type of Electrical Power Supply Interference, Web Article, viewed 10 June 2010, <http://www.thomasnet.com/articles/electrical-power-generation/power-supply-interference>

Troelsen, A 2003, *C# and the .NET Platform*, Springer-Verlag New York Inc., New York, NY.

Vachon, B & Graziani, R 2008, *Accessing the WAN, CCNA Exploration Companion Guide*, Cisco Press, Indianapolis,

Vliet, H 2001, *Software Engineering – Principles and Practice*, 2nd edn, John Wiley & Sons Inc., New York, NY.

Williams, M. A. J. and Balling, R. C. (1996) *Interactions of Desertification and Climate*, WMO/UNEP, Arnold, London.

Webber, C, Roberson, J, Brown, R, Payne, C, Nordman, B & Koomey, J 2001, *Field Surveys of Office Equipment Operation Patterns*, Berkeley, CA: Lawrence Berkeley National Laboratory. Report No. LBNL- 46930, viewed 10 June 2010, <http://enduse.lbl.gov/Projects/OffEqpt.html>

Wikipedia 2010, *Coal*, Web article, viewed 10 June 2010, <http://en.wikipedia.org/wiki/Coal>

Wikipedia 2010, *Traffic Shaping,* Web article, viewed 10 June 2010, <http://en.wikipedia.org/wiki/Traffic_shaping>

Wysocki, R, Beck Jr., R & Crane, D 2000, *Effective Project Management*, 2nd edn, John Wiley & Sons Inc., New York

Appendix A - Research Project Specification

Student Name:	**S. Brooks** (Student Number: 0011120394)
Enrolment:	ENG4111
Supervisor:	Dr. Alexander Kist (FOES)
Project Aim:	The development and testing of a distributed PC power management software tool. This tool, through the use of efficient scheduling and the control of a PCs start-up and shutdown, is likely to reduce the amount of power wasted by an idle PC.
Revision:	Issue A – 22 March 2010

1. Research and evaluate the average power use, idle time and other statistics of a PC that relate to power consumption in a standard office environment.

2. Research the potential for, and severity of power spikes, network traffic floods and other issues in a standard office environment due to simultaneous PC start-up.

3. Develop a software tool that controls start-up and shutdown of a PC either via scheduling or manual control.

4. In conjunction with the local software tool described in component 3, develop a software tool that remotely controls the start-up and shutdown of either singular or grouped PCs (within a designated domain) over a Local Area Network (LAN).

5. In addition to specifications outlined in components 3 and 4 above, the software tool shall:

   c. Implement a phased start-up (to prevent power spikes) where multiple PCs on a single LAN may be scheduled to start-up.
   d. Presentation of software tool options, controls, and critical usage statistics within a user friendly Graphical User Interface (GUI).

6. Evaluate and test the tool in a small office/network environment

Desirable components (as time permits)

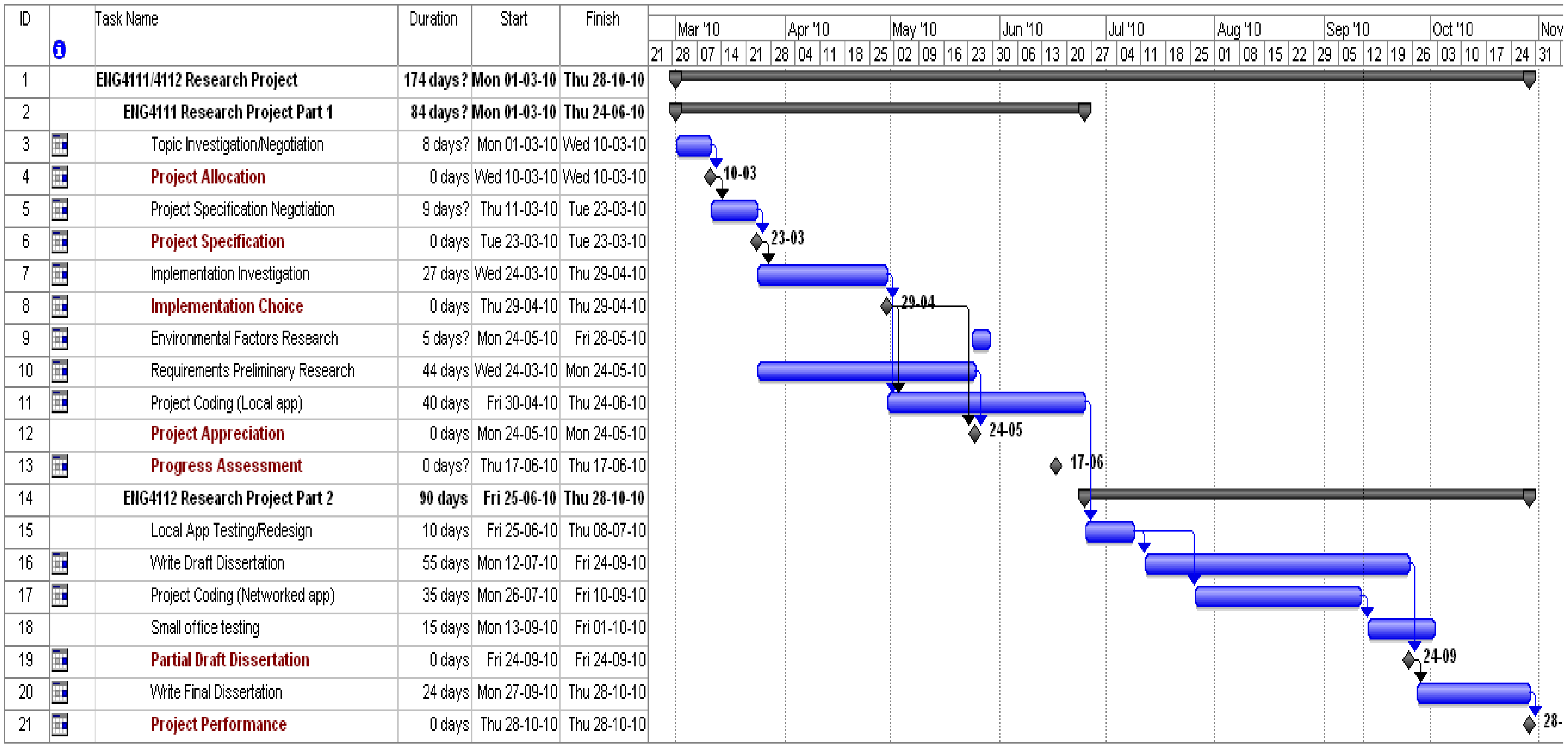
7. Evaluate and test the tool in a large office/network environment.

8. Translate 'wasted power measurements', and also saved power with the use of the tool into figures relating to a 'carbon footprint' and greenhouse gas measurement.

---***APPROVED***---

AGREED	**Student**	**Supervisor**	**Examiner/Co-examiner**
	Date: / /2010	Date: / /2010	Date: / /2010

Appendix B - Research Project Schedule



ID	ℹ	Task Name	Duration	Start	Finish
1		**ENG4111/4112 Research Project**	**174 days?**	**Mon 01-03-10**	**Thu 28-10-10**
2		**ENG4111 Research Project Part 1**	**84 days?**	**Mon 01-03-10**	**Thu 24-06-10**
3	▦	Topic Investigation/Negotiation	8 days?	Mon 01-03-10	Wed 10-03-10
4	▦	Project Allocation	0 days	Wed 10-03-10	Wed 10-03-10
5	▦	Project Specification Negotiation	9 days?	Thu 11-03-10	Tue 23-03-10
6	▦	Project Specification	0 days	Tue 23-03-10	Tue 23-03-10
7	▦	Implementation Investigation	27 days	Wed 24-03-10	Thu 29-04-10
8	▦	Implementation Choice	0 days	Thu 29-04-10	Thu 29-04-10
9	▦	Environmental Factors Research	5 days?	Mon 24-05-10	Fri 28-05-10
10	▦	Requirements Preliminary Research	44 days	Wed 24-03-10	Mon 24-05-10
11	▦	Project Coding (Local app)	40 days	Fri 30-04-10	Thu 24-06-10
12		Project Appreciation	0 days	Mon 24-05-10	Mon 24-05-10
13	▦	Progress Assessment	0 days?	Thu 17-06-10	Thu 17-06-10
14		**ENG4112 Research Project Part 2**	**90 days**	**Fri 25-06-10**	**Thu 28-10-10**
15		Local App Testing/Redesign	10 days	Fri 25-06-10	Thu 08-07-10
16	▦	Write Draft Dissertation	55 days	Mon 12-07-10	Fri 24-09-10
17	▦	Project Coding (Networked app)	35 days	Mon 26-07-10	Fri 10-09-10
18		Small office testing	15 days	Mon 13-09-10	Fri 01-10-10
19	▦	Partial Draft Dissertation	0 days	Fri 24-09-10	Fri 24-09-10
20	▦	Write Final Dissertation	24 days	Mon 27-09-10	Thu 28-10-10
21	▦	Project Performance	0 days	Thu 28-10-10	Thu 28-10-10

Appendix C – PowerMan Client Code

The PowerMan Client code is included in the following appendices:

- Appendix C1 – Form1.cs
- Appendix C2 – Form1.Designer.cs
- Appendix C3 – Program.cs
- Appendix C4 – DateTimeState.cs
- Appendix C5 – ServerPacket.cs

Appendix C1 – Form1.cs

```
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Management;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using System.Threading;
using System.Net;
using System.Net.Sockets;
using System.Text.RegularExpressions;
using ClassLibrary1;

namespace PowerMan_Client1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            timer1.Enabled = true;
            formDTState = new DateTimeState();
            lastBootTime = new DateTime();
        }

        public enum EXECUTION_STATE : uint
        {
            ES_SYSTEM_REQUIRED = 0x00000001,
            ES_DISPLAY_REQUIRED = 0x00000002,
            ES_CONTINUOUS = 0x80000000
        }

        // declare windows API functions
        [DllImport("kernel32.dll")]
        extern static IntPtr CreateWaitableTimer(IntPtr lpTimerAttributes, bool
bManualReset, string lpTimerName);
        [DllImport("kernel32.dll")]
        extern static bool SetWaitableTimer(IntPtr hTimer, [In] ref long
pDueTime, int lPeriod, IntPtr pfnCompletionRoutine, IntPtr
lpArgToCompletionRoutine, bool fResume);
        [DllImport("kernel32", SetLastError = true, ExactSpelling = true)]
        extern static Int32 WaitForSingleObject(IntPtr handle, uint
milliseconds);
        [DllImport("kernel32.dll")]
        extern static bool CancelWaitableTimer(IntPtr hTimer);
        [DllImport("Kernel32.dll", CharSet = CharSet.Auto, SetLastError = true)]
        extern static EXECUTION_STATE SetThreadExecutionState(EXECUTION_STATE
state);
        [DllImport("user32.dll")]
        extern static bool GetLastInputInfo(ref LastInputStructure plii);

        // Form event methods
        private void Form1_Load(object sender, EventArgs e)
        {
            // create a SELECT query using 'LastBootUpTime' param
            SelectQuery query1 = new SelectQuery("SELECT LastBootUpTime FROM
Win32_OperatingSystem WHERE Primary='true'");
            // create a new object searcher and pass to the select query
            ManagementObjectSearcher searcher1 = new
ManagementObjectSearcher(query1);
            // get the datetime value and set the local boot time variable to
contain that value

            foreach (ManagementObject mgtObject1 in searcher1.Get())
            {
                // convert from universal time to normal format
                lastBootTime =
ManagementDateTimeConverter.ToDateTime(mgtObject1.Properties["LastBootUpTime"].V
alue.ToString());
                txtBootDate.Text = lastBootTime.ToLongDateString();
                txtBootTime.Text = lastBootTime.ToLongTimeString();
            }

            SystemEvents.PowerModeChanged += new
PowerModeChangedEventHandler(SystemEvents_PowerModeChanged);

            // create a select query using 'MacAddress.IPAddress' param
            ObjectQuery query2 = new ObjectQuery("Select MacAddress,IPAddress
from Win32_NetworkAdapterConfiguration where IPEnabled=TRUE");
            // create a new object searcher and pass to the select query
            ManagementObjectSearcher searcher2 = new
ManagementObjectSearcher(query2);

            foreach (ManagementObject mgtObject2 in searcher2.Get())
            {
                clientMACString = mgtObject2["MacAddress"].ToString();
                clientIPString =
((System.Array)(mgtObject2["IPAddress"])).GetValue(0).ToString();
            }

            // convert MAC to standard format for storage
            string formattedMAC = FormatMAC(clientMACString);

            // change formatted mac for file load
            string temp = formattedMAC.Replace(":", "");
            // attempt to load file
            myClient = (Client)FileToObject("c:\\client" + temp + ".dat");
            // if file not read or found then create new instance of client
            if (myClient == null)
            {
                myClient = new Client(formattedMAC);
            }

            //set initial time if not set
            if (!myClient.GetInitialTimeSet())
            {
                myClient.SetInitialTime(DateTime.Now);
                myClient.SetInitialTimeSet(true);
            }

            // set session progressive time
            myClient.SetSessionProgressiveTime(DateTime.Now);
            myClient.SetSessionTimeReset(true);

            // set session start time;
            startTime = DateTime.Now;

            // ensure trackbar is restored to last position
            trkBarEfficiency.Value = myClient.GetTrackBarIndex();
            trackBar1_Scroll(this, e);

            // update text boxes
            clientTextBoxMAC.Text = myClient.GetClientID();
            clientTextBoxIP.Text = clientIPString;
            serverTextBoxIP.Text = myClient.GetServerIP();
            serverTextBoxPort.Text = myClient.GetServerPort().ToString();

            // set server IP and port buttons to false
            buttonUpdateIP.Enabled = false;
            buttonUpdatePort.Enabled = false;
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            txtDate.Text = DateTime.Now.ToLongDateString();
            txtTime.Text = DateTime.Now.ToLongTimeString();
            TimeSpan elapsedTime = DateTime.Now - lastBootTime;
            txtElapsedTime.Text = elapsedTime.Hours.ToString() + "hrs, " +
elapsedTime.Minutes.ToString() + "mins, " +
                elapsedTime.Seconds.ToString() + "secs";

            if (formDTState.GetPostWakeReboot())
            {
                formDTState.SetPostWakeReboot(false);
                // this will be instaneous restart
                SetRestart("instant");
            }

            // update power usage figures
            // get high power time
            if (myClient.GetSessionTimeReset())
            {
                TimeSpan current = DateTime.Now -
myClient.GetSessionProgressiveTime();
                myClient.SetSessionProgressiveTime(DateTime.Now);
                TimeSpan old = myClient.GetHighPowerTime();
                myClient.SetHighPowerTime(current + old);
            }

            // low power will be total minus high power
            TimeSpan total = DateTime.Now - myClient.GetInitialTime();
            myClient.SetLowPowerTime(total - myClient.GetHighPowerTime());

            // get average daily hours turned on
            double averageDailyHoursOn = myClient.GetHighPowerTime().TotalHours
/ total.TotalDays;
            // get average daily hours turned off
            double averageDailyHoursOff = myClient.GetLowPowerTime().TotalHours
/ total.TotalDays;

            // using 0.250 kWh for PC on and 30 kWh for hibernate/sleep
            // set energy used
            myClient.SetEnergyUsed(averageDailyHoursOn * 0.25);
            //set energy saved
            myClient.SetEnergySaved(averageDailyHoursOff * 0.03);

            // calculate carbon saved use 0.956 kg per kWh
            double energySaved = myClient.GetEnergySaved();
            myClient.SetCarbonSaved(energySaved * 0.956);

            // calculate money saved use $0.17 per kWh
            myClient.SetMoneySaved(energySaved * 0.17);
            // publish saving figures
            string temp1 = string.Format("{0:0.00}", myClient.GetEnergyUsed());
            string temp2 = string.Format("{0:0.00}", myClient.GetEnergySaved());
            string temp3 = string.Format("{0:0.00}", myClient.GetCarbonSaved());
            string temp4 = string.Format("{0:0.00}", myClient.GetMoneySaved());
            lblEnergyUsed.Text = temp1 + " kWh";
            lblEnergySaved.Text = temp2 + " kWh";
            lblCarbonSaved.Text = temp3 + " kg";
            lblMoneySaved.Text = "$" + temp4;

            if (hibernationFlag)
            {
                startResumeSpan = DateTime.Now - postHibernateWakeTime;
            }
            else
            {
                startResumeSpan = DateTime.Now - startTime;
            }

            //establish connection after 15 seconds into commencement
            if (startResumeSpan.TotalSeconds > 15 && !connectionFlag)
            {
                // set flag to ensure connection is only tried once
                connectionFlag = true;
                //call connect function
                ConnectToServer();
                // set port and ip box so that it can not be changed
                serverTextBoxPort.Text = myClient.GetServerPort().ToString();
                serverTextBoxPort.BackColor =
System.Drawing.Color.LightSteelBlue;
                serverTextBoxPort.ReadOnly = true;
                buttonUpdatePort.Enabled = false;
                serverTextBoxIP.Text = myClient.GetServerIP();
                serverTextBoxIP.BackColor = System.Drawing.Color.LightSteelBlue;
                serverTextBoxIP.ReadOnly = true;
                buttonUpdateIP.Enabled = false;
            }
            else if (!connectionFlag)
            {
                textBoxConnectStatus.Text = String.Format("Connecting in {0}
secs", (15 - startResumeSpan.Seconds));
                // check every 2 seconds
                if (startResumeSpan.Seconds % 2 == 0)
                {
                    textBoxConnectStatus.BackColor = System.Drawing.Color.Red;
                }
                else
                {
                    textBoxConnectStatus.BackColor =
System.Drawing.Color.Orange;
                }
            }

            //wait 20 seconds before sending first packet
            if ((startResumeSpan.TotalSeconds > 20) && !sendFlag)
            {
                // set flag to ensure only 1 send
                sendFlag = true;
                SendClientDetails();
            }

            // check socket status every 2 seconds
            if ((startResumeSpan.TotalSeconds > 15) && (startResumeSpan.Seconds
% 2 == 0))
```

```
            {
                if (SocketTest())
                {
                    UpdateControls(true);
                }
                else
                {
                    UpdateControls(false);
                }
            }

            if ((DateTime.Now >= formDTState.GetEndDateTime()) &&
formDTState.GetEndTimeSet() && !formDTState.GetWakeOccurred())
            {
                if (formDTState.GetShutdownMethod() == "Hibernation")
                {
                    SetHibernate();
                }
                else if (formDTState.GetShutdownMethod() == "Shutdown")
                {
                    SetShutdown();
                }
                else // must be restart
                {
                    SetRestart("notinstant");
                }
            }

            // Measure system idle time
            // time in milliseconds since the last time computer was started
            int systemUptime = Environment.TickCount;
            // The tick at which the last input was recorded
            int lastInputTicks = 0;
            // The number of ticks that passed since last input
            int realIdleTicks = 0;
            // Set the structure
            LastInputStructure aLastInput = new LastInputStructure();
            aLastInput.structureSize = (uint)Marshal.SizeOf(aLastInput);
            aLastInput.idleCountTime = 0;
            // If we have a value from the function
            if (GetLastInputInfo(ref aLastInput))
            {
                // Get the number of ticks at the point when the last activity
was seen
                lastInputTicks = (int)aLastInput.idleCountTime;
                // Number of idle ticks = system uptime ticks - number of ticks
at last input
                realIdleTicks = systemUptime - lastInputTicks;
            }

            // convert idle time to seconds and minutes
            int realIdleSeconds = realIdleTicks / 1000;
            // get actual time of last input
            lastInputTime = DateTime.Now.AddSeconds(-realIdleSeconds);

            // ensure PC will hibernate after periods of set idle time even if
woken without activity
            if (hibernationFlag)
            {
                if (lastInputTime < postHibernateWakeTime)
                {
                    idleSpan = DateTime.Now - postHibernateWakeTime;
                }
                else
```

```
                    {
                        idleSpan = TimeSpan.FromSeconds(realIdleSeconds);
                    }
                }
                else
                {
                    idleSpan = TimeSpan.FromSeconds(realIdleSeconds);
                }

                txtIdleTime.Text = idleSpan.Hours.ToString() + "hrs, " +
idleSpan.Minutes.ToString() + "mins, " +
                idleSpan.Seconds.ToString() + "secs";

                if ((myClient.GetEfficiencyValue() != 0) && (idleSpan.TotalMinutes
>= myClient.GetEfficiencyValue()))
                {
                    //****
                    //testString1 = testString1 + " tp1 " +
DateTime.Now.ToLongTimeString();
                    //textBox1.Text = testString1;
                    //****
                    SetHibernate();
                }
            }

//********************************************************************************
**
//****************************** Event Driven methods
******************************
//********************************************************************************
**

        private void SystemEvents_PowerModeChanged(object sender,
PowerModeChangedEventArgs e)
        {
            // Suspend
            if (e.Mode == PowerModes.Suspend)
            {
                // check for hibernate done outside of application
                myClient.SetSessionTimeReset(false);
                hibernationFlag = false;
                SetHibernate();
                //*****
                //testString1 = testString1 + " tp2 " +
DateTime.Now.ToLongTimeString();
                //textBox1.Text = testString1;
                //****
            }
            //Resume
            if (e.Mode == PowerModes.Resume)
            {


                myClient.SetSessionProgressiveTime(DateTime.Now);
                myClient.SetSessionTimeReset(true);

                if (!SocketTest())
                {
                    // reset time and connection flags
                    hibernationFlag = true;
                    postHibernateWakeTime = DateTime.Now;
                    connectionFlag = false;
                    sendFlag = false;
                }
                //****
                //testString1 = testString1 + " tp3 " +
DateTime.Now.ToLongTimeString();
                //textBox1.Text = testString1;
                //****

                // when waking clear advertised waking time (if past)
                if (DateTime.Now >= formDTState.GetStartDateTime())
                {
                    formDTState.SetStartTimeSet(false);
                    txtWakeTime.Text = "-- Not set --";
                    txtWakeTime.BackColor =
System.Drawing.SystemColors.ScrollBar;
                    // resuming here after hibernation set state to indicate a
shutdown action has occured
                    formDTState.SetWakeOccurred(true);
                }
            }
        }

        private void btnSetEnd_Click(object sender, EventArgs e)
        {
            // check that end time is less than start time if set
            if ((formDTState.GetStartTimeSet()) &&
((dTPickStop.Value.AddMinutes(1) > formDTState.GetStartDateTime())))
            {
                MessageBox.Show("If start time SET, end time must be less than
set start time!\n"
                    + "Please select a new end time, or CLEAR start time!\n");
                formDTState.SetShutdownMethod("-- Not set --");
                formDTState.SetEndTimeSet(false);
                formDTState.SetWakeOccurred(false); //reset if set previously
                cmbBoxSessionEnd.SelectedIndex = -1;
                txtStopTime.Text = "-- Not set --";
                txtStopMethod.Text = formDTState.GetShutdownMethod();
                txtStopTime.BackColor = System.Drawing.SystemColors.ScrollBar;
                txtStopMethod.BackColor = System.Drawing.SystemColors.ScrollBar;
                return;
            }
            // check end time is at least a minute greater than current time
            if (dTPickStop.Value < DateTime.Now.AddMinutes(1))
            {
                MessageBox.Show("End time must be at least a minute greater than
current time!\n");
                formDTState.SetShutdownMethod("-- Not set --");
                formDTState.SetEndTimeSet(false);
                formDTState.SetWakeOccurred(false); //reset if set previously
                cmbBoxSessionEnd.SelectedIndex = -1;
                txtStopTime.Text = "-- Not set --";
                txtStopMethod.Text = formDTState.GetShutdownMethod();
                txtStopTime.BackColor = System.Drawing.SystemColors.ScrollBar;
                txtStopMethod.BackColor = System.Drawing.SystemColors.ScrollBar;
                return;
            }

            if (cmbBoxSessionEnd.SelectedIndex != -1)
            {
                formDTState.SetEndDateTime(dTPickStop.Value);

formDTState.SetShutdownMethod(cmbBoxSessionEnd.SelectedItem.ToString());
                formDTState.SetEndTimeSet(true);
                formDTState.SetWakeOccurred(false); //reset if set previously
                txtStopTime.Text =
formDTState.GetEndDateTime().ToLongTimeString();
                txtStopMethod.Text = formDTState.GetShutdownMethod();
                txtStopTime.BackColor = System.Drawing.Color.Red;
                txtStopMethod.BackColor = System.Drawing.Color.Red;
            }
            else
            {
                MessageBox.Show("You must select a Session End Method!");
            }
        }

        private void btnClearEnd_Click(object sender, EventArgs e)
        {
            formDTState.SetShutdownMethod("-- Not set --");
            formDTState.SetEndTimeSet(false);
            formDTState.SetWakeOccurred(false); //reset if set previously
            cmbBoxSessionEnd.SelectedIndex = -1;
            txtStopTime.Text = "-- Not set --";
            txtStopMethod.Text = formDTState.GetShutdownMethod();
            txtStopTime.BackColor = System.Drawing.SystemColors.ScrollBar;
            txtStopMethod.BackColor = System.Drawing.SystemColors.ScrollBar;
        }

        private void btnSetStart_Click(object sender, EventArgs e)
        {
            // check that start time is greater than end time if set
            if ((formDTState.GetEndTimeSet()) && (dTPickStart.Value <
formDTState.GetEndDateTime().AddMinutes(1)))
            {
                MessageBox.Show("If end time SET, start time must be greater
than set end time!\n"
                    + "Please select a new start time, or CLEAR start time!\n");
                formDTState.SetStartTimeSet(false);
                txtWakeTime.Text = "-- Not set --";
                txtWakeTime.BackColor = System.Drawing.SystemColors.ScrollBar;
                return;
            }

            if (dTPickStart.Value < DateTime.Now.AddMinutes(1))
            {
                MessageBox.Show("Start time must be at least a minute greater
than current time!\n");
                formDTState.SetStartTimeSet(false);
                txtWakeTime.Text = "-- Not set --";
                txtWakeTime.BackColor = System.Drawing.SystemColors.ScrollBar;
                return;
            }

            formDTState.SetStartDateTime(dTPickStart.Value);
            formDTState.SetStartTimeSet(true);

            txtWakeTime.Text =
formDTState.GetStartDateTime().ToLongTimeString();
            txtWakeTime.BackColor = System.Drawing.Color.Lime;
        }

        private void btnClearStart_Click(object sender, EventArgs e)
        {
            formDTState.SetStartTimeSet(false);
            txtWakeTime.Text = "-- Not set --";
            txtWakeTime.BackColor = System.Drawing.SystemColors.ScrollBar;
        }
        private void btnHibernate_Click(object sender, EventArgs e)
        {
            SetHibernate();
        }

        private void btnShutdown_Click(object sender, EventArgs e)
        {
            SetShutdown();
        }

        private void btnRestart_Click(object sender, EventArgs e)
        {
            SetRestart("notinstant");
        }

        private void buttonClose_Click(object sender, EventArgs e)
        {
            CloseApplication();
        }

        private void buttonUpdateIP_Click(object sender, EventArgs e)
        {
            // See if we have a valid IP address
            bool temp = IsValidIP(serverTextBoxIP.Text.ToString());

            if (!temp)
            {
                MessageBox.Show("Must enter valid IP Address in ***.***.***.***
format!");
                serverTextBoxIP.Text = myClient.GetServerIP();
                buttonUpdateIP.Enabled = false;
                return;
            }
            else
            {
                myClient.SetServerIP(serverTextBoxIP.Text.ToString());
                buttonUpdateIP.Enabled = false;
                return;
            }
        }

        private void buttonUpdatePort_Click(object sender, EventArgs e)
        {
            bool temp = IsValidPort(int.Parse(serverTextBoxPort.Text));

            if (!temp)
            {
                MessageBox.Show("Must enter port in range 50000-65000!");
                serverTextBoxPort.Text = myClient.GetServerPort().ToString();
                buttonUpdatePort.Enabled = false;
                return;
            }
            else
            {
                myClient.SetServerPort(int.Parse(serverTextBoxPort.Text));
                buttonUpdatePort.Enabled = false;
                return;
            }
        }

        private void serverTextBoxIP_TextChanged(object sender, EventArgs e)
        {
            buttonUpdateIP.Enabled = true;
        }
        private void serverTextBoxPort_TextChanged(object sender, EventArgs e)
        {
            buttonUpdatePort.Enabled = true;
        }

        private void trackBar1_Scroll(object sender, EventArgs e)
        {
            myClient.SetTrackBarIndex(trkBarEfficiency.Value);

myClient.SetEfficiencyValue(efficiencyScale[trkBarEfficiency.Value]);
            if (trkBarEfficiency.Value == 0)
            {
                txtEfficiency.Font = new Font(txtEfficiency.Font,
FontStyle.Bold);
                txtEfficiency.ForeColor =
System.Drawing.SystemColors.WindowText;
                txtEfficiency.Text = "-- Never --";
                txtEfficiency.BackColor = System.Drawing.SystemColors.ScrollBar;
            }
            else
            {
                txtEfficiency.Font = new Font(txtEfficiency.Font,
FontStyle.Regular);
                txtEfficiency.ForeColor = System.Drawing.Color.Green;
                txtEfficiency.BackColor = System.Drawing.SystemColors.Window;
                txtEfficiency.Text = "after " +
myClient.GetEfficiencyValue().ToString() + " minute(s)";
            }
        }

//******************************************************************************
*
//**************************** Other methods
************************************
//******************************************************************************
*
        // Calls the system hibernate function or calls the waitable timer
        // function if a wake time has been set.
        private void SetHibernate()
        {
            myClient.SetSessionTimeReset(false);
            CloseSocket();
            SaveToFile();
            if (formDTState.GetStartTimeSet())
            {
                // reset state to indicate wake action has taken place
                formDTState.SetStartTimeSet(false);

                //****
                //testString1 = testString1 + " tp4 " +
DateTime.Now.ToLongTimeString();
                //textBox1.Text = testString1;
                //****
                // will be here if prior to wakeable timer
                SetWaitForWakeUpTime(formDTState.GetStartDateTime());
                // will be here if woke by wakeable timer
                //****
                //testString1 = testString1 + " tp5 " +
DateTime.Now.ToLongTimeString();
                //textBox1.Text = testString1;
                //****

                // reset time and connection flags
                hibernationFlag = true;
                postHibernateWakeTime = DateTime.Now;
                connectionFlag = false;
                sendFlag = false;

                if (chkBoxReboot.Checked)
                {
                    formDTState.SetPostWakeReboot(true);
                }

                // when waking clear advertised waking time (if past)
                if (DateTime.Now >= formDTState.GetStartDateTime())
                {
                    formDTState.SetStartTimeSet(false);
                    txtWakeTime.Text = "-- Not set --";
                    txtWakeTime.BackColor =
System.Drawing.SystemColors.ScrollBar;
                    // resuming here after hibernation set state to indicate a
shutdown action has occured
                    formDTState.SetWakeOccurred(true);
                }

                // when waking clear advertised end time (if past)
                if (DateTime.Now >= formDTState.GetEndDateTime())
                {
                    formDTState.SetShutdownMethod("-- Not set --");
                    formDTState.SetEndTimeSet(false);
                    formDTState.SetWakeOccurred(false); //reset if set
previously
                    cmbBoxSessionEnd.SelectedIndex = -1;
                    txtStopTime.Text = "-- Not set --";
                    txtStopMethod.Text = formDTState.GetShutdownMethod();
                    txtStopTime.BackColor =
System.Drawing.SystemColors.ScrollBar;
                    txtStopMethod.BackColor =
System.Drawing.SystemColors.ScrollBar;
                }

            }
            else // hibernate indefinately
            {
                //****
                //testString1 = testString1 + " tp6 " +
DateTime.Now.ToLongTimeString();
                //textBox1.Text = testString1;
                //****
                Application.SetSuspendState(PowerState.Hibernate, true, false);
            }
        }

        // Initiates the formal shutdown process together with generating an
        // appropriate system message.
        private void SetShutdown()
        {
            myClient.SetSessionTimeReset(false);
            CloseSocket();
            SaveToFile();
            Process.Start("Shutdown", "/s /f /c \"Shutdown initiated by PowerMan
power management application!\"");
            Close();
        }

        // Initiates the formal shutdown (restart) process together with
generating an
```

```
        // appropriate system message.
        private void SetRestart(string when)
        {
            myClient.SetSessionTimeReset(false);
            CloseSocket();
            SaveToFile();
            if (when == "instant")
            {
                // will be instant directly after wake from hibernation if
selected
                Process.Start("Shutdown", "/r /f /t 00");
                Close();
            }
            else
            {
                Process.Start("Shutdown", "/r /f /c \"Restart initiated by
PowerMan power management application!\"");
                Close();
            }
        }

        // method to gracefully close the PowerMan application.
        private void CloseApplication()
        {
            myClient.SetSessionTimeReset(false);
            CloseSocket();
            SaveToFile();
            Close();
        }

        // method returns a Boolean true value if the entered IP address is in
"***.***.***.***" correct format,
        // else returns false.
        public bool IsValidIP(string IPAddress)
        {
            // create IP pattern for test
            string IPPattern =  @"^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-
9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$";
            Regex expr1 = new Regex(IPPattern);
            //check to make sure an ip address was provided
            if (IPAddress == "")
            {
                return false;
            }
            else
            {
                return(expr1.IsMatch(IPAddress, 0));
            }
        }

        // This method returns a true value if the entered port is in the
correct range (50000-65000),
        // else returns false.
        public bool IsValidPort(int portNumber)
        {
            if ((portNumber >= 50000) && (portNumber <= 65000))
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        //method recognies 3 formats of MAC address and returns a consistent
format
        // ':' seperated, '-' seperated, and no seperation between bytes
        private string FormatMAC(string aMACString)
        {
            //get string to common format before
            string temp1 = aMACString.Replace(":", "");
            string temp2 = temp1.Replace("-", "");
            string temp3 = temp2.ToUpper();
            // temp2 should now be length 12 chars
            StringBuilder final = new StringBuilder(temp3.Length + 5);
            for (int i = 1; i < (temp3.Length + 1); i++)
            {
                final.Append(temp3[i - 1]);
                if ((i % 2 == 0) && (i != temp3.Length))
                {
                    final.Append(':');
                }
            }
            return final.ToString();
        }

        // Method to close sockets and place a null refernce on socket variables
        private void CloseSocket()
        {
            // test for valid socket
            if (clientSocket != null)
            {
                if (clientSocket.Connected)
                {
                    clientSocket.Shutdown(SocketShutdown.Both);
                    clientSocket = null;
                }
            }
        }

        // Method to test for a valid and connected socket
        private bool SocketTest()
        {
            if (clientSocket != null)
            {
                return clientSocket.Connected;
            }
            else
            {
                return false;
            }
        }

        // method to save client data to file
        private void SaveToFile()
        {
            string temp = myClient.GetClientID().Replace(":", "");
            ObjectToFile(myClient, "c:\\client" + temp + ".dat");
        }

        // create structure for idle time function
        internal struct LastInputStructure
        {
            public uint structureSize;
            public uint idleCountTime;
        }
        // Creates and cancels the waitable timer handle and suspends the 
computer.
        public void SetWaitForWakeUpTime(DateTime dt)
        {
            // create the Timer handle
            waitTimeHandle = CreateWaitableTimer(IntPtr.Zero, true, 
"ThisWaitableTimer");
            // generate ticks until wake time
            long wakeTicks = dt.ToUniversalTime().Ticks;
            // allow wake up events in hibernate mode
            SetWaitableTimer(waitTimeHandle, ref wakeTicks, 0, IntPtr.Zero, 
IntPtr.Zero, true);
            // set hibernate state during timer counting
            Application.SetSuspendState(PowerState.Hibernate, true, false);
            // delete timer and handle
            CancelWaitableTimer(waitTimeHandle);
            // enable display at wakeup without user prompt
            SetThreadExecutionState(EXECUTION_STATE.ES_DISPLAY_REQUIRED);
        }

        // method to connect to server after called within timer function
        public void ConnectToServer()
        {
            try
            {
                UpdateControls(false);
                // Create the socket instance
                clientSocket = new Socket(AddressFamily.InterNetwork, 
SocketType.Stream, ProtocolType.Tcp);
                // Get the remote IP address
                IPAddress ip = IPAddress.Parse(myClient.GetServerIP());
                int iPortNo = myClient.GetServerPort();
                // Create the end point
                IPEndPoint ipEnd = new IPEndPoint(ip, iPortNo);
                // Connect to the specific remote ip address and port number
                clientSocket.Connect(ipEnd);
                // test if connected
                if (SocketTest())
                {
                    //Wait for data asynchronously
                    WaitForData();
                }
            }
            catch (SocketException se)
            {
                UpdateControls(false);
                //****
                //testString1 = testString1 + " tp7 " + 
DateTime.Now.ToLongTimeString();
                //textBox1.Text = testString1.ToString();
                //****
            }
        }

        // Start waiting for data from the server
        public void WaitForData()
        {
            try
            {
                if (clientpfnCallBack == null)
                {
                    clientpfnCallBack = new AsyncCallback(OnDataReceived);
                }
                ServerPacket aServerPkt = new ServerPacket();
                aServerPkt.SetSocket(clientSocket);
                // Start listening to the data asynchronously
                clientAsyncResult = clientSocket.BeginReceive(aServerPkt.buffer,
0, aServerPkt.buffer.Length, SocketFlags.None, clientpfnCallBack, aServerPkt);
            }
            catch (SocketException se)
            {
                //****
                //testString1 = testString1 + " tp8 " +
DateTime.Now.ToLongTimeString();
                //textBox1.Text = testString1.ToString();
                //****
            }
        }

        // Call back function which is invoked when the socket detects any
writing on the stream
        public void OnDataReceived(IAsyncResult asyn)
        {
            try
            {
                ServerPacket socketData = (ServerPacket)asyn.AsyncState;
                // set bool to determine if wait will occur post receive
                bool wait = false;
                // get length of received stream
                int read = socketData.GetSocket().EndReceive(asyn);
                if (read > 0)
                {
                    for (int i = 0; i < read; i++)
                    {
                        socketData.TransmissionBuffer.Add(socketData.buffer[i]);
                    }
                    byte[] newBuffer = socketData.TransmissionBuffer.ToArray();
                    String serverMessage = (String)ByteArrayToObject(newBuffer);
                    switch (serverMessage)
                    {
                        case "Restart":
                            SetRestart("notinstant");
                            break;

                        case "Hibernate":
                            SetHibernate();
                            break;

                        case "Shutdown":
                            SetShutdown();
                            break;

                        case "Refresh":
                            SendClientDetails();
                            wait = true;
                            break;

                        case "CloseApplication":
                            CloseApplication();
                            break;
                    }
                }
                if (wait)
                {
                    //Wait for data asynchronously
                    WaitForData();
                }
            }
            catch (SocketException se)
            {
                //****
                //testString1 = testString1 + " tp9 " +
DateTime.Now.ToLongTimeString();
                //textBox1.Text = testString1.ToString();
                //****
            }
        }

        // method to send client object details
        private void SendClientDetails()
        {
            byte[] byData = ObjectToByteArray(myClient);
            if (SocketTest())
            {
                clientSocket.Send(byData);
            }
        }

        // This method converts the client object to a byte array so that it can
be sent over the socket.
        public object ByteArrayToObject(byte[] serverByteArray)
        {
            try
            {
                // convert byte array to memory stream
                System.IO.MemoryStream myMemoryStream = new
System.IO.MemoryStream(serverByteArray);
                // create new BinaryFormatter
                System.Runtime.Serialization.Formatters.Binary.BinaryFormatter
myBinaryFormatter
                        = new
System.Runtime.Serialization.Formatters.Binary.BinaryFormatter();
                // set memory stream position to starting point
                myMemoryStream.Position = 0;
                // Deserializes a stream into an object graph and return as a
object.
                return myBinaryFormatter.Deserialize(myMemoryStream);
            }
            catch (Exception ex)
            {
                // Error
                Console.WriteLine("Exception caught in process: {0}",
ex.ToString());
            }
            // Error occured, return null
            return null;
        }

        // Will provide an appropriate indication of socket status to the
PowerMan GUI.
        private void UpdateControls(bool connected)
        {
            if (connected)
            {
                textBoxConnectStatus.Text = "Connected to remote host";
                textBoxConnectStatus.BackColor = System.Drawing.Color.Lime;
            }
            else
            {
                textBoxConnectStatus.Text = "Not connected to remote host";
                textBoxConnectStatus.BackColor = System.Drawing.Color.Red;
            }
        }

            // Method to convert object to byte array then save to file
        public bool ObjectToFile(object clientObject, string clientFileName)
          {
              try
              {
                  // create new memory stream
                  System.IO.MemoryStream fileMemoryStream = new
System.IO.MemoryStream();
                  // create new BinaryFormatter
                  System.Runtime.Serialization.Formatters.Binary.BinaryFormatter
fileBinaryFormatter
                              = new
System.Runtime.Serialization.Formatters.Binary.BinaryFormatter();
                  // Serializes an object, or graph of connected objects, to the
given stream.
                  fileBinaryFormatter.Serialize(fileMemoryStream, clientObject);
                  // convert stream to byte array
                  byte[] fileByteArray = fileMemoryStream.ToArray();
                  // Open file for writing
                  System.IO.FileStream clientFileStream = new
System.IO.FileStream(clientFileName, System.IO.FileMode.Create,
System.IO.FileAccess.Write);
                  // Writes a block of bytes to this stream using data from a
byte array.
                  clientFileStream.Write(fileByteArray.ToArray(), 0,
fileByteArray.Length);
                  // close file stream
                  clientFileStream.Close();
                  // cleanup
                  fileMemoryStream.Close();
                  fileMemoryStream.Dispose();
                  fileMemoryStream = null;
                  fileByteArray = null;
                  return true;
              }
              catch (Exception ex)
              {
                  // Error
                  Console.WriteLine("Exception caught in process: {0}",
ex.ToString());
              }
          // Error occured, return null
              return false;
        }

        // method open saved file and then convert to obejct
        public object FileToObject(string clientFileName)
       {
              try
              {
                  // Open file for reading
                  System.IO.FileStream clientFileStream = new
System.IO.FileStream(clientFileName, System.IO.FileMode.Open,
System.IO.FileAccess.Read);
                  // attach filestream to binary reader
                  System.IO.BinaryReader fileBinaryReader = new
System.IO.BinaryReader(clientFileStream);
                  // get total byte length of the file
                  long _TotalBytes = new
System.IO.FileInfo(clientFileName).Length;
                  // read entire file into buffer
                  byte[] fileByteArray =
fileBinaryReader.ReadBytes((Int32)_TotalBytes);
                  // close file reader and do some cleanup
                  clientFileStream.Close();
                  clientFileStream.Dispose();
                  clientFileStream = null;
                  fileBinaryReader.Close();
                  // convert byte array to memory stream
                  System.IO.MemoryStream fileMemoryStream = new
System.IO.MemoryStream(fileByteArray);
                  // create new BinaryFormatter
                  System.Runtime.Serialization.Formatters.Binary.BinaryFormatter
fileBinaryFormatter
                              = new
System.Runtime.Serialization.Formatters.Binary.BinaryFormatter();
                  // set memory stream position to starting point
                  fileMemoryStream.Position = 0;
                  // Deserializes a stream into an object graph and return as a
object.
                  return fileBinaryFormatter.Deserialize(fileMemoryStream);
              }
              catch (Exception ex)
              {
                  // Error
                  Console.WriteLine("Exception caught in process: {0}",
ex.ToString());
              }
              // Error occured, return null
              return null;
        }

        // method to convert object to byte array
          public byte[] ObjectToByteArray(object clientObject)
          {
              try
              {
                  // create new memory stream
                  System.IO.MemoryStream byteMemoryStream = new
System.IO.MemoryStream();
                // create new BinaryFormatter
                  System.Runtime.Serialization.Formatters.Binary.BinaryFormatter
byteBinaryFormatter
                              = new
System.Runtime.Serialization.Formatters.Binary.BinaryFormatter();
                  // Serializes an object, or graph of connected objects, to the
given stream.
                  byteBinaryFormatter.Serialize(byteMemoryStream, clientObject);
                  // convert stream to byte array and return
                  return byteMemoryStream.ToArray();
              }
              catch (Exception ex)
              {
                  // Error
                  Console.WriteLine("Exception caught in process: {0}",
ex.ToString());
              }
              // Error occured, return null
              return null;
        }

        // variables/fields
        private DateTimeState formDTState;
        private DateTime lastBootTime;
        private string clientMACString;
        private string clientIPString;
        private Client myClient;
        static IntPtr waitTimeHandle;
        IAsyncResult clientAsyncResult;
        public AsyncCallback clientpfnCallBack;
        public Socket clientSocket;
        private DateTime startTime = new DateTime();
        private TimeSpan startResumeSpan;
        private bool connectionFlag = false;
        private bool sendFlag = false;
        private int[] efficiencyScale = new int[7] {0, 60, 50, 40, 30, 20, 10};
        private bool hibernationFlag = false;
        private DateTime postHibernateWakeTime = new DateTime();
        private DateTime lastInputTime = new DateTime();
        private TimeSpan idleSpan;
        //private string testString1;
    }
}
```

Appendix C2 – Form1.Designer.cs

```
namespace PowerMan_Client1
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        private System.Windows.Forms.Label lblWelcome;
        private System.Windows.Forms.Timer timer1;
        private System.Windows.Forms.Label lblDate;
        private System.Windows.Forms.Label lblTime;
        private System.Windows.Forms.Button btnShutdown;
        private System.Windows.Forms.Button btnRestart;
        private System.Windows.Forms.TextBox txtBootDate;
        private System.Windows.Forms.TextBox txtBootTime;
        private System.Windows.Forms.TextBox txtElapsedTime;
        private System.Windows.Forms.GroupBox boxControls;
        private System.Windows.Forms.GroupBox boxDateTime;
        private System.Windows.Forms.GroupBox boxOpData;
        private System.Windows.Forms.TextBox txtTime;
        private System.Windows.Forms.TextBox txtDate;
        private System.Windows.Forms.Label lblBootDate;
        private System.Windows.Forms.Label lblElapsedTime;
        private System.Windows.Forms.Label lblBootTime;
        private System.Windows.Forms.Button btnHibernate;
        private System.Windows.Forms.DateTimePicker dTPickStop;
        private System.Windows.Forms.GroupBox boxSchedule;
        private System.Windows.Forms.GroupBox boxEndSession;
        private System.Windows.Forms.GroupBox boxStartSession;
        private System.Windows.Forms.DateTimePicker dTPickStart;
        private System.Windows.Forms.Label lblSessionEnd;
        private System.Windows.Forms.Label lblSessionStart;
        private System.Windows.Forms.Label lblSessionEndMethod;
        private System.Windows.Forms.Label lblSchedEndMethod;
        private System.Windows.Forms.Label lblSchedEndTime;
        private System.Windows.Forms.TextBox txtWakeTime;
        private System.Windows.Forms.Label lblSchedStartTime;
        private System.Windows.Forms.TextBox txtStopMethod;
        private System.Windows.Forms.TextBox txtStopTime;
        private System.Windows.Forms.Button btnClearEnd;
        private System.Windows.Forms.Button btnSetEnd;
        private System.Windows.Forms.Button btnClearStart;
        private System.Windows.Forms.Button btnSetStart;
        private System.Windows.Forms.ComboBox cmbBoxSessionEnd;
        private System.Windows.Forms.CheckBox chkBoxReboot;
        private System.Windows.Forms.GroupBox boxNetwork;
        private System.Windows.Forms.TextBox clientTextBoxMAC;
        private System.Windows.Forms.TextBox clientTextBoxIP;
        private System.Windows.Forms.Label lblIP;
        private System.Windows.Forms.Label lblMAC;
        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.PictureBox pictureBox2;
        private System.Windows.Forms.PictureBox pictureBox3;
        private System.Windows.Forms.TrackBar trkBarEfficiency;
        private System.Windows.Forms.GroupBox boxFinalDetails;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Button buttonClose;
        private System.Windows.Forms.TextBox textBoxConnectStatus;
        private System.Windows.Forms.TextBox serverTextBoxPort;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.TextBox serverTextBoxIP;
        private System.Windows.Forms.Button buttonUpdateIP;
        private System.Windows.Forms.Button buttonUpdatePort;
        private System.Windows.Forms.Label lblTrackBar1;
        private System.Windows.Forms.GroupBox boxTimeOut;
        private System.Windows.Forms.TextBox txtEfficiency;
        private System.Windows.Forms.Label lblTrackBar2;
        private System.Windows.Forms.Label lblHibernateTime;
        private System.Windows.Forms.TextBox txtIdleTime;
        private System.Windows.Forms.GroupBox boxSavings;
        private System.Windows.Forms.Label lblIdeTime;
        private System.Windows.Forms.Button btnCarbonSaved;
        private System.Windows.Forms.Button btnEnergySaved;
        private System.Windows.Forms.Button btnMoneySaved;
        private System.Windows.Forms.Label lblStaticEnergySaved;
        private System.Windows.Forms.Label lblStaticCarbonSaved;
        private System.Windows.Forms.Label lblStaticMoneySaved;
        private System.Windows.Forms.Label lblMoneySaved;
        private System.Windows.Forms.Label lblCarbonSaved;
        private System.Windows.Forms.Label lblEnergySaved;
        private System.Windows.Forms.Button btnEnergyUsed;
        private System.Windows.Forms.PictureBox pictureBox4;
        private System.Windows.Forms.Label lblEnergyUsed;
        private System.Windows.Forms.Label lblStaticEnergyUsed;
    }
}
```

Appendix C3 – Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace PowerMan_Client1
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Appendix C4 – DateTimeState.cs

```
using System;

namespace PowerMan_Client1
{
    public class DateTimeState
    {
        public DateTimeState()
        {
            endDateTime = new DateTime();
            startDateTime = new DateTime();
            SetWakeOccurred(false);
            SetEndTimeSet(false);
            SetStartTimeSet(false);
            SetPostWakeReboot(false);
            SetShutdownMethod("-- Not set --");
        }

        //Other methods
        public void SetEndDateTime(DateTime dt)
        {
            endDateTime = dt;
        }
        public void SetStartDateTime(DateTime dt)
        {
            startDateTime = dt;
        }
        public void SetWakeOccurred(bool wakeOccurred)
        {
            this.wakeOccurred = wakeOccurred;
        }
        public void SetEndTimeSet(bool endTimeSet)
        {
            this.endTimeSet = endTimeSet;
        }
        public void SetStartTimeSet(bool startTimeSet)
        {
            this.startTimeSet = startTimeSet;
        }
        public void SetPostWakeReboot(bool postWakeReboot)
        {
            this.postWakeReboot = postWakeReboot;
        }
        public void SetShutdownMethod(string shutdownType)
        {
            this.shutdownType = shutdownType;
        }
        public DateTime GetEndDateTime()
        {
            return endDateTime;
        }
        public DateTime GetStartDateTime()
        {
            return startDateTime;
        }
        public bool GetWakeOccurred()
        {
            return wakeOccurred;
        }
        public bool GetEndTimeSet()
        {
            return endTimeSet;
        }
        public bool GetStartTimeSet()
        {
            return startTimeSet;
        }
        public string GetShutdownMethod()
        {
            return shutdownType;
        }
        public bool GetPostWakeReboot()
        {
            return postWakeReboot;
        }

        // variables/fields
        private bool wakeOccurred;
        private bool endTimeSet;
        private bool startTimeSet;
        private bool postWakeReboot;
        private DateTime endDateTime;
        private DateTime startDateTime;
        private string shutdownType;
    }
}
```

Appendix C5 – ServerPacket.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net.Sockets;

namespace PowerMan_Client1
{
    class ServerPacket
    {
        //constructors
        public ServerPacket()
        {
        }

        //Other methods
        public Socket GetSocket()
        {
            return aCurrentSocket;
        }

        public void SetSocket(Socket aNewSocket)
        {
            aCurrentSocket = aNewSocket;
        }

        // variables/fields
        private Socket aCurrentSocket;
        public List<byte> TransmissionBuffer = new List<byte>();
        public byte[] buffer = new byte[128];
    }
}
```

Appendix D – PowerMan Server Code

The PowerMan Server code is included in the following appendices:

- Appendix D1 – Form1.cs
- Appendix D2 – Form1.Designer.cs
- Appendix D3 – Program.cs
- Appendix D4 – ClientPacket.cs
- Appendix D5 – ServerClient.cs

Appendix D1 – Form1.cs

```
using Microsoft.Win32;
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Management;
using System.Net;
using System.Net.Sockets;
using System.Runtime.InteropServices;
using System.Threading;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Web.UI.WebControls;
using System.Text.RegularExpressions;
using ClassLibrary1;

namespace PowerMan_Server1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            timer1.Enabled = true;
        }

        // declare windows API functions
        [DllImport("iphlpapi.dll", ExactSpelling = true)]
        private static extern int SendARP(int DestIP, int SrcIP, [Out] byte[]
pMacAddr, ref int PhyAddrLen);

        private void Form1_Load(object sender, EventArgs e)
        {
            // create a select query using 'MacAddress.IPAddress' param
            ObjectQuery query2 = new ObjectQuery("Select MacAddress,IPAddress
from Win32_NetworkAdapterConfiguration where IPEnabled=TRUE");
            // create a new object searcher and pass to the select query
            ManagementObjectSearcher searcher2 = new
ManagementObjectSearcher(query2);

            foreach (ManagementObject mgtObject2 in searcher2.Get())
            {
                serverMACString = mgtObject2["MacAddress"].ToString();
                serverIPString =
((System.Array)(mgtObject2["IPAddress"])).GetValue(0).ToString();
            }

            // convert MAC to standard format
            formattedServerMACString = FormatMAC(serverMACString);

            // update text boxes
            serverTextBoxMAC.Text = formattedServerMACString;
            serverTextBoxIP.Text = serverIPString;
            serverTextBoxPort.Text = serverPort.ToString();
            textBoxServerStatus.Text = "--Server not ready to host!--";
            textBoxServerStatus.BackColor = System.Drawing.Color.Red;
            // change formatted mac for file load
            string temp = formattedServerMACString.Replace(":", "");
            // attempt to load file

            ReadWakeList("c:\\server" + temp + ".txt");

            // set server port buttons to false
            buttonUpdatePort.Enabled = false;

            // set session start time;
            startTime = DateTime.Now;
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            txtDate.Text = DateTime.Now.ToLongDateString();
            txtTime.Text = DateTime.Now.ToLongTimeString();

            TimeSpan ts1 = DateTime.Now - startTime;
            //establish listenning after 10 seconds into commencement
            if (ts1.TotalSeconds > 10 && !listenTryFlag)
            {
                // set flag to ensure connection is only tried once
                listenTryFlag = true;
                //call connect function
                ListenForClients();
                textBoxServerStatus.Text = "--Hosting clients!--";
                //textBoxServerStatus.Font =
                textBoxServerStatus.BackColor = System.Drawing.Color.Lime;
                // set port so that it can not be changed
                serverTextBoxPort.Text = serverPort.ToString();
                serverTextBoxPort.BackColor =
System.Drawing.Color.LightSteelBlue;
                serverTextBoxPort.ReadOnly = true;
                buttonUpdatePort.Enabled = false;
            }
            else if (!listenTryFlag)
            {
                textBoxServerStatus.Text = String.Format("Hosting clients in {0}
secs", (10 -  ts1.Seconds));
                // swap every second
                if (ts1.Seconds % 2 == 0)
                {
                    textBoxServerStatus.BackColor = System.Drawing.Color.Red;
                }
                else
                {
                    textBoxServerStatus.BackColor = System.Drawing.Color.Orange;
                }
            }

            //update number of connected sockets
            UpdateClientSockets();

            //update number of connect clients
            textBoxConnectedClients.Text = CountClients().ToString();

            // Generate a manual wake event
            if (manualWakeInProgress)
            {
                if (StoredManualWakeDelay == 0)
                {
                    for (int i = 0; i < demandWakeList.Count; i++)
                    {
                        WakeUpClient(demandWakeList[i].ToString());
                    }
                    // indicate wake has finished
                    manualWakeInProgress = false;
                    // clear list
                    demandWakeList.Clear();
                }
                else
                {
                    if (DateTime.Now >=
manualWakeProgressiveTime.AddSeconds(StoredManualWakeDelay))
                    {
                        // update base to time to reflect delay interval
exceeded
                        manualWakeProgressiveTime =
manualWakeProgressiveTime.AddSeconds(StoredManualWakeDelay);
                        if (demandWakeList.Count != 0)
                        {
                            WakeUpClient(demandWakeList[0].ToString());
                            demandWakeList.RemoveAt(0);
                        }
                        else
                        {
                            // indicate wake has finished
                            manualWakeInProgress = false;
                            // reset delay value
                            StoredManualWakeDelay = 0;
                        }
                    }
                }
            }

            // Evaluate for a scheduled wake event has been set
            if (scheduledWakeSet)
            {
                TimeSpan tsp1 = new TimeSpan();
                tsp1 = DateTime.Now - scheduledWakeLastTime;
                // ensure scheduled wake only occurs once a day
                if ((tsp1.Days >= 1) && (DateTime.Now.TimeOfDay >=
scheduledWakeNextTime.TimeOfDay))
                {
                    WakeUp();
                    scheduledWakeSet = false;
                }
            }

            // Generate a scheduled wake event
            if (scheduledWakeInProgress)
            {
                if (StoredScheduledWakeDelay == 0)
                {
                    for (int i = 0; i < scheduledWakeList.Count; i++)
                    {
                        WakeUpClient(scheduledWakeList[i].ToString());
                    }
                    // indicate wake has finished
                    scheduledWakeInProgress = false;
                    // allow schedule to be re used
                    scheduledWakeSet = true;
                    // clear list
                    scheduledWakeList.Clear();
                    // update last and next dates
                    scheduledWakeLastTime = scheduledWakeNextTime;
                    scheduledWakeNextTime = scheduledWakeNextTime.AddDays(1);
                }
                else
                {
                    if (DateTime.Now >=
scheduledWakeProgressiveTime.AddSeconds(StoredScheduledWakeDelay))
                    {
                        // update base to time to reflect delay interval
exceeded
                        scheduledWakeProgressiveTime =
scheduledWakeProgressiveTime.AddSeconds(StoredScheduledWakeDelay);
                        if (scheduledWakeList.Count != 0)
                        {
                            WakeUpClient(scheduledWakeList[0].ToString());
                            scheduledWakeList.RemoveAt(0);
                        }
                        else
                        {
                            // indicate wake has finished
                            scheduledWakeInProgress = false;
                            // allow schedule to be re used
                            scheduledWakeSet = true;
                            // update last and next dates
                            scheduledWakeLastTime = scheduledWakeNextTime;
                            scheduledWakeNextTime =
scheduledWakeNextTime.AddDays(1);
                        }
                    }
                }
            }
        }

//********************************************************************************
**
//**************************** Event Driven methods
******************************
//********************************************************************************
**

        private void buttonClose_Click(object sender, EventArgs e)
        {
            CloseSockets();
            // change formatted mac for file load
            string temp = formattedServerMACString.Replace(":", "");
            // attempt to load file
            WriteWakeList("c:\\server" + temp + ".txt");
            Close();
        }

        private void buttonRestart_Click(object sender, EventArgs e)
        {
            if (clientListBoxConnected.SelectedIndex != -1)
            {
                SendServerRequest("Restart");
            }
            else
            {
                MessageBox.Show("You must select a Connected Client!");
            }
        }

        private void buttonHibernate_Click(object sender, EventArgs e)
        {
            if (clientListBoxConnected.SelectedIndex != -1)
            {
                SendServerRequest("Hibernate");
            }
            else
            {
                MessageBox.Show("You must select a Connected Client!");
            }
        }

        private void buttonShutdown_Click(object sender, EventArgs e)
        {
            if (clientListBoxConnected.SelectedIndex != -1)
            {
                SendServerRequest("Shutdown");
            }
            else
            {
                MessageBox.Show("You must select a Connected Client!");
            }
        }

        private void buttonRefresh_Click(object sender, EventArgs e)
        {
            if (clientListBoxConnected.SelectedIndex != -1)
            {
                SendServerRequest("Refresh");
            }
            else
            {
                MessageBox.Show("You must select a Connected Client!");
            }
        }

        private void buttonAppClose_Click(object sender, EventArgs e)
        {
            if (clientListBoxConnected.SelectedIndex != -1)
            {
                SendServerRequest("CloseApplication");
            }
            else
            {
                MessageBox.Show("You must select a Connected Client!");
            }
        }

        private void clientListBoxMAC_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if ((clientListBoxMAC.SelectedIndex != -1) &&
(!chkBoxWakeList.Checked))
            {
                buttonWakeClient.Enabled = true;
                buttonRemoveClients.Enabled = true;
                btnUpdateScheduleList.Enabled = true;
                chkBoxWakeList.Enabled = false;
            }
            else
            {
                buttonWakeClient.Enabled = false;
                buttonRemoveClients.Enabled = false;
                btnUpdateScheduleList.Enabled = false;
                chkBoxWakeList.Enabled = true;
            }
        }
        private void clientListBoxConnected_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if (clientListBoxConnected.SelectedIndex != -1)
            {
                ServerClient aServerClient =
(ServerClient)clientListBoxConnected.SelectedItem;
                if (aServerClient.GetClient() != null)
                {
                    if (aServerClient.GetClient().GetEfficiencyValue() == 0)
                    {
                        txtEfficiency.Font = new Font(txtEfficiency.Font,
FontStyle.Bold);
                        txtEfficiency.ForeColor =
System.Drawing.SystemColors.WindowText;
                        txtEfficiency.Text = "-- Never --";
                        txtEfficiency.BackColor =
System.Drawing.SystemColors.ScrollBar;
                    }
                    else
                    {
                        txtEfficiency.Font = new Font(txtEfficiency.Font,
FontStyle.Regular);
                        txtEfficiency.ForeColor = System.Drawing.Color.Green;
                        txtEfficiency.BackColor =
System.Drawing.SystemColors.Window;
                        string temp =
aServerClient.GetClient().GetEfficiencyValue().ToString();
                        txtEfficiency.Text = "after " + temp + " minute(s)";
                    }

                    // publish saving figures
                    string temp1 = string.Format("{0:0.00}",
aServerClient.GetClient().GetEnergyUsed());
                    string temp2 = string.Format("{0:0.00}",
aServerClient.GetClient().GetEnergySaved());
                    string temp3 = string.Format("{0:0.00}",
aServerClient.GetClient().GetCarbonSaved());
                    string temp4 = string.Format("{0:0.00}",
aServerClient.GetClient().GetMoneySaved());
                    lblEnergyUsed.Text = temp1 + " kWh";
                    lblEnergySaved.Text = temp2 + " kWh";
                    lblCarbonSaved.Text = temp3 + " kg";
                    lblMoneySaved.Text = "$" + temp4;

                }
                UpdateClientControls(true);
            }
            else
            {
                txtEfficiency.Font = new Font(txtWakeDelay.Font,
FontStyle.Bold);
                txtEfficiency.ForeColor =
System.Drawing.SystemColors.WindowText;
                txtEfficiency.Text = "No Client Selected!";
                txtEfficiency.BackColor = System.Drawing.SystemColors.ScrollBar;
                lblEnergySaved.Text = "No Client";
                lblCarbonSaved.Text = "No Client";
                lblMoneySaved.Text = "No Client";
                lblEnergyUsed.Text = "No Client";
                UpdateClientControls(false);
            }
        }
        private void buttonWakeClient_Click(object sender, EventArgs e)
        {
            if (manualWakeInProgress)
            {
                MessageBox.Show("Wake procedure already underway!");
                // deselect all items
                for (int i = 0; i < clientListBoxMAC.Items.Count; i++)
                {
                    clientListBoxMAC.SetSelected(i, false);
                }
                buttonWakeClient.Enabled = false;
                buttonRemoveClients.Enabled = false;
                btnUpdateScheduleList.Enabled = false;
                chkBoxWakeList.Checked = false;
                return;
            }

            if (clientListBoxMAC.SelectedIndex != -1)
            {
                for (int i = 0; i < clientListBoxMAC.SelectedItems.Count; i++)
                {

demandWakeList.Add(clientListBoxMAC.SelectedItems[i].ToString());
                }
                // deselect all items
                for (int i = 0; i < clientListBoxMAC.Items.Count; i++)
                {
                    clientListBoxMAC.SetSelected(i, false);
                }
                // set this wake events delay value
                StoredManualWakeDelay = wakeDelayValue;
                // set wake in progress to true
                manualWakeInProgress = true;
                // set start time
                manualWakeProgressiveTime = DateTime.Now;
                // update buttons
                buttonWakeClient.Enabled = false;
                buttonRemoveClients.Enabled = false;
                btnUpdateScheduleList.Enabled = false;
                chkBoxWakeList.Checked = false;
            }
            else
            {
                MessageBox.Show("You must select a PC ID!");
            }
            buttonWakeClient.Enabled = false;
            buttonRemoveClients.Enabled = false;
            btnUpdateScheduleList.Enabled = false;
            chkBoxWakeList.Checked = false;
        }

        private void btnUpdateScheduleList_Click(object sender, EventArgs e)
        {
            if (clientListBoxMAC.SelectedIndex != -1)
            {
                for (int i = 0; i < clientListBoxMAC.SelectedItems.Count; i++)
                {
                    // check that entrys are not duplicated
                    int myIndex =
listBoxSchedule.FindStringExact(clientListBoxMAC.SelectedItems[i].ToString());
                    if (myIndex == -1)
                    {

listBoxSchedule.Items.Add(clientListBoxMAC.SelectedItems[i].ToString());
```

```
                    }
                }
                // deselect all items
                for (int i = 0; i < clientListBoxMAC.Items.Count; i++)
                {
                    clientListBoxMAC.SetSelected(i, false);
                }
                buttonWakeClient.Enabled = false;
                buttonRemoveClients.Enabled = false;
                btnUpdateScheduleList.Enabled = false;
                chkBoxWakeList.Checked = false;
            }
            else
            {
                MessageBox.Show("You must select a PC ID!");
            }
            buttonWakeClient.Enabled = false;
            buttonRemoveClients.Enabled = false;
            btnUpdateScheduleList.Enabled = false;
            chkBoxWakeList.Checked = false;
        }

        private void btnScheduleSet_Click(object sender, EventArgs e)
        {
            if (listBoxSchedule.Items.Count == 0)
            {
                MessageBox.Show("You must create a schedule list!");
            }
            else
            {
                scheduledWakeNextTime = dTPickerWake.Value;
                scheduledWakeLastTime = scheduledWakeNextTime.AddDays(-1);

                txtWakeTime.Text = scheduledWakeNextTime.ToLongTimeString();
                txtWakeTime.BackColor = System.Drawing.Color.Lime;
                scheduledWakeSet = true;

                StoredScheduledWakeDelay = wakeDelayValue;

                if (StoredScheduledWakeDelay == 0)
                {
                    txtDailyWakeDelay.Font = new Font(txtWakeDelay.Font,
FontStyle.Bold);
                    txtDailyWakeDelay.ForeColor =
System.Drawing.SystemColors.WindowText;
                    txtDailyWakeDelay.Text = "-- No Delay --";
                    txtDailyWakeDelay.BackColor =
System.Drawing.SystemColors.ScrollBar;
                }
                else
                {
                    txtDailyWakeDelay.Font = new Font(txtWakeDelay.Font,
FontStyle.Regular);
                    txtDailyWakeDelay.BackColor =
System.Drawing.SystemColors.Window;
                    txtDailyWakeDelay.Text = wakeDelayValue.ToString() + "
second(s)";
                }
            }
        }

        private void btnScheduleClear_Click(object sender, EventArgs e)
        {
            scheduledWakeNextTime = DateTime.Now;
            scheduledWakeLastTime = DateTime.Now;

            txtWakeTime.Text = "-- Not set --";
            txtWakeTime.BackColor = System.Drawing.SystemColors.ScrollBar;

            txtDailyWakeDelay.Font = new Font(txtWakeDelay.Font,
FontStyle.Bold);
            txtDailyWakeDelay.ForeColor =
System.Drawing.SystemColors.WindowText;
            txtDailyWakeDelay.Text = "-- No Delay --";
            txtDailyWakeDelay.BackColor = System.Drawing.SystemColors.ScrollBar;

            listBoxSchedule.Items.Clear();
            scheduledWakeSet = false;
            scheduledWakeInProgress = false;
        }

        private void newclientTextBox_TextChanged(object sender, EventArgs e)
        {
            buttonUpdateList.Enabled = true;
        }

        private void buttonUpdateList_Click(object sender, EventArgs e)
        {
            // See if we have a valid MAC address
            string temp = IsValidMAC(newclientTextBox.Text.ToString());
            if (temp == "")
            {
                MessageBox.Show("Must enter valid MAC Address in
**:**:**:**:**:** format!");
                newclientTextBox.Text = "Enter Client MAC Address";
                buttonUpdateList.Enabled = false;
                return;
            }
            else
            {
                int myIndex = clientListBoxMAC.FindStringExact(temp);
                if (myIndex != -1)
                {
                    MessageBox.Show("Client ID already added!");
                    newclientTextBox.Text = "Enter Client MAC Address";
                    buttonUpdateList.Enabled = false;
                    return;
                }
                else
                {
                    clientListBoxMAC.Items.Add(temp);
                    newclientTextBox.Text = "Enter Client MAC Address";
                    buttonUpdateList.Enabled = false;
                    return;
                }
            }
        }

        private void buttonRemoveClients_Click(object sender, EventArgs e)
        {
            DialogResult clientRemove = MessageBox.Show("Are you sure you want
to delete client IDs?", "Delete Clients",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question);

            if (clientListBoxMAC.SelectedIndex != -1 && clientRemove ==
DialogResult.Yes)
            {
                clientListBoxMAC.Items.Remove(clientListBoxMAC.SelectedItem);
            }
            buttonRemoveClients.Enabled = false;
            buttonWakeClient.Enabled = false;
            btnUpdateScheduleList.Enabled = false;
            clientListBoxMAC.SelectedIndex = -1;
        }

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (chkBoxWakeList.Checked)
            {
                for (int i = 0; i < clientListBoxMAC.Items.Count; i++)
                {
                    clientListBoxMAC.SetSelected(i, true);
                }
                clientListBoxMAC.Enabled = false;
                buttonRemoveClients.Enabled = true;
                buttonWakeClient.Enabled = true;
                btnUpdateScheduleList.Enabled = true;
            }
            if (!chkBoxWakeList.Checked)
            {
                for (int i = 0; i < clientListBoxMAC.Items.Count; i++)
                {
                    clientListBoxMAC.SetSelected(i, false);
                }
                clientListBoxMAC.Enabled = true;
                buttonRemoveClients.Enabled = false;
                buttonWakeClient.Enabled = false;
                btnUpdateScheduleList.Enabled = false;
            }
        }

        private void serverTextBoxPort_TextChanged(object sender, EventArgs e)
        {
            buttonUpdatePort.Enabled = true;
        }

        private void buttonUpdatePort_Click(object sender, EventArgs e)
        {
            bool temp = IsValidPort(int.Parse(serverTextBoxPort.Text));

            if (!temp)
            {
                MessageBox.Show("Must enter port in range 50000-65000!");
                serverTextBoxPort.Text = serverPort.ToString();
                buttonUpdatePort.Enabled = false;
                return;
            }
            else
            {
                serverPort = int.Parse(serverTextBoxPort.Text);
                buttonUpdatePort.Enabled = false;
                return;
            }
        }

        private void trkBarDelay_Scroll(object sender, EventArgs e)
        {
            wakeDelayValue = wakeDelayScale[trkBarDelay.Value];
            if (trkBarDelay.Value == 0)
            {
                txtWakeDelay.Font = new Font(txtWakeDelay.Font, FontStyle.Bold);
                txtWakeDelay.ForeColor = System.Drawing.SystemColors.WindowText;
                txtWakeDelay.Text = "-- No Delay --";
                txtWakeDelay.BackColor = System.Drawing.SystemColors.ScrollBar;
            }
            else
            {
                txtWakeDelay.Font = new Font(txtWakeDelay.Font,
FontStyle.Regular);
                txtWakeDelay.BackColor = System.Drawing.SystemColors.Window;
                txtWakeDelay.Text = wakeDelayValue.ToString() + " second(s)";
            }
        }

//*******************************************************************************
*
//**************************** Other methods
************************************
//*******************************************************************************
*

        // method to test valid tcp port between 50000-65000
        public bool IsValidPort(int portNumber)
        {
            if ((portNumber >= 50000) && (portNumber <= 65000))
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        // method to test enterred MAC is in "**:**:**:**:**:**" format
        public string IsValidMAC(string macAddress)
        {
            // test for 6 bytes seperated by ':'
            Regex expr1 = new Regex("([0-9a-fA-F][0-9a-fA-F]:){5}([0-9a-fA-F][0-
9a-fA-F])", RegexOptions.IgnoreCase);
            Match match1 = expr1.Match(macAddress);
            string result = match1.Groups[0].Value;
            // if length is 17 then valid MAC
            if (result.Length == 17)
            {
                return result.ToUpper();
            }
            else
            {
                return "";
            }
        }

        // Method uses an Address Resolution Protocol request to get a MAC
address
        // from a given IP address.
        public string IpToMacAddress(IPAddress ipAddress)
        {
            byte[] mac = new byte[6];
            int len = mac.Length;
            int res = SendARP((int)ipAddress.Address, 0, mac, ref len);
            if (res != 0)
            {
                MessageBox.Show("Error " + res + " looking up " +
ipAddress.ToString());
                return "none found";
            }
            else
            {
                string temp;
                StringBuilder hex = new StringBuilder(mac.Length * 2);
                foreach (byte b in mac)
                {
                    if (hex.Length <= 12)
                    {
                        hex.AppendFormat("{0:x2}:", b);
                    }
                    else
                    {
                        hex.AppendFormat("{0:x2}", b);
                    }
                }
                temp = hex.ToString();
                return temp;
            }
        }

        //method recognies 3 formats of MAC address and returns a consistent
format
        // ':' seperated, '-' seperated, and no seperation between bytes
        private string FormatMAC(string aMACString)
        {
            //get string to common format before
            string temp1 = aMACString.Replace(":", "");
            string temp2 = temp1.Replace("-", "");
            string temp3 = temp2.ToUpper();

            // temp2 should now be length 12 chars
            StringBuilder final = new StringBuilder(temp3.Length + 5);
            for (int i = 1; i < (temp3.Length + 1); i++)
            {
                final.Append(temp3[i - 1]);
                if ((i % 2 == 0) && (i != temp3.Length))
                {
                    final.Append(':');
                }
            }
            return final.ToString();
        }

        // Method creates a beginning primary socket and then after listening
        // for clients begins to accept new client connections.
        public void ListenForClients()
        {
            try
            {
                // Create the listening socket...
                serverMainSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
                IPEndPoint ipLocal = new IPEndPoint(IPAddress.Any, serverPort);
                // Bind to local IP Address...
                serverMainSocket.Bind(ipLocal);
                // Start listening...
                serverMainSocket.Listen(4);
                // Create the call back for any client connections...
                serverMainSocket.BeginAccept(new AsyncCallback(OnClientConnect),
null);
            }
            catch (SocketException se)
            {
                MessageBox.Show(se.Message);
            }
        }

        // This is the call back function, which will be invoked when a client
is connected
        public void OnClientConnect(IAsyncResult asyn)
        {
            try
            {
                // create new client object
                ServerClient aClient = new ServerClient();
                // Call EndAccept() to return reference to new Socket object
                aClient.SetSocket(serverMainSocket.EndAccept(asyn));
                // get IP address of newly connected client
                IPEndPoint newClientEndPoint =
(IPEndPoint)aClient.GetSocket().RemoteEndPoint;
                // get MAC address for given client IP address and format
                string newClientID =
FormatMAC(IpToMacAddress(newClientEndPoint.Address));
                // add client ID to wake list if not already on there
                int myIndex = clientListBoxMAC.FindStringExact(newClientID);
                if (myIndex == -1)
                {
                    clientListBoxMAC.Items.Add(newClientID);
                }
                // update clients IP field
                aClient.SetClientIP(newClientEndPoint.Address.ToString());
                // update clients ID field
                aClient.SetClientID(newClientID);
                // add object with ID to connected list
                clientListBoxConnected.Items.Add(aClient);
                // Let worker Socket do further processing for the just
connected client
                WaitForData(aClient.GetSocket());
                // display a connection message for new client
                String str = String.Format("{0} online", newClientID);
                textBoxStatusMsg.Text = str;
                // main Socket now free, it can return go back and wait for
other clients connecting
                serverMainSocket.BeginAccept(new AsyncCallback(OnClientConnect),
null);
            }
            catch (SocketException se)
            {
                MessageBox.Show(se.Message);
            }
        }

        // method to send server object details
        private void SendServerRequest(string serverRequest)
        {
            if (clientListBoxConnected.SelectedIndex != -1)
            {
                try
                {
                    // convert string to byte array
                    byte[] serverData = ObjectToByteArray(serverRequest);
                    ServerClient aClient =
(ServerClient)clientListBoxConnected.SelectedItem;
                    if (aClient.GetSocket() != null)
                    {
                        if (aClient.GetSocket().Connected)
                        {
                            aClient.GetSocket().Send(serverData);
                        }
                    }
                }
                catch (SocketException se)
                {
                    MessageBox.Show(se.Message);
                }
            }
        }

        // This method converts a server object to a byte array so that it
        // can be transmitted over the socket.
        public byte[] ObjectToByteArray(object clientObject)
        {
            try
            {
                // create new memory stream
                System.IO.MemoryStream byteMemoryStream = new
System.IO.MemoryStream();
                // create new BinaryFormatter
                System.Runtime.Serialization.Formatters.Binary.BinaryFormatter
byteBinaryFormatter
                            = new
System.Runtime.Serialization.Formatters.Binary.BinaryFormatter();
                // Serializes an object, or graph of connected objects, to the
given stream.
                byteBinaryFormatter.Serialize(byteMemoryStream, clientObject);
                // convert stream to byte array and return
                return byteMemoryStream.ToArray();
            }
            catch (Exception ex)
            {
                // Error
                Console.WriteLine("Exception caught in process: {0}",
ex.ToString());
            }
            // Error occured, return null
            return null;
        }

        // method to terminate all open sockets
        public void CloseSockets()
        {
            if (serverMainSocket != null)
            {
                serverMainSocket.Close();
            }

            for (int i = 0; i < clientListBoxConnected.Items.Count; i++)
            {
                ServerClient aClient =
(ServerClient)clientListBoxConnected.Items[i];
                if (aClient.GetSocket() != null)
                {
                    aClient.GetSocket().Close();
                    aClient.SetSocket(null);
                }
            }
        }

        // This method tests all sockets and then upodates the lists that
containreferences
        // the sockets
        public void UpdateClientSockets()
        {
            //create list to store clients for removal
            List<int> clientList = new List<int>();
            for (int i = 0; i < clientListBoxConnected.Items.Count; i++)
            {
                ServerClient aClient =
(ServerClient)clientListBoxConnected.Items[i];
                if (aClient.GetSocket() != null)
                {
                    if (!SocketTest(aClient.GetSocket()))
                    {
                        clientList.Add(i);
                    }
                }
            }
            // remove clients with no connection
            foreach (int myIndex in clientList)
            {
                ServerClient aClient =
(ServerClient)clientListBoxConnected.Items[myIndex];
                String str = String.Format("{0} offline",
aClient.GetClientID());
                textBoxStatusMsg.Text = str;
                clientListBoxConnected.Items.RemoveAt(myIndex);
            }
        }

        // method to poll socket to determine if connected NOW!
        public bool SocketTest(Socket aSocket)
        {
            try
            {
                return !(aSocket.Poll(1, SelectMode.SelectRead) &&
aSocket.Available == 0);
            }
            catch (SocketException)
            {
                return false;
            }
        }

        // Start waiting for data from the client
        public void WaitForData(Socket soc)
        {
            try
            {
                if (serverpfnCallBack == null)
                {
                    // Call back function to be invoked when write activity by
the connected client
                    serverpfnCallBack = new AsyncCallback(OnDataReceived);
                }
                ClientPacket theServerClient = new ClientPacket();
                theServerClient.SetSocket(soc);
                // Start receiving any data written by the connected client
asynchronously
                soc.BeginReceive(theServerClient.buffer, 0,
theServerClient.buffer.Length, SocketFlags.None, serverpfnCallBack,
theServerClient);
            }
            catch (SocketException se)
            {
                MessageBox.Show(se.Message);
            }

        }

        // Call back function which is invoked when the socket detects any
writing on the stream
        public void OnDataReceived(IAsyncResult asyn)
        {
            try
            {
                ClientPacket socketData = (ClientPacket)asyn.AsyncState;
                // get length of received stream
                int read = socketData.GetSocket().EndReceive(asyn);
                if (read > 0)
                {
                    for (int i = 0; i < read; i++)
                    {
                        socketData.TransmissionBuffer.Add(socketData.buffer[i]);
                    }
                    // add something here
                    byte[] newBuffer = socketData.TransmissionBuffer.ToArray();
                    Client myClient = (Client)ByteArrayToObject(newBuffer);
                    if (myClient != null)
                    {
                        for (int i = 0; i < clientListBoxConnected.Items.Count;
i++)
                        {
                            ServerClient aServerClient =
(ServerClient)clientListBoxConnected.Items[i];
                            if (myClient.GetClientID() ==
aServerClient.GetClientID())
                            {
                                aServerClient.SetClient(myClient);
                                clientListBoxConnected.Items[i] = aServerClient;
                            }
                        }
                    }
                 }
                WaitForData(socketData.GetSocket());
            }
            catch (SocketException se)
            {
                MessageBox.Show(se.Message);
            }
        }

        // Method converts the server byte array to an object after it has been
        // received over the socket.
        public object ByteArrayToObject(byte[] serverByteArray)
        {
              try
              {
                  // convert byte array to memory stream
                  System.IO.MemoryStream myMemoryStream = new
System.IO.MemoryStream(serverByteArray);
                  // create new BinaryFormatter
                  System.Runtime.Serialization.Formatters.Binary.BinaryFormatter
myBinaryFormatter
                          = new
System.Runtime.Serialization.Formatters.Binary.BinaryFormatter();
                // set memory stream position to starting point
                myMemoryStream.Position = 0;
                // Deserializes a stream into an object graph and return as a
object.
```

```
                return myBinaryFormatter.Deserialize(myMemoryStream);
            }
              catch (Exception ex)
            {
                  // Error
                  Console.WriteLine("Exception caught in process: {0}",
ex.ToString());
            }
              // Error occured, return null
              return null;
        }

        // Uses list of selected clients and delay to wake clients
        public void WakeUp()
        {
            for (int i = 0; i < listBoxSchedule.Items.Count; i++)
            {
                scheduledWakeList.Add(listBoxSchedule.Items[i].ToString());
            }
            // set wake in progress to true
            scheduledWakeInProgress = true;
            // set start time
            scheduledWakeProgressiveTime = DateTime.Now;
        }

        // Sends a Wake On Lan packet to the specified MAC address using UDP.
        public void WakeUpClient(string clientID)
        {
            // get hex byte array of MAC string
            byte[] tempBytes = MACStringToBytes(clientID);
            // WOL packet is sent over UDP 255.255.255.0 on designated 40000.
            UdpClient client = new UdpClient();
            client.Connect(IPAddress.Broadcast, 40000);
            // WOL packet contains a 6-bytes trailer and 16 times a 6-bytes
sequence containing the MAC address.
            byte[] packet = new byte[17 * 6];
            // Trailer of 6 times 0xFF.
            for (int i = 0; i < 6; i++)
            {
                packet[i] = 0xFF;
            }
            // Body of magic packet contains 16 times the MAC address.
            for (int i = 1; i <= 16; i++)
            {
                for (int j = 0; j < 6; j++)
                {
                    packet[i * 6 + j] = tempBytes[j];
                }
            }
            // Send a Wake on LAN packed by UDP.
            client.Send(packet, packet.Length);
        }

        // Convert a MAC ID string to a byte array
        public byte[] MACStringToBytes(string clientID)
        {
            string temp = clientID.Replace(":", "");
            int NumberChars = temp.Length;
            byte[] macBytes = new byte[NumberChars / 2];
            for (int i = 0; i < NumberChars; i += 2)
            {
                macBytes[i / 2] = Convert.ToByte(temp.Substring(i, 2), 16);
            }
            return macBytes;
        }

        // method to count and return the actual number of connected clients
        public int CountClients()
        {
            return clientListBoxConnected.Items.Count;
        }

        // method to write wake list to file
        private void WriteWakeList(string fileName)
        {
            StreamWriter writer;
            try
            {
                writer = new StreamWriter(fileName);
                for (int i = 0; i < clientListBoxMAC.Items.Count; i++)
                {

writer.WriteLine(Convert.ToString(clientListBoxMAC.Items[i]));
                }
                writer.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(Convert.ToString(ex.Message));
                return;
            }
        }

        // method to read wake list from file
        private void ReadWakeList(string fileName)
        {
            StreamReader reader;
            try
            {
                reader = new StreamReader(fileName);
                while (reader.Peek() >= 0)
                {
                    clientListBoxMAC.Items.Add(reader.ReadLine());
                }
                reader.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(Convert.ToString(ex.Message));
                return;
            }
        }

        // method to update client buttons/controls
        public void UpdateClientControls(bool result)
        {
            buttonAppClose.Enabled = result;
            buttonShutdown.Enabled = result;
            buttonRefresh.Enabled = result;
            buttonRestart.Enabled = result;
            buttonHibernate.Enabled = result;
        }

        // variables/fields
        public AsyncCallback serverpfnCallBack;
        private Socket serverMainSocket;
        private bool listenTryFlag = false;  // Set default
        private string serverMACString;
        private string serverIPString;
        private string formattedServerMACString;
        private DateTime startTime = new DateTime();
        private int serverPort = 55000; // Set default
        private int[] wakeDelayScale = new int[6] { 0, 1, 2, 3, 4, 5 };
        private int wakeDelayValue = 0;  // Set default
        private List<string> demandWakeList = new List<string>();
        private int StoredManualWakeDelay = 0; // Set default
        private bool manualWakeInProgress = false; // set default
        private DateTime manualWakeProgressiveTime = new DateTime();
        private DateTime scheduledWakeNextTime = new DateTime();
        private DateTime scheduledWakeLastTime = new DateTime();
        private DateTime scheduledWakeProgressiveTime = new DateTime();
        private List<string> scheduledWakeList = new List<string>();
        private int StoredScheduledWakeDelay = 0; // Set default
        private bool scheduledWakeSet = false; // Set default
        private bool scheduledWakeInProgress = false;// set default
    }
}
```

Appendix D2 – Form1.Designer.cs

```
namespace PowerMan_Server1
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        private System.Windows.Forms.Timer timer1;
        private System.Windows.Forms.GroupBox boxConnectionDetails;
        private System.Windows.Forms.TextBox serverTextBoxPort;
        private System.Windows.Forms.TextBox serverTextBoxIP;
        private System.Windows.Forms.TextBox serverTextBoxMAC;
        private System.Windows.Forms.Label lblMAC;
        private System.Windows.Forms.TextBox textBoxStatusMsg;
        private System.Windows.Forms.Button buttonClose;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Button buttonWakeClient;
        private System.Windows.Forms.GroupBox boxServerDetails;
        private System.Windows.Forms.Label lblWelcome;
        private System.Windows.Forms.TextBox textBoxServerStatus;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.TextBox textBoxConnectedClients;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.Button buttonRestart;
        private System.Windows.Forms.Button buttonHibernate;
        private System.Windows.Forms.Button buttonShutdown;
        private System.Windows.Forms.Button buttonRefresh;
        private System.Windows.Forms.Button buttonAppClose;
        private System.Windows.Forms.ListBox clientListBoxMAC;
        private System.Windows.Forms.TextBox newclientTextBox;
        private System.Windows.Forms.GroupBox boxDateTime;
        private System.Windows.Forms.TextBox txtTime;
        private System.Windows.Forms.TextBox txtDate;
        private System.Windows.Forms.Label lblTime;
        private System.Windows.Forms.Label lblDate;
        private System.Windows.Forms.GroupBox boxWakeClientDetails;
        private System.Windows.Forms.Button buttonUpdateList;
        private System.Windows.Forms.Button buttonRemoveClients;
        private System.Windows.Forms.CheckBox chkBoxWakeList;
        private System.Windows.Forms.Button buttonUpdatePort;
        private System.Windows.Forms.ListBox clientListBoxConnected;
        private System.Windows.Forms.Label lblConnectedClients;
        private System.Windows.Forms.GroupBox boxClientManagement;
        private System.Windows.Forms.GroupBox boxControls;
        private System.Windows.Forms.TrackBar trkBarDelay;
        private System.Windows.Forms.Label lblEnterID;
        private System.Windows.Forms.Label lblTrackBar2;
        private System.Windows.Forms.Label lblTrackBar1;
        private System.Windows.Forms.TextBox txtWakeDelay;
        private System.Windows.Forms.Label lblWakeDelay;
        private System.Windows.Forms.Label lblClientBox1;
        private System.Windows.Forms.Label lblClientBox2;
        private System.Windows.Forms.Label lblWakeBox1;
        private System.Windows.Forms.GroupBox boxSavingsSelected;
        private System.Windows.Forms.Label lblMoneySaved;
        private System.Windows.Forms.Label lblCarbonSaved;
        private System.Windows.Forms.Label lblEnergySaved;
        private System.Windows.Forms.Label lblStaticMoneySaved;
        private System.Windows.Forms.Label lblStaticCarbonSaved;
        private System.Windows.Forms.Label lblStaticEnergySaved;
        private System.Windows.Forms.PictureBox pictureBox2;
        private System.Windows.Forms.PictureBox pictureBox3;
        private System.Windows.Forms.Button btnCarbonSaved;
        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.Button btnEnergySaved;
        private System.Windows.Forms.Button btnMoneySaved;
        private System.Windows.Forms.GroupBox boxWakeSchedule;
        private System.Windows.Forms.Button btnUpdateScheduleList;
        private System.Windows.Forms.ListBox listBoxSchedule;
        private System.Windows.Forms.Label lblWakeSchedule;
        private System.Windows.Forms.Label lblWakeStart;
        private System.Windows.Forms.Button btnScheduleSet;
        private System.Windows.Forms.Button btnScheduleClear;
        private System.Windows.Forms.Label lblDailyWakeDealy;
        private System.Windows.Forms.Label lblSchedSetTime;
        private System.Windows.Forms.TextBox txtWakeTime;
        private System.Windows.Forms.TextBox txtDailyWakeDelay;
        private System.Windows.Forms.DateTimePicker dTPickerWake;
        private System.Windows.Forms.Label lblHibernateTime;
        private System.Windows.Forms.TextBox txtEfficiency;
        private System.Windows.Forms.Label lblEnergyUsed;
        private System.Windows.Forms.Label lblStaticEnergyUsed;
        private System.Windows.Forms.PictureBox pictureBox4;
        private System.Windows.Forms.Button btnEnergyUsed;
    }
}
```

Appendix D3 – Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace PowerMan_Server1
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Appendix D4 – ClientPacket.cs

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net.Sockets;

namespace PowerMan_Server1
{

    class ClientPacket
    {
        //constructors
        public ClientPacket()
        {
        }

        //Other methods
        public Socket GetSocket()
        {
            return aCurrentSocket;
        }

        public void SetSocket(Socket aNewSocket)
        {
            aCurrentSocket = aNewSocket;
        }

        // variables/fields
        private Socket aCurrentSocket;
        public List<byte> TransmissionBuffer = new List<byte>();
        public byte[] buffer = new byte[2048];
    }
}
```

Appendix D5 – ServerClient.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net.Sockets;
using ClassLibrary1;

namespace PowerMan_Server1
{
    class ServerClient
    {
        //constructors
        public ServerClient()
        {
        }

        //Other methods
        public void SetClientID(string newClientID)
        {
            clientID = newClientID;
        }
        public string GetClientID()
        {
            return clientID;
        }
        public void SetClientIP(string newClientIP)
        {
            clientIP = newClientIP;
        }
        public string GetClientIP()
        {
            return clientIP;
        }
        public void SetSocket(Socket aNewSocket)
        {
            aCurrentSocket = aNewSocket;
        }
        public Socket GetSocket()
        {
            return aCurrentSocket;
        }
        public void SetClient(Client aNewClient)
        {
            aClient = aNewClient;
        }
        public Client GetClient()
        {
            return aClient;
        }
        // set standard format for tostring
        public override string ToString()
        {
            return "ID: " + GetClientID() + "     IP Address: " + GetClientIP();
        }
        // variables/fields
        private string clientID;
        private string clientIP;
        private Socket aCurrentSocket;
        private Client aClient;
    }
}
```

Appendix E – PowerMan Batch File Contents

PowerMan Client

```
ECHO OFF
START C:\......\PowerMan_Client1.exe
```

PowerMan Server

```
ECHO OFF
START C:\......\PowerMan_Server1.exe
```

Appendix F – PowerMan Class Library Code

The PowerMan Class Library code is included in the following appendices:

- Appendix F1 – Client.cs

Appendix F1 – Client.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net.Sockets;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;

namespace ClassLibrary1
{
    [Serializable]
    public class Client
    {
        //constructors
        public Client(string newClientID)
        {
            SetClientID(newClientID);
        }
        public Client()
        {
        }

        //Other methods
        public void SetClientID(string clientID)
        {
            this.clientID = clientID;
        }
        public string GetClientID()
        {
            return clientID;
        }
        public void SetServerIP(string serverIP)
        {
            this.serverIP = serverIP;
        }
        public string GetServerIP()
        {
            return serverIP;
        }
        public void SetServerPort(int serverPort)
        {
            this.serverPort = serverPort;
        }
        public int GetServerPort()
        {
            return serverPort;
        }
        public void SetEfficiencyValue(int efficiencyValue)
        {
            this.efficiencyValue = efficiencyValue;
        }
        public int GetEfficiencyValue()
        {
            return efficiencyValue;
        }
        public void SetEnergyUsed(double energyUsed)
        {
            this.energyUsed = energyUsed;
        }
        public double GetEnergyUsed()
        {
```

```
    return energyUsed;
}
public void SetEnergySaved(double energySaved)
{
    this.energySaved = energySaved;
}
public double GetEnergySaved()
{
    return energySaved;
}
public void SetMoneySaved(double moneySaved)
{
    this.moneySaved = moneySaved;
}
public double GetMoneySaved()
{
    return moneySaved;
}
public void SetCarbonSaved(double carbonSaved)
{
    this.carbonSaved = carbonSaved;
}
public double GetCarbonSaved()
{
    return carbonSaved;
}
public void SetTrackBarIndex(int trackBarIndex)
{
    this.trackBarIndex = trackBarIndex;
}
public int GetTrackBarIndex()
{
    return trackBarIndex;
}
public void SetInitialTime(DateTime initialTime)
{
    this.initialTime = initialTime;
}
public DateTime GetInitialTime()
{
    return initialTime;
}
public void SetInitialTimeSet(bool initialTimeSet)
{
    this.initialTimeSet = initialTimeSet;
}
public bool GetInitialTimeSet()
{
    return initialTimeSet;
}
public void SetSessionProgressiveTime(DateTime sessionProgressiveTime)
{
    this.sessionProgressiveTime = sessionProgressiveTime;
}
public DateTime GetSessionProgressiveTime()
{
    return sessionProgressiveTime;
}
public void SetSessionTimeReset(bool sessionTimeReset)
{
    this.sessionTimeReset = sessionTimeReset;
}
public bool GetSessionTimeReset()
{
```

```
                return sessionTimeReset;
            }
            public void SetHighPowerTime(TimeSpan highPowerTime)
            {
                this.highPowerTime = highPowerTime;
            }
            public TimeSpan GetHighPowerTime()
            {
                return highPowerTime;
            }
            public void SetLowPowerTime(TimeSpan lowPowerTime)
            {
                this.lowPowerTime = lowPowerTime;
            }
            public TimeSpan GetLowPowerTime()
            {
                return lowPowerTime;
            }

            // variables/fields
            private string clientID;
            private string serverIP = "192.168.15.102"; // set default
            private int serverPort = 55000; // set default
            private int efficiencyValue = 0; // set default
            private int trackBarIndex = 0;
            private double energyUsed = 0;
            private double energySaved = 0;
            private double carbonSaved = 0;
            private double moneySaved = 0;
            private DateTime initialTime = new DateTime();
            private bool initialTimeSet = false;
            private DateTime sessionProgressiveTime = new DateTime();
            private bool sessionTimeReset = false;
            private TimeSpan highPowerTime = new TimeSpan();
            private TimeSpan lowPowerTime = new TimeSpan();
        }
    }
```