Final Report

Submitted for the

BSc honours in Software Development

April 2008

Laptop Location Determination

by

Ben Best

Contents

Introduction

Many people in today's ever-more technology driven world carry portable computing devices such as laptops. These devices are often connected to multiple networks using many different network protocols and methods, dependent on what is available to them in the location that they are in at any given time. This means that the user will have to reconfigure the device to use the available network connections as they change. A better solution would be for the laptop to recognise when the availability of a network or networks changes so that it could re-adjust itself to suit the new environment. Such a system would need to be able to recognise when the availability of networks changes and what the settings should be for each location. This could be achieved by setting a profile for each network the first time the laptop connects to it. This would have to be overseen by the user as there is no way for the laptop to know what settings are required for each individual network. Alternatively the solution could simply spot when the connectivity of the device changes and ask the user for the new settings.

The aim of this project is to provide a piece of software which will automatically recognise when a device's network availability changes and to inform the user that they have changed, gained or lost a network connection. It will also attempt to recognise if the device has been connected to the network before and inform the user what network they are connected to. A secondary objective is to reconfigure the device settings for the new network(s).

Original Specification

Location Determination for Laptop Users

The aim of this project is to produce a software application that can categorize the location in which a laptop user is currently using their laptop (for example, at work at home, at an airport...). The application will use measureable information about the laptop and its connections to other devices to determine a 'signature' which the user can associate with a given location. Once the application has been configured in this way it should thereafter attempt to display the current location to the user. This will require continuous monitoring of the laptop and its environment and fuzzy or probabilistic matching of the environmental conditions against the stored profile.

Aims

- Primary Aim: To develop an algorithm to determine what location a laptop computer is in by monitoring the network(s) it is connected to, and comparing this information to stored data about locations.
  - This will be a System Tray Application which will run in the background of the windows session and call the secondary program when certain events are triggered such as a new network being discovered or a network disconnecting.
- Secondary Aim: To automatically update the laptops settings to match those required by the current location and set relevant preferences.
  - This will take the form of a windows application which will be called by the System Tray Application when certain tasks need to be performed. This program will also allow the user to set the preferences for each location.

Context

Mobile computing has increased in popularity in recent times and many people now have a network at home, a network at work and may use a network at an internet café or similar. These networks usually require individual configuration and may require certain settings to be changed each time a different network is connected to. The purpose of this project is to simplify this process by recognising when a new network is connected, comparing it to a list of previously connected networks and updating settings should they need changing. The main part of the project will involve developing a way to recognise when a network is connected that has been connected before, because things on that network may change. For example; servers may be renamed, printers may be added or removed, IP Addresses may change and other network devices may be modified in some way. The network may also connect wirelessly or though a wired link, or through a different network adapter in the same computer.

Analysis

This project is to create a software solution to automatically configure network settings for laptop users who use multiple networks. The software will allow the user to configure a network the first time they connect to it, and then will automatically select the network next time they connect to it. There will be multiple network profiles, allowing different parameters to be configured. The program may import current network settings and save them into a profile. This software is aimed at laptop users who frequently change locations and networks, for example people who use their laptops at work and also use them on their home network or regularly use wireless facilities at coffee shops, airports or other places with a wireless network.

For this project I intend to use the Windows Management Instrumentation (WMI), this allows me to request and change various parameters on the computer the software is running on and also to call methods in the program when system properties change. For this project I intend to make two pieces of software. The first (the Checker) will use the WMI to discover what is connected to the computer and make a 'signature' of the computer's network connection. This will be saved into a profile which will be editable by the user. The second (The Changer) will run when the first service detects a change in the network location of the computer. It will update the computer's settings to reflect the changes detected by the 'Checker' service. The project will be implemented in C# as it is the language I am most familiar with and allows use of the WMI easily.

Similar Software

Switchpro

SwitchPro is a similar program, but it does not attempt to automatically switch profiles, it requires the user to select the correct profile whenever they change networks. My proposed project will automatically switch profiles based on network information it collects.

MultiNetwork Manager

MultiNetwork Manager is software which is similar to what I propose, but in my trial it crashed several times and was generally unstable.

Network Configuration Manager

Network Configuration Manager is composed of the following;

- Bandwidth Monitor
- IP Manager
- Service Manager
- Process Manager
- Task Manager
- Asterisk Password Recovery
- Socket to Process Mapper
- WhoIs
- IP Information, Netstat
- Ping, Trace Route

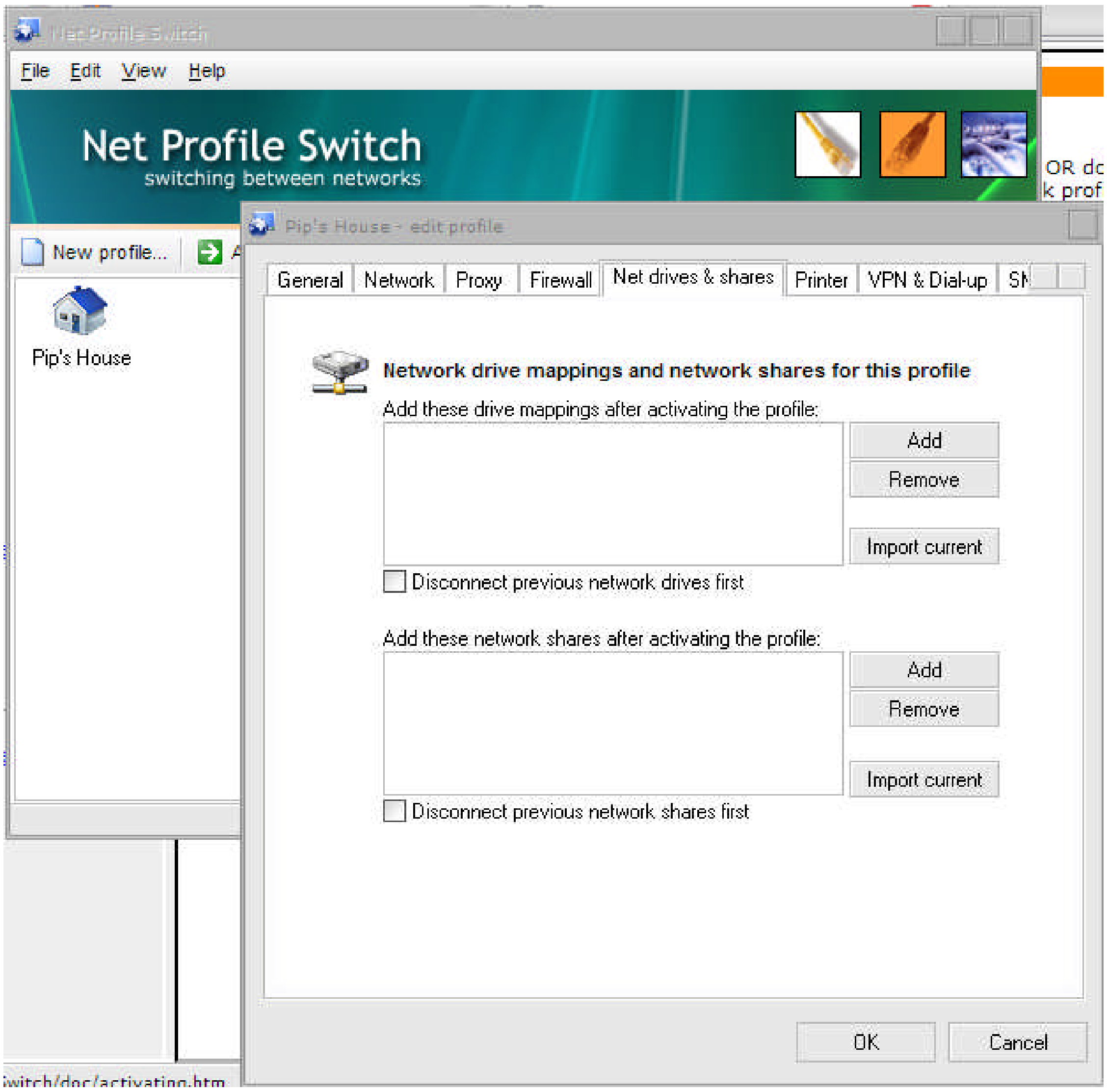
It is not a very user friendly piece of software and although it is designed to make the users life easier it supplies a lot of unwanted confusing information for not a lot of features.

Network Profile Manager

After finding this software in my initial research, when I came to test it, their website was down, and I was unable to review it.

Net Profile Switch

Net Profile Switch was a good piece of software which had all the features I was looking for, except again it required the user to switch profiles manually. This is the closest to the piece of software I intend to create.



Network drive mappings and network shares for this profile

Add these drive mappings after activating the profile:

☐ Disconnect previous network drives first

Add these network shares after activating the profile:

☐ Disconnect previous network shares first

Background

Windows Management Instrumentation

The Windows Management Instrumentation (WMI) is a means to collect information about a computer and notify the software implementing it when events occur or system settings change. It provides a common and simple way to collect data from many sources. The WMI comes with a similar query language to SQL (Structured Query Language) called the WQL (WMI Query Language). This system allows the user to search for and locate the information they require from the large amounts of data that the WMI gives access to. The WMI is a convenient and simple way to gather data about many different systems while using a common method. This project will make considerable use of the WMI as it gives easy access to a lot of useful information without forcing the user to have extra windows popping up while the software is running (As would be the case if a command prompt was used to run commands external to the program to fetch data.)

The WMI can be used to:

- "Start a Process on a Remote Computer
- Schedule a process to run at specific times on specific days.
- Reboot a computer remotely.
- Get a list of applications installed on a local or remote computer
- Query the Windows event logs on a local or remote computer"

(Microsoft Corporation 2008)

The WMI retrieves information about a system and displays it by modelling the object it is retrieving information about in a class. For example the Win_32_LogicalDisk class models a logical disk on a computer that is running the WMI. This is based on the Common information Model (CIM). The CIM is a publically available standard from the Distributed Management Task Force. (Available at http://www.dmtf.org/standards/cim/)
Using the WMI to collect data about the local system is possible as a user with restricted rights, however attempting to collect data about a remote machine using the WMI requires Administrator rights on the remote machine. This would be a problem in the context of this project as it is highly unlikely that the user will have administrator rights on all the systems connected to a network, especially when the network in question is not a home network but is at a workplace or a public venue.

The WMI uses the WMI Query Language to query the system for information. This system is based on the Structured Query Language and uses similar syntax. For Example;

would return the MAC address of all the network adapters on that computer. This is a familiar and simple to use interface which makes the WMI a good choice of system to use to gather the information necessary for this project.

MAC Addresses and IP Addresses

Media Access Control (MAC) Addresses are link-layer addresses that are permanently assigned to a network adapter when it is manufactured. They are unique to the adapter they are assigned to and never change. Internet Protocol (IP) Addresses are Network Layer addresses which are intended to be used as a locating device for a computer or other device. However they are not intended to *uniquely* identify a device. IP addresses can be static or dynamically assigned and are not always the same for each adapter, as MAC addresses are.

MAC Addresses are unique for each individual adapter. This is possible because the IEEE manages the MAC address space. A company which makes network adapters buys the right to use a certain amount of the address space. It can then use this set space to give unique MAC addresses from in that space to each adapter that they manufacture. The IEEE sets the first 24 bits for the company and then lets the company create unique combinations for the second half of the address (MAC addresses are 6 bytes long, meaning that 2^{48} combinations are possible). MAC addresses are useful because they are always the same for an adapter, whereas an IP address may change.

"An adapter's MAC address is analogous to a person's social security number, which also has a flat addressing structure and which doesn't change no matter where the person goes. An IP address is analogous to a person's postal address, which is hierarchical and which needs to be changed when a person moves." (Ross and Kurose 2005).

When an adapter needs to send a message across the network to another adapter it sends it to the MAC address of the other adapter. This message is then read by every node on the network that it comes across. If the node's MAC address matches that which the message is sent to then it reads the message. If not then it ignores it.

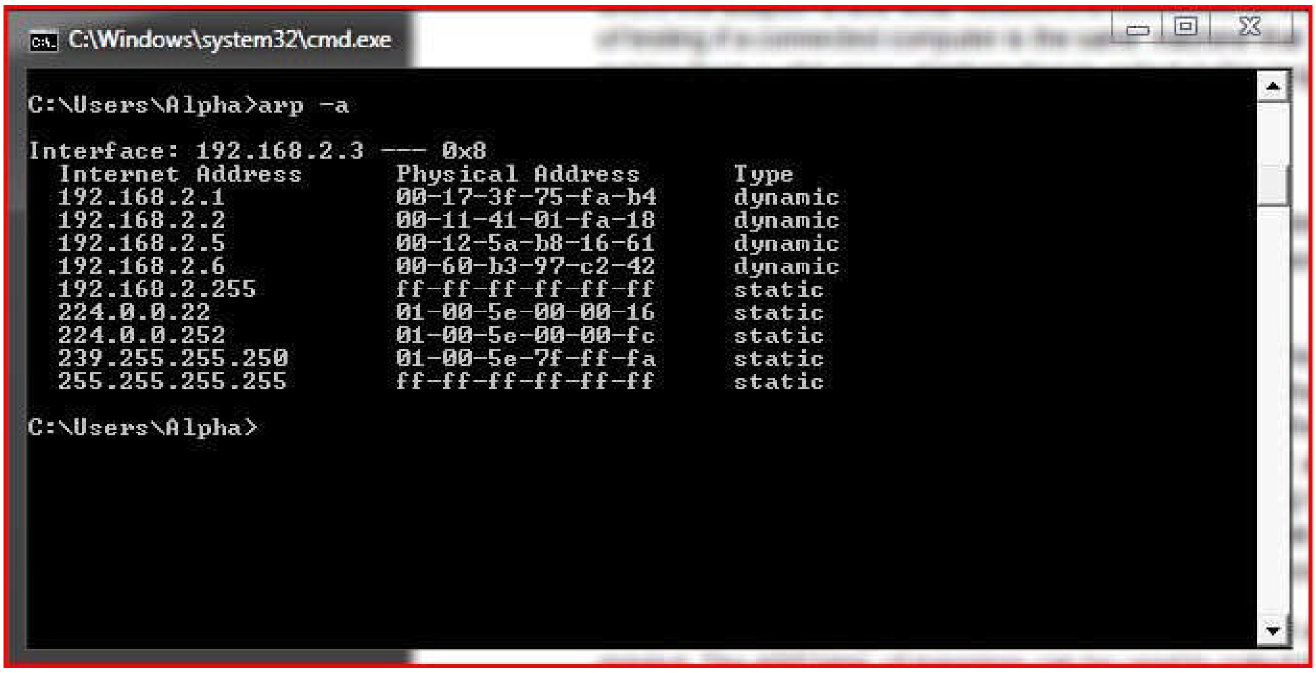
Because MAC addresses are always the same, but IP addresses can change depending on where the adapter is and what network it is connected to, they are a much more reliable means of testing if a connected computer is the same machine that has been connected previously, making them a vital piece of information to collect in this project.

ARP

Because both MAC and IP addresses are used to identify devices on a network there is a need to be able to translate between them. This issue is addressed by the Address Resolution Protocol (ARP)

An ARP module takes IP addresses as input and returns the corresponding MAC Address. For example; if a network node needs to send a packet to another node on the same network, but that node is also a web server and the sending node gets the receiving node's IP from a DNS, then the sending node will need to send both the IP packet and the Receiving Node's MAC address. But the sending node does not know the receiving node's MAC address, so it must use ARP to discover it. The sending node gives the ARP module the IP address of the receiving node and the ARP module returns the corresponding MAC address for that node.

ARP holds the mappings for IP to MAC addresses for a set amount of time, after which they are deleted. The ARP table of mappings can be used to collect the MAC addresses of all known IP addresses on the subnet. (See below for Screen Shot)



```
C:\Users\Alpha>arp -a

Interface: 192.168.2.3 --- 0x8
  Internet Address      Physical Address      Type
  192.168.2.1           00-17-3f-75-fa-b4     dynamic
  192.168.2.2           00-11-41-01-fa-18     dynamic
  192.168.2.5           00-12-5a-b8-16-61     dynamic
  192.168.2.6           00-60-b3-97-c2-42     dynamic
  192.168.2.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

C:\Users\Alpha>
```

Identifying a Unique Network

Identifying a network to be unique and then recognising it later as the same network poses quite a few challenges:

Network Variations

Networks are naturally updated periodically, when more storage space is required, when the network needs to be scaled to allow more users to connect or when a simple update is required for example. These changes could be as simple as a server being renamed, or a change of IP address, or as complex as an entirely redesigned network. This means that a system which recognises when a previously recognised network has changed must be able to deal with these variations. Using as little information as possible to give a network a signature would limit the amount of data that could change about a network but would also allow more margin for error during that process.

Using Different Network Adapters

Another factor to consider when attempting to recognise the same network after it has been connected once is that any subsequent connections to that network may not be through the same network adapter; for example the network could be connected using a wired link, and then the user could disconnect the wire and connect to the same network using a wireless link. Alternatively the laptop may have more than one wired network adapter. The solution must be able to deal with whatever means the laptop uses to connect to a network. It may also be possible for a network to be connected to multiple networks at once. This must also be considered during the design and implementation of the solution.

Similar Networks

Similar networks could pose a problem for this project unless a reliable and reasonably quick solution can be found. If the networks to be compared (i.e. the stored network, and the network that the laptop is currently connected to) are too similar then they may be mistaken for the same network. For example; if the networks both have a server with an IP address of 111.22.33.44, then if the system is not built to cope with such a situation then it will assume the networks are the same.

A simple way to avoid this situation would be to implement a system that bases the distinction between networks on variable that are less likely to change. Basing the system on the MAC Address would achieve this as it is a unique value for each network device. However, this is a more difficult piece of information to extract from a remote computer. A more complex way would be to create a system that 'weights' the data according to how reliable it is for distinguishing networks. This is a difficult solution as deciding how to weight the data would require a lot of research into how networks are changed and how often certain changes are likely to be made which there will not be time for in this project.

Proposed Solutions

The proposed solution must include an algorithm or method for distinguishing between unique networks and a method for changing settings to match the needs of the network. The project will make use of the Windows Management Instrumentation (WMI) to collect measurable information about the computer and the network(s) it is currently attached to. This data will be used to create a signature for that network which, when the computer connects to a new network, can be used to decide which network it is connected to, or if it has connected to a network for the first time.

The software will make use of the WMI to collect information about the network. This information could include:

- What other devices are connected to the network
- Meta-data about those devices, including;
  - IP Address
  - MAC Address
  - Name
- Whether the network is assigned IP addresses using DHCP or static addresses
- Proxy settings
- Network drives and whether they are connected
- Network printers and other devices.

Solution One: “Voting System”

Once the above information has been collected a ‘vote system’ will be used to decide if the network connected is the same as a stored network. This will work by checking each piece of data against the corresponding stored data, if it matches then it will ‘vote’ in favour of the network being the stored one. Each setting will have a different weighting to try and compensate for the likelihood of the settings changing.

This solution should provide a reliable means to judge whether the current network has been used before. It uses a lot of factors to decide if it is a stored network or a new one which means that there are fewer margins for error. However it may be a very slow method of deciding as it has to check a lot of variables (many per device connected!) and work out which are the most important pieces of information to use to base the decision on. It also would be difficult to implement as there is a lot of fuzzy logic to enable the software to make the decision.

Solution Two: “MAC Address based network signing”

This solution is based on the assumption that the actual hardware of most network routers or servers doesn’t change or if it does it is very rare. The software would collect as many MAC Addresses for connected equipment as it could and would then store these in a file to be retrieved when a network connects. Since MAC addresses are unique for each piece of equipment they are a good way to make sure that the devices connected are the same devices that were connected the last time the laptop connected to the network. This system would work in a similar way to the above system but would only check and store MAC addresses, and would have a similar voting system. For instance if the router and two connected computers had the same MAC addresses as the last time the laptop connected but a printer for example had a different address, then the program would vote in favour of the network being the same because a printer might be changed reasonably regularly. If, however, the router or the main server had a different address then it is more likely that the network is different and at this point the software would ask the user for clarification as to whether this was a new network or a stored one which had changed.

This solution should provide an equally reliable means to check the networks connected to the laptop, it uses one factor but it is a reliable factor and if it changes then the user can be asked what they would like to do. It should also be faster as it does not have nearly as many checks to make, only one per device connected to the network. It still has some complex logic to implement but this is kept to a minimum.

Solution Three: "User Input"

This solution relies on the user to select the location that they are in and then will simply configure the laptop to how the user specifies when they first set up the network. This would be the simplest way to configure the system for new networks but it requires the user to say which network they are connected to which would become tiresome if they change networks often. Also this solution does not meet the specification fully as the system should automatically update the location of the computer, not rely on the user to update the location manually. There are already a few solutions like this 'in the field' and so this is 'reinventing the wheel' so to speak. There is already software which does this available (NetProfile switch Pro for example).

Chosen Solution

The solution I have chosen to implement is the second solution 'MAC Address based Network Signing' as this allows a reliable means of distinguishing between network, but should be faster and easier to implement than the first solution which would be more likely to be correct in situations where it would be hard to decide if the network is known; such as when the router or main server hardware changes.

Different networks are very unlikely to be made using devices that have been used in networks that have previously been saved in the programs memory. This solution ignores these situations as they are very unlikely and the solution saves time by not checking to see if the devices have the same IP addresses and other changeable details, it simply relies on the unique MAC address to match the network devices. This opens the solution to a potential problem whereby if an individual were to use the same physical piece of equipment to create a new network then the software would potentially regard the device as being on the 'old' network and this could cause the user confusion. These situations however are extremely unlikely to occur and therefore will be ignored for this application.

Since this solution requires much less data than the other proposed solutions it should be much faster to run and this should allow for a much more user-friendly experience for the end user. However this solution requires a more complex method of collecting the information as it will have to collect the MAC addresses of remote machines and devices. This would require administrative rights if the WMI was used to collect the data as the software will have to set up a remote connection to the device. Using another method which does not have this drawback would be preferable. An alternative solution which does not require administrative rights on the remote device is to use the WMI to get the IP of the local machine's Default Gateway which should be a router or hub or similar on the network in question, and ARP that IP to get the devices MAC Address. This method should work reasonably quickly but will only work if the remote device is on the same subnet as the local machine.

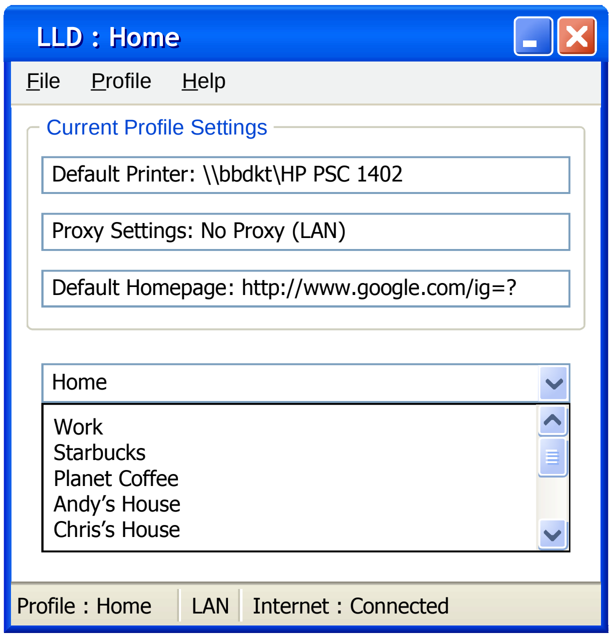
Design

Basic Design
The software will be designed in two parts, a ‘Network Checker’ and a ‘Network Changer’. The Network Checker is the main part of the project and the main aim. The Network Changer is an extra aim if all should go well, but does not take such high priority as the Checker. These are discussed in more detail below;

Network Checker
This will be the part of the program which does the ‘hard work’. It will be a System Tray Application which will run in the background on the laptop and will periodically check whether the network(s) connected to it have changed and if they have then it will attempt to identify the location of the laptop by comparing the data gathered from the current network to the data stored in each of the files stored as outlined above. This will only occur if the network has changed since the last check. The checks will occur every minute. This is because if the checks occur too frequently then the software will consume system resources, but if it runs too infrequently then when the network does change it could potentially take the software a long time to register this. When this program registers a change in the network and recognises the network from the saved profiles it will inform the network Changer which will change the settings to match those which are saved.

Network Changer
This portion of the software will only be run when the Network Changer registers a change and recognises the network. It will change the local settings to match those required by the network. These settings will be sent by the Network Checker at the same time as it calls the Network Changer itself. These changes will be made by using the WMI. WMI supports the setting of many different settings by a simple and consistent method, which makes it ideal for this task. Also all of the settings to be changed will be local; none will be on the remote machine, which means that the WMI can be used successfully as it does not require administrative rights to use on a local machine. This section of the solution is not as high-priority as the Network Changer, as it is in addition to the main aim of the project.

Initial Mock-Screenshot of Idea
This screenshot shows my initial idea of how the program might look when it is finished. It shows the settings and a list of profiles which the user can choose from, as well as the program select them automatically.



Default Printer: \\bbdkt\HP PSC 1402

Proxy Settings: No Proxy (LAN)

Default Homepage: http://www.google.com/ig=?

Home

Work
Starbucks
Planet Coffee
Andy’s House
Chris’s House

Profile : Home | LAN | Internet : Connected

Choice of Language
The language which will be used for this project

is C# in Visual Studio. This has been chosen because it offers a good environment to work in and supports WMI natively. It is also managed code which although being slower to run is easier to use. The developer also has more experience with C# than with any other language.

Storage

The system will have to store data such as saved network profiles in a form which can be read by the software when required, including after the program has been closed and re-opened at a later date as the software is designed to be used to 'remember' network profiles and settings. It would be no use if the software did not have a mechanism for saving this data because it would then have to re-profile each network every time it connected to it which would defeat the point of the program. Also it would be likely that the laptop would be closed down between changing networks if they are in separate locations so the program would only ever remember one location.

The software will save the data to a file by means of serializing the object which holds the profile to disk. Each network profile will have its own file inside a 'saved profile' folder in the software's root folder. This serialization could be encrypted if security issues arise with the data, for example saving a companies network settings onto a portable computer may compromise the security of that company, especially as portable devices are more likely to be stolen. Encrypting the serialized data would solve this problem as it would no longer mean storing the data in clear text. This would also stop any malicious attacks at changing a person's settings by editing the saved files as they would appear to be gibberish to any human.

User Interface

The solution will be made in two parts;

The Network Checker

This section of the program will run in the background for the most part, it will only inform the user when the network has changed, and ask them what to do if it doesn't t recognise a network location. When either of these events occurs it will pop up a message asking what to do, or informing of the changes made. There will also be a Windows form for editing settings for network profiles and viewing the saved profiles. It will also inform the user when the network Changer changes any settings, which will occur when the network checker recognises a new network.

The Network Changer

This section of the program will not have a user interface as it will take its commands and settings to use from the saved settings and the Network Checker part of the implementation. It will simply be called by the Network Checker, will make the changes and will then exit until it is called again.

In-Depth Design

Problem Statement
Laptop users may use many different networks in different places at different times. These may each require different settings, but it may not be apparent to the user what settings need changing for each network, and it may not even be obvious when a network has changed due to the transparent nature of wireless networking.

Business Goals
Below are the goals for this project, arranged according to priority.

HIGH

- Profile Networks and save data about each network in a re-usable format.
- Recognise when a network connection is lost, gained, or a new network connects and Inform the User.
- Develop a system to compare networks and determine if they have been accessed before.
- Compare saved networks when new networks connect to determine if the new network is the same as a saved one.

LOW

- Recognise when System settings need changing for a network.
- Change system settings for a network when required.

System Scope
The proposed system will recognise when a network connection is lost, gained or added to a laptop. It will then attempt to decide if the currently connected network has been connected before by comparing it to stored data about previously connected networks. The system will then inform the user that a network connection has been detected and attempt to tell the user the network location the laptop is in. If a network is not recognised it will ask the user to enter information about the network and will then save this data to disk to be used for later comparisons. It will store this data in a secure manner to ensure that potentially sensitive data about the networks the user has access to is not available should the stored data be lost or stolen.

System Vision
The proposed system will provide a simple and easy to use way for a user to know which networks they are connected to, and update their settings accordingly.

Project Risks
Risk Analysis

In the table below are listed possible risks which are rated for severity and likelihood. The significance of each risk has then been calculated. The high significance risks must be considered carefully when undertaking this project.

Hardware Failure – This is a high significance risk because although it is unlikely, if it were to occur then I would potentially lose all my data. To combat this I will create regular backups of my project an various places and keep them up to date.

Missing the Final Deadline – This is a high significance risk because if I were to miss the final deadline then I would lose marks rapidly and potentially fail my project. I will avoid this by keeping on top of all my work and regularly referring to my Gantt chart to check my progress.

Loss of Data – A medium severity risk, to avoid this occurring, I will ensure my backups are regular and up to date.

Cognitive Shortfall – A failure to understand something required for the project, this could be a problem. To avoid this problem I have undertaken much background research for important phases of the project.

Incompatibility Issues – If the program has incompatibility issues then the project will have a much smaller potential user base, which will reduce its usefulness, this is a fairly likely problem, but to iron out as many incompatibilities as possible, the program will be written for the most common operating system and using a common framework.

Personal Injury – If something were to happen to me, the project may suffer. This is a low significance risk as I feel it is unlikely to occur during the project.

(Note: Significance = Likelihood x Severity)

Risk	Severity	Likelihood	Significance
Personal Injury	Low	Low	Low
Loss of Data	Medium	Low	Medium
Hardware Failure	High	Low	High
Cognitive Shortfall	Medium	Low	Medium
Incompatibility Issues	Low	Medium	Medium
Missing Final Deadline.	High	Low	High

Requirements

Functional Requirements

The user should not concern themselves with the solution until it is unsure of what to do. It should run in the background.

The User should be able to view, edit and delete saved network profiles.

The User should be able to specify settings for newly discovered networks.

Task List

#1 Initial Report*

A report outlining what I plan to achieve with this project and the time span I intend to achieve it in.

Estimated Duration: 7 Days

#2 Initial Design of Network Checker

Design the first component of the software which will gather information on the current network.

Estimated Duration: 8 Days

#3Initial Design of Network Changer

Design the component which will update the computer to reflect the information gathered from the Network Checker.

Estimated Duration: 6 Days

#3 Final Design of Network Checker

Refine the initial design of the Network Checker.

Estimated Duration: 10 Days

#4 Final Design of Network Changer

Refine the initial design of the Network Changer.

Estimated Duration: 6 Days

#5 Create the Network Checker from the final design

Estimated Duration: 8 Days

#6 Create the Network Changer.

Implementation of System Tray Application to update the computer to allow for network changes.

Estimated Duration: 14 Days

#7 Testing

Test the project. This will involve Unit Testing and Integration Testing

Estimated Duration: 23 Days in total

#8 Interim Report*

This will involve a progress review and any updates and revisions that may need pointing
This will involve a progress review and any updates and revisions that may need pointing out at this stage of the project.

Estimated Duration: 13 Days

#9 Documentation (ongoing)

This will be documents detailing my project as I complete it.

Estimated Duration: Ongoing Alongside Development

#10 Final Report*

This will show what I have achieved in this project.

Estimated Duration: 66 Days

#11 Project Appraisal*

This will review how I have handled my project and managed my time.

Estimated Duration: 14 Days

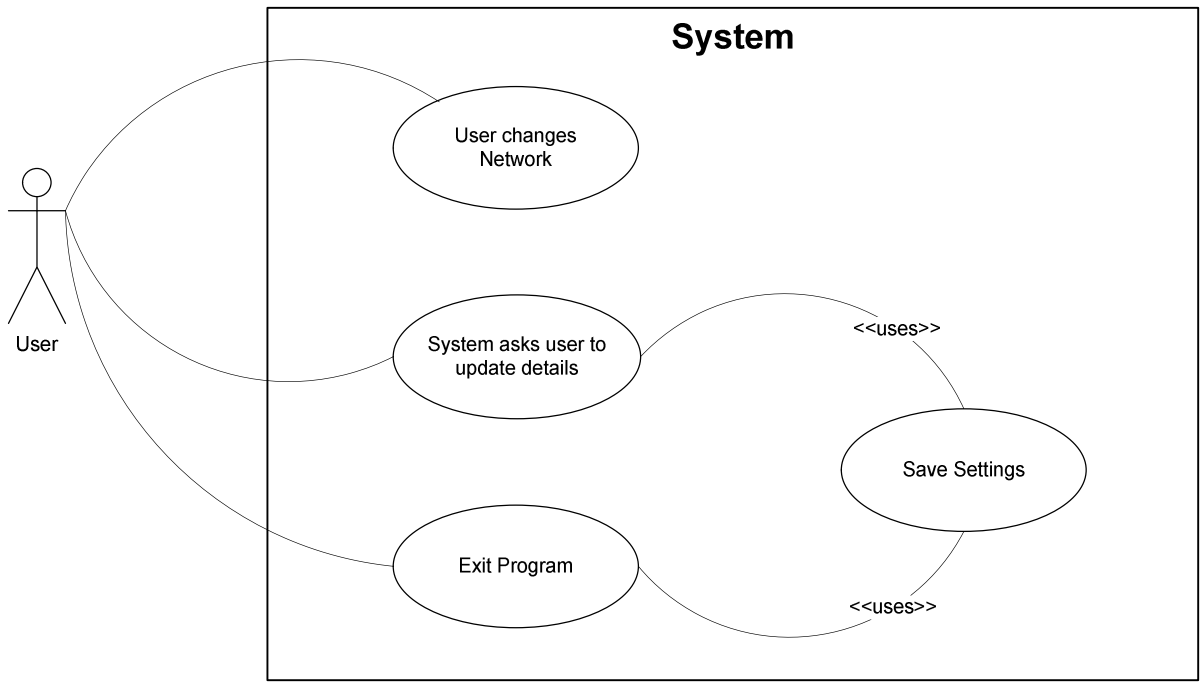
#12 Presentation

Presentation on My Project

Estimated Duration: 43 Days

Use-Case Diagram

The system has a simple use-case diagram as it is designed for the most part to run in the background of the user's machine and only 'talk' to the user when it is unsure of what to do.



Operational and Infrastructure Requirements

Performance

The system must respond to a network changing or being lost within a minute of the change occurring. If the system takes longer than this then the user may become impatient with the system and look for an alternative solution.

Scalability

The system is designed for single system use. It is not required to be used to set up or inform multiple systems of changes.

Accessibility

The system must have visual notification when a network changes and when settings are changed on the laptop.

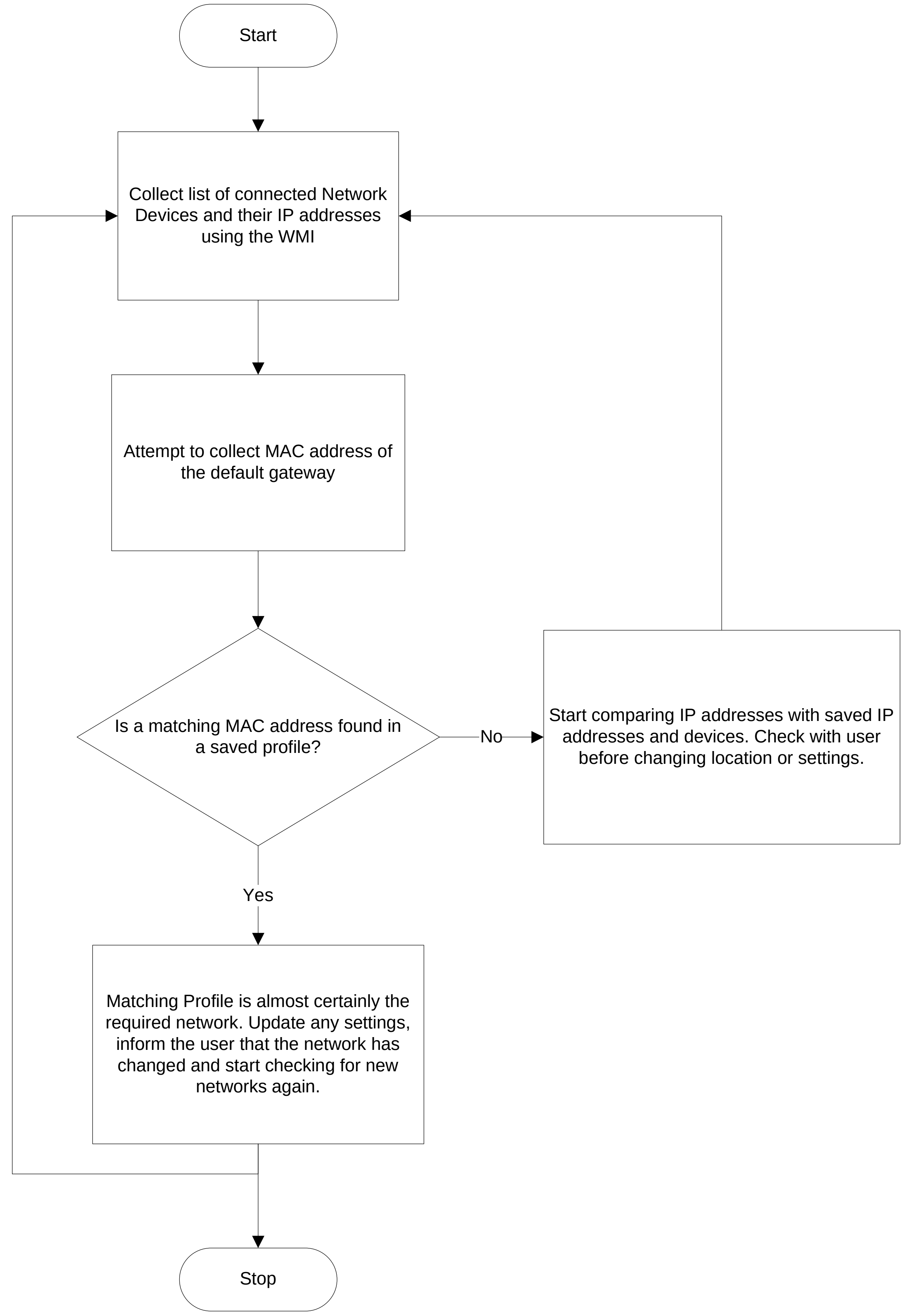
Security

The system must store any system or network data in a secure manner as this data could be sensitive information

Algorithm

There is one main algorithm that has had to be developed for this project; the algorithm to check whether the network that is currently connected is the same as a stored network profile. This is explained below.

Algorithm Flow Chart


Start
Collect list of connected Network Devices and their IP addresses using the WMI
Attempt to collect MAC address of the default gateway
Is a matching MAC address found in a saved profile?
No
Start comparing IP addresses with saved IP addresses and devices. Check with user before changing location or settings.
Yes
Matching Profile is almost certainly the required network. Update any settings, inform the user that the network has changed and start checking for new networks again.
Stop

When the system starts it will begin a loop of execution which will 'watch' for changes in the network status of the laptop. This will happen by means of the WMI which can call a method when certain system events occur. When the network changes or disconnects then this algorithm will be run.

When the algorithm starts it firstly collects a list of IP addresses and the devices they belong to on the network using a WMI Query. Once this has been completed, the program will attempt to collect the MAC address of the laptop's Default Gateway to the internet. This is because the Default Gateway is likely to be a constant on the network. If this is successful, the software will start comparing the discovered MAC address to that of each of the stored profiles Default Gateway MAC addresses. If it is unsuccessful then it will start comparing the IP addresses of the devices and the device names to those in the stored saved profiles. This is a slower and less concrete method for determining the network, and as such it is used as the 'backup' method.

Once the system has found a recognised network it will inform the user, and if it has found the location by using the MAC address method then it will update the settings too. If it has found the location by using the IP method it will inform the user but ask for confirmation before updating any system settings. When this is complete it will begin to re-run the loop which will check for network changes.

If a network is not recognised, or the user rejects a location that the laptop is thought to be at then the software will ask the user to enter simple details about the network, such as the name and location of the network, ready to be stored in a new profile. This will happen by means of a windows form which will appear when the system detects the need or when the user manually selects it. This data will then be saved into a profile with the current system and network settings to be used for identifying the network at a later date.

Algorithm to retrieve the Default Gateway's MAC Address

The Default Gateway's IP address is a simple matter to retrieve, it can be found very quickly using the WMI, but the MAC address is a different matter. Because the MAC address is on the link layer and the program trying to retrieve it is some 3 layers up on the Application Layer, it can be difficult to retrieve. Two methods have been looked at during the course of this project. The first; and the method which is implemented in the program is outlined first.

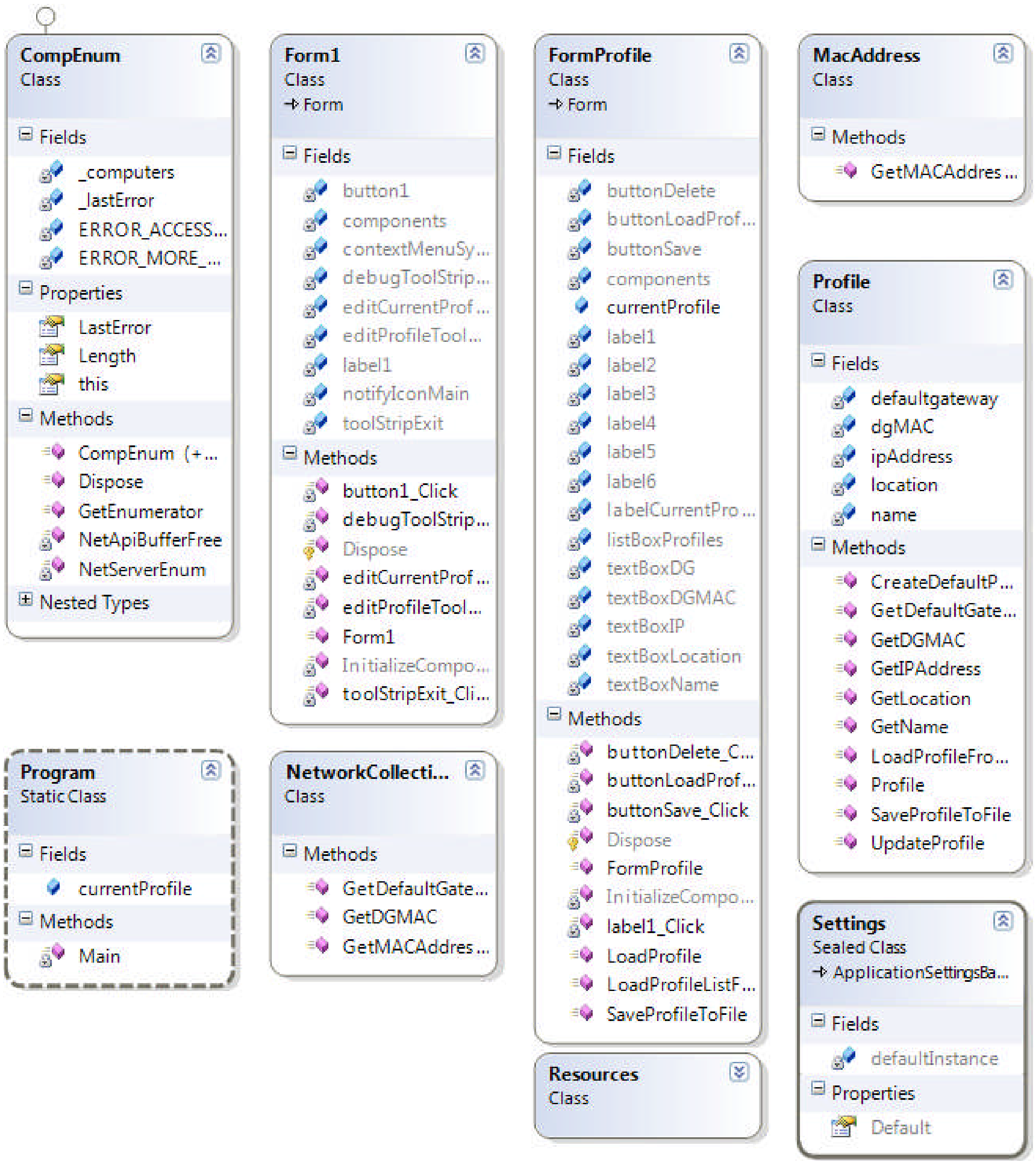
Method One

Collect the Default Gateway IP by using the WMI method. Once this has been collected, set up a connection to that IP and request the MAC address of that Network Adapter. This method will not work if the user does not have authentication details. These are not always needed and in some cases it will work fine. This method appeared to be the optimal method until late in development when it was tested and discovered to be extremely temperamental and dependant on a lot of settings outside the local user's control. Sample Code for this algorithm is available in Appendix One.

Method Two

The default Gateway IP address is collected in the same way as the first method, but instead of using the WMI to connect to a remote machine, a local ARP table request is made. This is made by using a command prompt which is programmatically invoked and hidden from the user. The output from this command window is fed into the network profile program and processed to find the corresponding IP address and thereby the MAC address. This method is more reliable than the first assuming that the laptop has 'spoken' to the default gateway in the last twenty minutes or so, which it should have done as it is usually the main "hub" of the network. The only drawback to this method is that the ARP request may flash onto the user's screen briefly while it is set to hidden, this should not be a problem, but may become annoying if the network changes frequently. This method would be one of the changes made to the program if more time was available, but it was not discovered to be viable until late on in the development cycle.

Class Diagram



Test Plan

Below are the main tests run to check the programs functionality.

Test	Expected Result	Actual Result	Pass or Fail	Comments
The software should be unobtrusive to the user under normal operation	Software should only prompt the user for information when it is unsure of what to do	Software works correctly	Pass	The software passes this test with flying colours. It only requests the user's attention when it does not recognise a network.
Disconnecting a Network	The software should recognise when a network disconnects.	Software works correctly	Pass	The software successfully recognises when a network disconnects and informs the user by means of a 'bubble notification'
Connecting a Network when no others are connected	Software should spot new network and attempt to match it to stored networks.	Software notices new network and runs network matching algorithm	Pass	Software attempts to recognise network by running the algorithm, Test passed.
Connecting to a previously connected network with no changes	Software should recognise network from the stored files	Software attempts to find network, and is successful	Pass	Successful test.
Connecting to a previously stored network with new devices added to it.	Software should recognise network from the stored information	Software sometimes recognises the network, dependent on what devices have been added.		Inconclusive Test
Connecting a network with devices removed	Software should recognise network from the stored information	Software sometimes recognises the network, dependent on what devices have been added.		Inconclusive Test
Connecting a Network with devices added, removed and changed	Software should recognise network from the stored information	Software sometimes recognises the network, dependent on what devices have been added.		Inconclusive Test

Test Report

This particular Project is difficult to test due to the nature of it. Many factors could change between using a network and reconnecting to it at a later date. It could also depend on the length of time it was disconnected; for instance a device on the network could be down when the laptop connects which could affect the results. It is extremely difficult to test the project in its intended usage. For this reason a test harness which would send the program 'fake' data to test its behaviour when the system tried to use the data as it would normally. This data included both extreme data and data which could reasonably be expected to be received by the system in its normal use.

Project Appraisal

This project requires more work to become a fully functional piece of software. It has many problems which need ironing out before it could be put into daily use. However I feel that I have learnt a good deal about the software development process and about networking. The project has taught me that planning and sticking to the plan is an extremely important part of the process. I feel I have come away from this project with a much better idea of how to specify, design and implement a large scale program. I have also learnt a lot about communicating with people, which is not always the easiest thing to do.

The project has been very challenging and has been extremely difficult to implement. The main reasons for this are;

- Insufficient experience in the field of networking and identifying unique entities when two entities may be very similar but subtly different.
- Technical challenges when attempting to gather the MAC addresses of remote machines. It is possible to gather the MAC addresses of remote machines but it is difficult, especially when the user does not have any kind of privileges on the remote computer, as is the usually the case in this type of project.
- Time Constraints. This project has taken up much more time than was first predicted when the project began. This is because all sorts of technical and implementation issues have been discovered. Also a lot more research has had to be carried out than first expected due to the issues discussed above.
- Other work/projects. More time has been spent on other work than was expected. This is partly down to poor time management, and partly down to having less time to spend on these projects due to personal issues.

All in all the project went ok, the secondary aims of the project were not met, but they were always intended to be 'extras' if the project finished with time to spare. A lot has been learned from undertaking this project; Time management skills have been improved to cope with the demands of having several coursework and other projects on the go at the same time. If the project were to be undertaken again more time would be allocated to research and development as these are the main areas I feel were lacking during the project, they had to be rushed to keep within the time plan of the project. Also more consideration would be given to the amount of time other projects and work would take up during the project and the time plan adjusted accordingly.

The project also was difficult to test as it attempts to cover a lot of ground with regards to possible network changes. The program is designed to allow many different variations to occur on a network and yet still recognise that network as one it has met before. This was attempted to be circumvented by 'spoofing' data from networks and passing it into the system programmatically as it would be extremely difficult and time-consuming to change network settings repeatedly for the purpose of testing the software. There would also be a huge number of possible variations to test. All of which could cause an unexpected problem or error, and it would require thousands of man-hours to test the software exhaustively.

Further Work

Further work on this project would include a faster algorithm for collecting the MAC address of a remote machine. This was the most challenging aspect of the work and in hindsight a better way of collecting this information could be found by using ARP tables and extracting the relevant data rather than trying to connect directly to the remote machine. This would avoid security and privilege issues that have arisen while using the WMI method to collect this data. Connecting to the machine directly requires the user to have administrator privileges on the remote machine which is unlikely if the user is simply connecting to a network at work or at a public WIFI hotspot. Using the local machines ARP table to get the MAC address, although it seemed like more work at the start of the project because of the amount of string manipulation that would be required to get the required data out, with hindsight would appear to be the better solution as it does not require administrator privileges on either machine. This is a much more reliable approach, even though the ARP table may not have the Default Gateway’s MAC address it is more likely to have it (given that the laptop will be using it to connect to the internet which is one of the main uses of a laptop these days) than the user is an administrator of the network.

The other obvious piece of work to finish this project would be to implement the Network Changer part of the application. This is a fairly simple task now that the main bulk of the application is in place. It is a simple matter to add details to the network settings class and these would then only need setting using the WMI, which would only take at the very most ten lines of code per setting and more likely only one or two lines.