University of Southern Queensland

Faculty of Health, Engineering & Sciences

Remote Access Laboratory Design and Installation

A dissertation submitted by

Keith Dickmann

in fulfilment of the requirements of

ENG4112 Research Project

towards the degree of

Bachelor of Engineering (Computer Systems)/Bachelor of Information Technology (Applied Computer Science)

Submitted: October, 2013

Abstract

The university of Southern Queensland (USQ) employs remote access laboratories (RAL) to allow external students to perform practical experiments remotely without the need to travel to USQ to perform these experiments on campus. Many topics require practical experiments as part of their learning experience. One such topic is computer networking. This dissertation details the research, design and implementation of a networking specific remote access laboratory.

The concepts of a remote access laboratory and several existing technologies were researched. Based on this research and the existing practical networking course at USQ the requirements for a networking lab were created an analysed. From this a design was created which comprised of multiple virtual machines, a web interface, a configurable switch, and a program written to supervise the lab. This design was then implemented and tested.

Certification of Dissertation

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.

Keith Dickmann

0050101371

Signature

Date

Contents

List of Figures

List of Tables

.

Nomenclature

CIFS	Common Internet File System
CLI	Command Line Interface
DHCP	Dynamic Host Configuration Protocol
IP	Internet Protocol
IT	Information Technology
LAN	Local Area Network
NFS	Network File System
RAL	Remote Access Laboratory
RDP	Remote Desktop Protocol
SMB	Server Message Block
SSH	Secure Shell
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNC	Virtual Network Computing
VPN	Virtual Private Network
VRDP	VirtualBox Remote Display Protocol

Chapter 1

Introduction

1.1 Overview

The University of Southern Queensland provides Remote Access Laboratories (RAL) in order to provide students with off-site access to practical and laboratory experiments. Many courses require practical learning, using dedicated laboratories and equipment with which students can use to perform experiments in order to gain hands-on practical experience. However sometimes a student may not find the time to do the experiments, or is absent due to conflicts or illness. There are also many external students throughout the world who's learning is provided solely online or through textbooks and have no means to do the practical exercises. For example the former Faculty of Engineering and Surveying approximately 70% of students were studying externally. In order to complete the practical courses external students have to travel to USQ during the on-campus holiday period to attend Residential Schools to perform the experiments. While not meant to replace the Residential School system, Remote Access Labs do provide students with the means to perform practical experiments remotely, supplementing their on-campus practical classes, or providing them an opportunity to perform the practical experiments and prepare for residential school activities.

One topic that requires practical experiments is computer networking. A computer network is a communications network that groups a number of computers together and allows them to exchange data and communicate with each other. Setting up a computer network usually involves connecting multiple computers together and setting up the computer's operating system and software to use the network. These activities involve physically interacting with the computers that are networked together and hence makes up the practical learning part of computer networking. A Remote Access Lab that provides this practical experience will be beneficial to students.

1.2 Project Aims

The aim of this project was to design and implement a remote access laboratory that will allow students to learn the concepts of computer networking. The lab uses multiple systems and multiple operating systems. One of the systems is configured to supervise the network while the rest are set up to allow students to perform experiments on them, as well as the software needed to perform those experiments. The software used is, as much as possible, be made up of free/open source software, as this minimized the costs involved in the implementation and maintenance of the laboratory. The computers and the network configuration must be able to be manipulated by the students in accordance with their experiments, and must be reset to a default state when needed. The laboratory also provides a means to allow students to access the machines to perform the experiments.

Experiments have also been created for the students to perform. The content of the experiments is be based mainly upon the Electrical and Electronic Practice E course at USQ, which is focused on operating systems and networking, as well as around the activities that networking technicians and system administrators do in their jobs in the real world. These Exercises aim to teach students the basics of computer networking, how to configure networks on both Windows and Unix systems, how to use commonly used Windows and Unix networking tools, and set up and configure network based services.

The laboratory also provides a remote interface that is simple and easy for the students to use. It must provide all the information for the exercises to the student in a uniform and consistent manner. A simple and easy means for the student to connect to the laboratory computers must also be included. The students should be able to manipulate the laboratory computers as if they were physically sitting in front of them.

1.3 Specific Objectives

The Specific Objectives are as follows:

- Undertake a basic requirements analysis and establish potential experiments and learning outcomes of the experiments.
- Undertake a comprehensive literature review covering all aspects of this project including computer networking, network management, automatic system restoration, and remote interfacing.
- Design a networked system consisting of a configurable switch and multiple virtual systems, that allows a user to manipulate the network configuration and the computers on the network, and can be reset to a known default state when needed.
- Build and test the system.
- Create a number of practical experiments that will enable students to comprehend networking concepts and learn about network configuration.
- Trial the experiments and evaluate the practicality, ease of use, and reliability of the system and make modifications as necessary.
- Undertake trials with students to evaluate the system and experiments.
- Document the project and write an academic dissertation on the research.

1.4 Thesis Outline

The outline of this thesis is as follows:

1. Introduction
2. Literature Review
3. System Design
4. Implementation
5. Testing

6. Conclusion

Chapter 2

Background Information and Literature Review

2.1 Overview

Accessing a computer remotely has is not a new concept, in fact methods of accessing a computer remotely have been around since networking has been invented. Configuring network services on a remote machine is not a new concept either. Many of the worlds servers are located in data centres and have little to no physical contact with its users. Using remote labs is also an existing concept as USQ already has remote labs in place for other topics.

This section looks at the concepts related to the design of a networking remote access laboratory such as computer networking, operating systems, and system administration software. The frequent use of these concepts and their learning value. The methods used to manage a computer network. The software and methods available to automatically restore remote computer systems, and the software and methods used to access a computer remotely.

2.2 Online Remote Laboratories

The use of the internet is becoming more prevalent in higher education. Many Educational institutions are implementing or have implemented e-learning environments as a way of teaching to a more geographically disperse audience (Gomes & Garca-Zuba 2007). In all fields of engineering, laboratory test and experimentation are an essential part of a student's education, and are usually provided with hands-on laboratory classes. However these classes to come at a cost of space, equipment, and maintenance staff. Remote laboratories allows for remote access to experiments, equipment (physical or simulated), ready all the time, without the restrictions associated with classes such as personal time and location (Gomes & Bogosyan 2009). There are however different factors that can contribute to the effectiveness of Remote Laboratories.

2.2.1 Time on Task

This is probably the biggest advantage remote labs have over traditional classes. Classes usually have a limited time to perform the experiments and in some cases, the experiment may span multiple classes. Much of the class is dedicated to learning about the experiment, and knowledge may be forgotten between classes, therefore more time will need to be spent on the experiment. The working speed of the student also comes into play. With remote labs students get much more time to absorb information, and more crucially, they are presented with the opportunity to re-perform experiments (Bright, Lindsay, Lowe, Murray & Lui 2008).

2.2.2 Learning Style

The learning style of student plays a significant role in the students learning. Students have different preferences as to how they are taught, which affects their motivation and satisfaction in a learning environment (Bright et al. 2008). This factor can be both an advantage and a disadvantage to a remote lab. The disadvantage is that some students prefer to handle physical hardware work alongside peers, and seek teacher assistance, which is unavailable in a remote environment. Another disadvantage is that if the student at home they can be easily distracted. Some student however prefer working alone, and using a remote environment can remove pressure off the student to complete the experiments, especially if they work at a slow pace, or intend to re-perform the experiments.

2.2.3 Prior Experience

Students with prior experience tend to perform better in experiments than students who are only starting to learn the subject material. This is significant as students without prior knowledge tend to seek teacher assistance, which is not available in a remote environment. The networking lab however is designed to give students experience prior to performing the experiments in the lab classes.For example, students, who only have little knowledge of Java programming will have problems performing experiments and need special support of a tutor to learn Java programming. While students with experience in programming will be able to work mostly independently (**?**).

2.2.4 Social Interaction

The biggest disadvantage of remote labs to the complete removal of social interactions between students, and more crucially, between student and teacher. Laboratory classes provide a learning environment where instructional support can be critical for the student's learning (Bright et al. 2008). With a remote lab there is no teaching support and students have to learn the subject material by themselves, using provided information, or by researching other sources of Information. However this can also be seen as an advantage, especially where that the norm in some working environments (Bright et al. 2008). Laboratory classes also allow students to work together on a common task. While this is also possible with a remote lab it proves to be much more difficult. This is because organising everyone to access the lab at the same time, and to organise a common form of communication (e.g. VoIP, IRC, etc) can be very difficult, whilst lab classes have a specific time and location allocated to it. The USQ's lab, however, will comprise of experiments that can be completed individually, and with the wealth of information on the internet, lessens the need for peer interaction.

2.2.5 Perception of Hardware

There can be a difference between students using physical versus using virtualised or remote hardware. Lab classes give students the opportunity to handle, inspect and experiment with the hardware directly, allowing for a more in-depth understanding of the hardware used (Bright et al. 2008). It also allows for experiments where hardware needs to be changed or modified. With remote labs this physical handling of hardware is non-existent, so students just have to accept "it's there", but this doesn't give the student an understanding of the hardware. Changing or modifying equipment is usually virtualised, or not included at all, meaning that experiments where hardware is modified are limited, or non-existent.

2.2.6 Building Blocks of a Remote Lab

A recent paper (Alves, Gericota, Silva & Alves 2007) outlines the basic building blocks of a remote laboratory.

Experiment Server

The experiment server is what the student performs the experiment on. It is either connected directly to the experiment equipment or is the experiment equipment.(Alves et al. 2007). The networking lab makes use of four experiment servers, as required by the experiments.

Media Server

The media server provides audio and visual feedback to the student from the remote experiment. For some experiments the media server can be integrated into the experiment server, or not used at all (Alves et al. 2007). The networking lab does not use a media server as visual feedback will come from the experiment servers.

Web Server

The web server gives the student with all the information required to perform the exercises (Alves et al. 2007).

Access Server

An access server provides client with access to the lab (Alves et al. 2007). The USQ remote access labs system provides this functionality.

Provider Server

The provider server provides access to a number of remote labs (Alves et al. 2007). The USQ remote access labs system provides this functionality.

User Clients

User clients can either be students, performing experiments, or teachers, using the lab as a demonstration (Alves et al. 2007).

2.3 Computer Networking Education

Computer networking is an integral component of Information Technology (IT) systems. With many organisations using their own computer networks and the increasing dependency on the internet, the importance of computer networking cannot be understated. For this reason many IT, engineering and business courses teach computer networking because it is heavily used in their areas (Sarkar 2006).

Computer Networking is becoming increasingly difficult to teach to students. Ten years ago a single course was sufficient to teach a student about computer networking, in the present day however an entire undergraduate program can be dedicated to computer networking (Chang 2004). Computer networking is also an abstract topic to many students. In a typical classroom they cannot see networking equipment and cannot visualise packets and protocols for themselves (Chang 2004). Even when students actually learn networking concepts in the classroom, it does not mean that the student has a complete understanding of the topic, nor does it mean that they are able to apply those concepts in real life (Chen 2003).

Creating an actual computer network and performing experiments draws students closer to computer networking principles and reinforces information that they have learned in classes. For a small class in 2002 80 percnt of students strongly agreed that practical experiments helped them understand computer networking and helped the aquire practical skill (Chang 2004).

There are many aspects to computer networking. The basics however usually cover creating customized networks, network protocols, file permissions, and the setting up of server applications (Sarkar 2006).

2.4 Network Management

A computer network like the one the lab will use typically involves many computers behind a gateway. The lab will have to be managed somehow. A previous implementation of a remote access lab consisted of a management station, several hosts, and several routers and makes use of a management LAN to manage the lab machines (Yoo & Hovis 2007). The main difference between this lab and one that will be created is that users do not access the lab machines directly, they connect to a user interface which in turn connects to the lab machines and configures them. In the USQ networking lab students will be required to interface directly with the lab machines to perform the experiments.

Another previous implementation makes use of a web server, virtual machines and something called virtual patch panels. These patch panels (also called physical layer switch) allows connections to be created without having physically plug and unplug cables (Rigbey & Dark 2006). for the USQ lab this functionality can also be done using the VLAN feature in a configurable switch. The previous implementation has students connect to a web server, connect to the lab via a virtual private network (VPN), and then connect to the virtual machines using remote desktop software (Rigbey & Dark 2006). As access to USQ's lab will be done through USQ's remote access lab system the use of a VPN is not required, and the webserver will be running on a machine that manages the lab.

2.5 Virtual Machines and Automated System Restoration

Automated system restoration refers to automatically restoring a computer system to an earlier known state. While this can be done for both physical and virtual computer systems, it is much easier to utilise virtual systems as the restoration actions can be done in software.

The initial definition of a virtual machine is a software extraction that looks like a computer system's hardware (Rosenblum 2004). This definition is called hardware-level virtualization. As the hardware is virtualised, the virtual machine will run all software, operating system and applications written for the hardware being emulated (Rosenblum 2004). Virtual machines are also isolated from other virtual machines as well as the physical machine that they run on, ensuring that any problems with the software to be run is contained purely within the virtual machine (Rosenblum 2004).

The two most popular virtual machine solutions are called VMWare Workstation/-Player created by VMWare Inc. and VirtualBox developed by Oracle. Both software packages are capable of emulating a full x86 based computer (i.e. a normal desktop pc). Both software packages provide remote access and can be controlled by a command line interface.

Oracle's VirtualBox has a feature called immutable disks, which essentially means that all disk writes are only saved temporarily and are discarded when the machine is re-booted. It does this by directing all writes to a differencing image, which is reset every time the machine is powered on (Oracle n.d.*a*). This means that the same hard disk image can be used for multiple virtual machines.

2.6 Delivery of Information

Because students are accessing the laboratory remotely there must be a means to provide the student with the information to use the laboratory. By far the most common way of providing information on the internet is by using a web page. Other methods include printing the information and sending it to the student or making the information available as an e-book so the student can print out the information themselves.

2.7 Remote Access

There is a number of protocols and computer software available that allows remote access. As one of the project aims is that the student is able to manipulate machines as if they were in physically in front of them, remote access solutions are focused toward 'Remote Desktop' solutions.

2.7.1 Remote Desktop Protocol

Remote Desktop Protocol (RDP) is a remote desktop protocol developed by Microsoft for Windows platforms, although many third-party cross-platform implementations exist. RDP is the base protocol used in Microsoft's remote desktop services (Microsoft 2012). Remote desktop services (and by extension RDP) allows for clients to connect to a virtual destop on a remote Machine. A RDP server runs on the remote machine and a RDP client runs on the user's machine. The RDP Client connects to the RDP server on the remote machine, and displays a virtual destop of the remote machine on the user's machine.

Oracle's VirtualBox has a feature called VirtualBox Remote Display Protocol (VRDP) via an extension to VirtualBox. VRDP is a backwards-compatible extension to Microsoft's RDP (Oracle n.d.*b*), allowing any RDP client to access a VirtualBox virtual machine remotely.

2.7.2 Virtual Network Computing

Virtual Network Computing (VNC) operates in a very similar way. VNC can also utilise a mirror driver and it allows you to work on a remote computer as if you were sitting in front of it (Hsiao 2009). But it can also create a virtual desktop that is not associated with any physical display (Bezroukov, D, 2009). It also uses the client-server model, but the software also runs on multiple platforms and in a web browser, this is good for remotely connecting to and from multiple operating systems and a client can even be embedded in a web browser (Bezroukov, D, 2009). Also a VNC session can be shared so that multiple users can access it (Bezroukov, D, 2009), this allows one person to use the remote desktop, and another person to supervise.

Chapter 3

System Design

3.1 Overview

This chapter covers the requirements analysis and the overall design of the remote access laboratory.

3.2 Requirements Analysis

Based on the project aims and the concepts researched in the literature review, a detailed requirements list was created. These influenced the overall design of the remote access lab.

3.2.1 Requirements

Laboratory

- Contains a supervisory lab program which monitors the lab
- Contains a web interface which the user access to configure the lab.
- Contains multiple virtual machines which are configurable for the student to perform experiments on
- A user configurable network which the user uses in the experiments

Supervisory Lab Program

- Monitors virtual systems to ensure they are still running and are still accessible.
- Must not be accessible by users.
- Continuously checks whether the lab state has changed.
- Can start and stop virtual machines depending on the lab state.
- Can start and stop the DHCP server depending on the state.
- Can reset the configurable switch when the lab state is changed
- Can tell the web interface the current lab state.

3.2.2 Physical Machines

- Must be capable of running at least one virtual machine.
- Must contain enough ethernet ports to accommodate the virtual machines (two per virtual machine).
- Must not be accessible by the user. This includes direct access or access through any of the virtual machines.
- Must be able to communicate with each other (in the case of multiple physical machines).
- One physical machine must have a DHCP server running on it.
- One physical machine must be running the lab program and the virtual machine containing the web interface.
- The physical machine running the lab program must have a serial connection.

Virtual machines

The virtual machines fill the role of the ”media/experiment server” concept outlined in the literature review.

- Must be accessible using the remote access tools built into the virtualization software
- Each virtual machine must have at least two ethernet ports
- The user should be able to configure the system as if he/she were in front of the physical machine
- The virtual machines must have installed all the necessary tools the user needs to configure the system in accordance with the experiments
- The virtual machines must be in a frozen state with all changes discarded when the virtual machine is rebooted
- The system must comprise of windows and Linux machines

Web interface

The web interface fills the role of the ”web server” concept outlined in the literature review.

- Runs in its own virtual machine.
- Allows users to select experiment.
- Provides instructions for the users to perform experiments.
- Allows users to set the state of the lab in accordance with the experiments.
- Communicates with the lab program when setting the state.

Experiments

- Based on ELE3915.
- Allows users to learn about the basics of computer networking.
- Allows users to create their own computer networks.
- Gives an introduction to the Linux Operating System.

- Allows users to learn about commonly used networking tools under Windows and Linux
- Allows users so set up services on a computer network

3.3 Laboratory Design

This section describes the overall design of the remote access lab as well as details the decisions made when designing the remote access lab.

3.3.1 Practical Experiments

The exercises performed in the lab have been derived from the exercises ELE3915 Electrical and Electronic Practice E, an existing practical course focused on computer networking. The purpose of this is to have the lab provide the same learning experience is the practice course. This will prepare students with the knowledge to perform the exercises when they attend the practical class, as well as assist with their learning with other networking classes. Due to the nature of the lab being remotely accessed, some exercises have been omitted. These include installing Linux in a virtual machine and using serial ports.

Using the ELE3915 Practice Book as a reference the following exercises have been created.

Static IPs The student creates a small network and manually assign IP addresses to each computer on the network.

Gateway and Routing The student will create multiple networks and create router that will route packets between them.

Wireshark Packet Analyser The student will learn how to use Wireshark to capture and analyse packets.

Introduction to the Linux Operating System The student will be introduced to the Linux operating system, including the filesystem, the command line interface and commonly used Linux commands.

Windows and Linux CLI networking tools The student will learn about commonly used networking tools used under Linux and Windows.

DHCP Server The student will set up a Dynamic Host Configuration (DHCP) server.

Windows File Sharing - NetBIOS Protocol The student will allow files and folders to be shared over the network using Windows' built in sharing functionality.

Unix-Windows File Sharing - Samba The student will allow files and folders to be shared over between Windows and Linux computers by setting up a Samba server.

Unix File Sharing - NFS: Network File System The student will share a folder between two Linux Machines by setting up a NFS server.

Apache Web Server The student will set up a web server.

Information on the exercises (such as background information and instructions) have also been derived from the ELE3915 Practical Book, as well as online instruction guides and software manuals, and the information has been modified and updated as needed, as well as adapted to suite a remote environment.

3.3.2 Operating System and Software Selection

To facilitate the ease of rebooting the computers the operating systems were installed in virtual machines. The two most commonly used virtualization programs available are VirtualBox and VMWare. VirtualBox will be used because it contains some very important features that will assist in the operation of the remote access lab. The first of this is the ability to freeze a hard drive image. This means than virtual machine that uses a frozen hard drive image, will have its changes discarded when the machine is turned off. The second feature is that VirutalBox has a built in RDP server (via an extension pack) which allows for easy integration into the USQ Remote Access Lab system, which also uses RDP. It also means that we do not have to run a RDP server within the virtual machine, and the supervisory machine does not have to connect to the virtual machine and check whether the RDP server is running. Instead it just has to check whether the virtual machine itself is running. VMWare is not viable for this lab as it does not support RDP nor can the hard drive image be frozen.

The experiments require using both Windows and Linux Operating Systems. As some experiments require communication between two Windows machines and communication between two Linux machines, no less than four virtual machines are required. Two machines are running Windows and two machines are running Linux. All the physical machines that the virtual machines that are running the virtual machines must be running Linux. This is because the lab program that monitors the virtual machines utilizes many Linux utilities to perform its task.

The distribution of Linux that is used in the remote access lab is CentOS version 6.4. CentOS is a free Linux operating system based upon Red Hat Enterprise Linux with all of the propriety components removed, leaving a completely free enterprise operating system. As the remote access lab will be operating in an enterprise environment, where stability, security and long term support are of the highest importance, an enterprise operating system, which uses highly tested and stable software, is highly desirable. Using an enterprise operating system ensures that the underlying software contains a minimum amount of bugs, minimum breakage of software when updating, and less downtime. Using a free operating system like CentOS removes the cost of purchasing a propriety operating system (e.g. Red Hat Enterprise Linux, upon which CentOS is based). The downside to this is that most free operating systems is the lack of real

Sheet1

Exercise	Description	Pre-Requisite	Min Systems Required	M1-L	M2-W	M3-L	M4-W	CGF-SW	Notes
static IP	students learn how to apply an IP address to an Ethernet interface	No IP, default switch state	2	✓	✓				
routing and multiple networks	students learn how to set up multiple networks and route packets between them	No IP, default switch state	2	✓	✓			✓	
Wireshark packet analyser	Students learn how to analyse packets using Wireshark	IP assigned, default switch state	1		✓				
Linux FS/CLI	students learn some basic Unix concepts, the FHS, using the terminal.	none	1	✓					
Windows/Linux CLI	students learn how to use commonly used command line networking tools under windows and Linux	none	1	✓	✓				
DHCP Server	students learn how to set up a DHCP server	No IP, default switch state	2	✓		✓			Windows assigns an arbitrary IP when it does not receive an IP from a DHCP server, this may cause confusion
NetBios	students learn how to set up a windows share	IP assigned, default switch state	2		✓		✓		
Samba	students learn how to set up a samba server	IP assigned, default switch state	2	✓	✓				
NFS	students learn how to set up a nfs server	IP assigned, default switch state	2	✓		✓			
Apache Server	students learn how to set up an apache server	IP assigned, default switch state	2	✓	-	-	-		

Page 1

time support. But as CentOS also derives its documentation from Red Hat's there is a lot of information to assist with support cases.

The version of Windows chosen was Windows 7 Professional as it was easiest version to obtain at the time of creating the virtual machines.

The physical machines are running CentOS 6 as their operating system. A Linux based operating system is used because a lot of the software that the lab program uses to function (such as grep, lua, and expect) are only available or more readily available under Linux.

The web server that is installed on the web interface is the Apache web server which is highly popular web server used in many servers on the internet.

A terminal emulator is also required to communicate with the configurable switch through a serial connection. In order to automate this communication a terminal emulator with scripting functionality is required. C-Kermit, originally written by Columbia University, provides this functionality.

3.3.3 Laboratory Network

The Laboratory consists of five virtual machines. One virtual machine runs the web server which hosts the web site containing the instructions for the experiments. The reason for this is simply that the remote access lab system at USQ is RDP based. By putting the web-interface on a virtual machine, the student access both the web-interface and the other virtual machines through the same system, instead of having to access them separately (e.g. accessing web-interface through a separate web site). The other virtual machines are used by the students to perform the experiments. The virtual machines can run on any number of physical hosts, the only restriction is that the virtual machine that is running the web server must be on the same physical machine that is running the script.

One of the host machines running the virtual machines must have a DHCP server installed and configured. This is because some of the experiments require that the virtual machines have IP addresses assigned to them, and some experiments require that the virtual machines do not have IP addresses. To make things easier the lab program does not connect to the virtual machines and change IP addresses based on the experiment. Instead it just starts and stops the DHCP server when required. It will be up to the student to determine the IP address of a particular virtual machine (this can also be part of the learning process). All host machines must be on the same network and are able of communicating with each other to allow the lab program to start and stop the virtual machines on any host machine.

All the ethernet interfaces of the virtual machines are bridged with the ethernet interfaces with the host, and the host ethernet interfaces will be connected to a configurable switch. This is required as one of the experiments requires the configuration of a switch to allow multiple networks. The bridging of the ethernet interfaces allows the virtual machines to connect directly to the switch and also allows the DHCP server running on one of the hosts to assign IP addresses to the virtual machines.

The Lab also utilises a configurable switch to virtualize creating multiple networks through its VLAN functionality. There is only one exercise that requires creating multiple networks, but the switch needs to be set back to its default setting afterwards. Resetting the configurable switch poses a problem as the student requires administrators access to the switch in order to configure multiple virtual networks but they could also change the settings to restrict access, which makes resetting the configurable switch through its graphical user interface or configuring a SSH user impractical. To overcome this manufacturers implement a console port on their switches. A console port provides command line access to the switch regardless of what the current settings are on the switch. The console port is designed to connect to a serial computer terminal which is usually physically secured on location with the switch. In the lab however the console port is connected to a serial port on the PC running the lab program and a terminal emulator is used to communicate with the switch.

Taking into account the above requirements, the schematic in figure 4.1 was created.

3.3.4 Web Interface and Lab Program

The students will access the lab using USQ's Remote Access Lab services. The student will connect remotely to the machine running the web server which automatically starts a web browser showing the web interface. Here they can set the state of the lab as

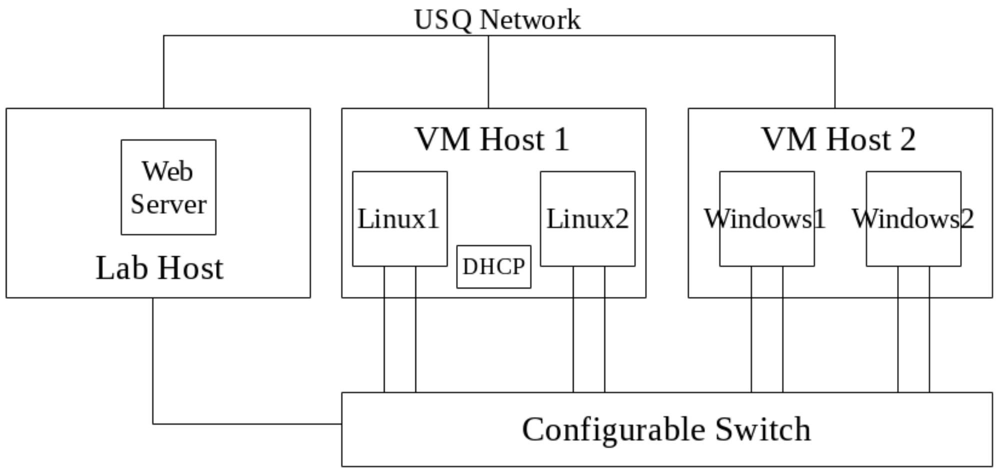


Figure 3.1: Lab Schematic

well as find the instructions for each of the experiments. When the user requests the lab state to be changed, the web interface saves a file containing the new state of the lab into a shared folder between the host and the virtual machine. The lab program running on the host continually for this file and when it finds it, reads it, and then resets the lab into new state.

The reason the web server is running in it's own virtual machine is because the remote access to the lab is RDP based. By putting the web server in a virtual machine students can connect to the web interface just like they would connect to any of the configurable machines, allowing for a consistent experience, rather than having students access the web interface separately from the lab.

The laboratory program, which is simply a script, continuously checks the machines in the lab to make sure that the currently required virtual machines and DHCP server are running. The lab program is written an a popular, lightweight and extensible scripting language called Lua. Writing the program as a script is preferable as it is not needed to compile the source code and it is easier to execute other programs. Scripting languages such as Lua also provide superior string handling abilities to compiled languages which is used extensively within the lab program.

The lab program uses SSH to connect to the machines hosting the virtual machines and is able to start, stop and check the virtual machines and DHCP server. OpesSSH is a tried and true program used to open command line sessions on other machines. The lab program also executes another script which which controls a serial connection to the console port of the configurable switch, allowing it to be factory reset when required.

3.4 Resource Requirements

The following gives a detailed view of the hardware and software requirements for the remote access lab based on the design detailed above.

3.4.1 Hardware Requirements

The hardware requirements for the project are as follows:

- Two desktop computers capable of running at least 2 virtual machines.

  - A CPU powerful enough to run at least 2 virtual machines (preferably multi-core with one core per VM).
  - At least 3 GB of RAM (1 GB for each VM and a spare 1 GB for the host).
  - Two ethernet cards/usb ethernet adaptors per VM.
- A configurable switch.
- A serial expansion card/usb serial adaptor.
- Network cables.

3.4.2 Software Requirements

The software requirements for the project are as follows:

- Each desktop machine must be running Linux as it's host operating system.
- The desktop running the lab program must have Lua installed (to run the lab program).

- The desktop running the lab program must have c-kermit installed (to connect to the configurable switch).
- Each desktop must have OpenSSH installed and configured (to allow the lab program to connect and check the status of the virtual machines).
- Each desktop must have VirtualBox installed.
- Windows Virtual Machine
  - Running a Windows Operating system.
  - Have Wireshark packet capture program installed.
- Linux Virtual Machine
  - Running a Linux Operating system.
  - Have Samba installed
  - Have NFS installed

3.5 Risk Analysis

Being an autonomous system that is accessed remotely the remote access lab poses little physical risk to operators and even lesser risk to users. The following outlines the risks to people who operate and use the laboratory. The biggest risk involves security. Since this lab will be connected to the USQ network there are security implications especially as users will have full control over the certain parts of the lab. Steps need to be taken to ensure that neither the lab computers or the USQ network become compromised by a third party. For the details of the risk analysis see appendix B.

3.6 Consequential Effects

3.6.1 Sustainability

The only sustainability issues for the remote access laboratory is making sure that the lab is maintained properly, hardware and software is up to date. Usage of off-the-shelf products makes maintainability easy. Also because of the high availability of computer hardware it is relatively simple to upgrade or replace parts. Smart design and making the system modular can also ease the maintence and modifiability of the system to include additional features or experiments should the administrator see fit.

3.6.2 Safety

All equipment used in implementing the lab are required to conform to Australian and international standards. As all components of the lab are off-the-shelf products they automatically come under mandatory standards compliance in order to be sold in Australia.

As the system operates entirely online there is very little danger posed to the user other than strain caused by using a computer.

3.6.3 Ethical Considerations

The main ethical consideration is that lab will be used to educate students. This entails the inclusion of human beings in the research project. For this reason Human Ethics Clearance will be required as per the guidelines set in the National Statement on Ethical Conduct in Human Research 2007.

Chapter 4

Implementation

4.1 Overview

This section outlines the details of the implementation of the remote networking lab using the requirements and design. Two desktop PCs and a configurable switch have been provided by USQ for the purpose of implementing the remote networking laboratory.

4.2 Physical Machines

Each PC was modified to include 5 ethernet adaptors as each PC hosts two of the virtual machines that the student will perform experiments. Four of the ethernet interfaces are connect to the configurable switch. The hard drives were then wiped and CentOS 6 was installed. Each PC had an OpenSSH server configured and restricted to key-only authentication, and SSH keys were generated and given to the PC running the lab program. Both machines also have VirtualBox installed on them. All the bridged ethernet interfaces except the one the DHCP server is listening on, are configured to not have any IP address assigned. This disallows any connection from the virtual machines to the physical machines. The remaining ethernet interfaces are used by the student to connect to the lab and connected to the USQ network. The physical machine running the web interface and the lab program also had Lua, Expect, c-kermit and a DHCP server installed. This machine was also connected directly to the configurable switch's console port.

4.3 Virtual Machines

There are five virtual machines in total in the lab: one for the web interface and four for the student to experiment on. Of the four experiment machines two are running CentOS and the other two are running Windows 7. Both operating systems are installed with the default settings as well as any extra necessary software is installed as per the experiment requirements. The virtual machines are named as follows: CentOS_6-1, CentOS_6-2, Windows_7-1 and Windows_7-2. Each virtual machine is configured to use one CPU core. Each of the Linux virtual machines are configured to use 512MB of RAM while the two Windows virtual machines use 1GB of ram each. Each virtual machine was given two ethernet interfaces and they were bridged to the ethernet ports on the physical machine that were connected to the switch.

In order to easily identify which virtual machine is connected to which port on the switch, connections were between each of the machines were organised and fixed.

Physical machine	Virtual Machine	Switch Port
PC1-eth1	CentOS_6-1-eth0	1
PC1-eth2	CentOS_6-1-eth1	2
PC1-eth3	CentOS_6-2-eth0	3
PC1-eth4	CentOS_6-2-eth1	4
PC2-eth1	Windows_7-1-LAN1	5
PC2-eth2	Windows_7-1-LAN2	6
PC2-eth3	Windows_7-2-LAN1	7
PC2-eth4	Windows_7-2-LAN2	8

Table 4.1: PC-VM-Switch ethernet connections

VirtualBox has a feature called immutable hard drives which means that the hard drive image is frozen and any changes that are made are discarded. Every time the virtual machine is run with an immutable hard drive image, a ”difference image” is created that temporarily contains the changes made to the virtual machine, thus preserving the original image. This is done automatically by VirtualBox. There are two hard drive images used in the lab. One hard drive image contains a CentOS 6 installation and is shared between the two Linux virtual machines. The two Windows virtual machine likewise share a single hard drive image. When each operating system was installed the hard drive image was not frozen attached to only one virtual machine. After the operating system was installed, updated and had the necessary software installed, the hard drive image was detached. The image was then made immutable using VirtualBox's media manager then reattached to the virtual machines that used that image.

Whenever the virtual machine is update the hard drive images have to be detached from their virtual machines, all difference images deleted (done via VirtualBox's media manager), the image set to normal (unfreezing it) then reattached to one of the virtual machines. Once the virtual machine is updated the hard drive image can be made immutable again.

VirtualBox also provides a RDP server via the Oracle VM VirtualBox Extension Pack, and extension to VirtuasBox freely available from the VirtualBox website. The settings for the RDP server are located within the display settings for each virtual machine. The web interface is configured to use port 37000 on the first physical machine and CentOS_6-1 and CentOS_6-2 use ports 37001 and 37002 respectively on the first physical machine. Windows_7-1 and Windows_7-2 are configured to use 37001 and 37002 respectively on the second physical machine. The physical machines have also had their firewall's configured to accept connections on these ports.

4.4 Web Interface

4.4.1 Virtual Machine

The web interface is simply a web server running in its own virtual machine with a web browser connected to that web server. The virtual machine runs a cut down version of CentOS (much of the user software is not installed) which is configured to automatically run the web browser on boot. The web browser that CentOS comes with by default is Mozilla Firefox. The apache web server was also installed on the web interface's virtual machine. In order to allow shared folders to be enabled on the virtual machine, VirtualBox guest additions were installed on the virtual machine. This however required additional software such as gcc, make and dkms which were also installed. All software except the VirtualBox guest additions were installed through

CentOS' package manager.

Mozilla Firefox has a 'kiosk mode' extension installed to make the web browser full screen as well as limit what the user is allowed to do with the web browser. The extension is called mKiosk which is freely available on the internet and is configured to automatically enter fullscreen mode and the menu bar is removed. The virtual machine hosting the web interface is also configured to automatically log in. This is done by adding the options AutomaticLogin=user and AutomaticLoginEnable=True to the daemon section of /etc/gdm/custom.conf. The virtual machine is also configured to start Firefox automatically after login. In order to protect the web interface from being tampered and for easy maintenance the source files for the web interface are not stored within the virtual machine. Instead they are stored on the host machine.

The web interface's virtual machine has there folders shared with the host machine. They are called: project, state and feedback and are located on the physical machine in /home/user/shared/ and /media/ on the web interface's virtual machine. All shared folders are owned by the vbox group and the apache user has been made part of the vbox group to allow it access to the shared folders. The user that the student runs the web browser as is not part of the vbox group and therefore cannot access the shared folders.

The project folder contains the source files for the web interface. It is configured within VirtualBox to be read-only meaning that it can not be modified from the virtual machine. Also the web interface can be modified easily be simply changing the source files on the host machine without having to boot into the virtual machine. The project folder is mapped to /media/sf_project/ within the virtual machine and the apache server in the virtual machine has been configured to use /media/sf_project as it's document root.

The state folder is folder is used by both the web interface and the lab program to communicate the current state of the lab between them. On the virtual machine the shared folder is mapped to /media/sf_state/. When the student sets a new state for the lab the web interface writes a new file called state to to the state folder which is read by the lab interface. The lab interface also writes a file called "status" which is read by the web server and displayed on the web interface.

The final folder called feedback is used by the web server to write any feedback the student writes about the remote access lab. The feedback page will be passed the exercise number for which exercise the feedback is for using PHP. The feedback for each of the exercises is stored in their own text file in the feedback folder and named to identify which exercise the feedback corresponds to (e.g. exercise1.txt). Each time feedback is left by a student it is appended to the file corresponding with the respective exercise. Each line of a feedback file corresponds to one student.

4.4.2 Source Code

All the source code for the web interface is written in xhtml and PHP functionality is used to read, write and display the lab state as well as save the student feedback. There is very little styling and the content is laid out in a consistent way that is easy for the student to read. To make writing the source code easier a macro processor called m4 was used. When using a macro processor you define a macro (or symbol) and its associated text. When the macro processor is run on a text file it searches for those macro and replaces them with their defined text. This is highly useful for things like custom PHP functions, where you define a list of functions and simply put the macro in the source code. This way the functions only have to be defined once and the macros placed in all the source files, instead of having to copy the functions into every source file.

```
define(‘XML_HEADER’, ‘<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
           "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">’)dnl
```

Figure 4.1: Defining a xhtml header as a macro

There are two folders associated with the source code for the web interface: project and project_build. The directory project_build contains the source code before the macro processor has processed. The macro processor m4 is then run over all these files and output into the project folder. The resulting output is then used as the document source for the web interface. To automate all this a makefile is used to automatically run m4 on all file as well as copy and extra files (such as images) to the project folder. Simply running the utility ”make” inside the project_build folder will do this.

m4 macro processor file	xhtml/PHP source file	Description.
defines.m4		Contains PHP function definitions as well as xhmtml header.
	index.php	Dummy file that redirects to exercises.php.
exercises.m4	exercises.php	Displays the server status and contains the list of experiments and feedback links.
state.m4	state.php	Outputs the state of the lab to be read by the lab program.
feedback.m4	feedback.php	Feedback for for the exercises.
exercise1.m4	exercise1.php	Exercise 1
exercise2.m4	exercise2.php	Exercise 2
exercise3.m4	exercise3.php	Exercise 3
exercise4.m4	exercise4.php	Exercise 4
exercise5.m4	exercise5.php	Exercise 5
exercise6.m4	exercise6.php	Exercise 6
exercise7.m4	exercise7.php	Exercise 7
exercise8.m4	exercise8.php	Exercise 8
exercise9.m4	exercise9.php	Exercise 9
exercise10.m4	exercise10.php	Exercise 10

Table 4.2: Description of Source Code Files

The web page exercises.php displays a list of the exercises the student can perform as well as feedback links for each exercise and also shows the current state of the lab. Each exercise web page shows the required lab state need to perform the exercise as well as a link to set the lab state. The link opens "state.php" and passes it the required state (in the form "D1234" or similar). The PHP code then outputs this to the state folder to be read by the web interface, and then the PHP function tells the web browser redirects back to the exercise page.

4.5 Lab Program

The script is named ”ral.lua” and is simply run by passing it as an argument to the lua interpreter (by running lua ral.lua). The actual laboratory script resides in /home-/user/shared/lab/.

Several aspects of the lab such as the names of the virtual machines, their locations, the directory containing state information, and the location of the DHCP server, were implemented as variables in the lab script. This allows the virtual machines and DHCP server to be located on any physical machine or if the IP address of a particular machine changes. Simply change the variable containing the name of the virtual machine and the IP address of the machine its sitting on within the script. This also allows the names and locations of folders and files to be changed easily in the script in the event the actual folders on the machines do change.

```
--predefined variables
VMMACHINES = {}
VMMACHINES[1] = {"127.0.0.1", "CentOS_6-1", "STOPPED"}
VMMACHINES[2] = {"127.0.0.1", "CentOS_6-2", "STOPPED"}
VMMACHINES[3] = {"10.1.0.228", "Windows_7-1", "STOPPED"}
VMMACHINES[4] = {"10.1.0.228", "Windows_7-2", "STOPPED"}
webip = "127.0.0.1"
webname = "WebServer"
quit = false
dhcpServer = "127.0.0.1"
labLanIp = "192.168.11.1"
tpScript = "tplink.ksc"
currentState = ""
switchTime = 0
stateFile = "/home/user/shared/state/state"
statusFile = "/home/user/shared/state/status"
errorLog = "/home/user/shared/lab/error"
--stateFile = "/var/www/state"
numMachines = 4
```

Figure 4.2: Variables used in the laboratory script

4.5.1 DHCP server

Since some experiments require the virtual machines to have IP addresses assigned a simple DHCP server was used to automatically assign IP addresses to the virtual machines. The DHCP server runs on the same physical machine that the lab script runs on. It is configured to give IP addresses in the range of 192.168.11.2 to 192.168.11.10. The physical machine has eth1 (the fist ethernet interface bridged with the virtual machine) with a static IP address of 192.168.11.1 to facilitate the DHCP server. The server is also configured to only listen on eth1 to make sure that the only requests come from the virtual machines.

4.5.2 Communication with Web Interface

As stated above a shared folder is used to communicate between the web interface and the lab program. When the use sets a new state within the lab the web server will output a file called "state" into the /media/sf_state/folder on the web interface's virtual machine. On the host this folder's path is /home/user/shared/state. The lab program continually checks for this file and if it is detected open it, read the lab state from the file, then deletes the file to avoid setting the state repeatedly. Whenever this file appears it signifies that the state has changed and the lab program proceeds to shutdown the DHCP server and all the virtual machines except the web interface.

The lab state is determined by five characters: 'D', '1', '2', '3' and '4'. These correspond to the DHCP server, CentOS_6-1, CentOS_6-2, Windows_7-1 and Windows_7-2 respectively and must be present in the state file and separated by spaces. The lab program reads a limited number of characters from the file, separate individual words by spaces, puts them into an array, then scans through the array and checks whether each element is a single character that matches 'D', '1', '2', '3' or '4'. Every other word or character is ignored. The same format is used when communicating the current lab status to the web interface. The lab program will continuously check the lab state and output a file called "status" to the state folder, which is read by the web server every time the page is refreshed.

4.5.3 Starting and stopping virtual machines/DHCP server

The script is designed to be able to start and stop the DHCP server and the virtual machines on any of the physical machines. In order to do that the script must be able to remotely connect to any of the physical machines. An expect script was used to provide this functionality. Expect is a program used to automate control of interactive CLI applications by reading input and send output just like a human would using a keyboard. The expect script allows for an automatic SSH session to a remote computer. The following code is an example which will connect to a remote computer and start a virtual machine.

```
set timeout 5
spawn ssh user@192.168.11.10\n
expect {
  "*\]" {
    send "DISPLAY=:0 vboxmanage startvm CentOS_6-1\n"
    expect "*\]"
    send "exit\n"
    expect eof
    exit 0
  }
}so
exit 1
```

Figure 4.3: Expect script example

Using the string handling functionality in Lua the IP address and the name of the virtual machine to be started can be changed to whatever is defined in the script and Expect can be run multiple times, each with a new IP address and VM name. There are multiple Expect scripts for different functionality. Expect can also take a string as it's script input, allowing for the Expect scripts to be embedded into the lab program. Expect scripts are also used to check the lab state by looking through the process list of the host machine (using the command line application ps) and looking for an entry for the virtual machine that it's currently checking (using the command line application grep).

4.5.4 Resetting the configurable switch

A small script that can be read and executed by c-kermit is used to automatically reset the configurable switch. This script opens a connection over a serial cable to the configurable switch and uses the switch's command line interface to initiate a factory reset. The script is executed whenever the lab state changes. As the configurable switch takes about thirty seconds to reset a timer has been written into the lab program in order to avoid attempting to reset the configurable switch while it's already being reset. The script is called tpscript.ks and is located in the same folder as the laboratory script.

```
set line /dev/ttyS0
set flow-control none
set carrier-watch off
set speed 38400
lineout
input 5 TL-SG3216>
lineout enable
input 5 TL-SG3216#
lineout reset
input 5 (Y/N)
lineout y
exit
```

Figure 4.4: Script used by c-kermit

Chapter 5

Testing

5.1 Overview

This chapter describes how the networking remote access lab was tested as well as the requirements needed to pass each test. As each port of the remote access lab was tested, any errors that appeared during testing were fixed and the lab retested.

5.2 Unit Testing

The following shows what each component of the lab must be able to do when undergoing testing.

5.2.1 Web Interface

The virtual machine containing the web interface was run by itself. Communication between the web interface and the lab program was simulated simply by writing a new file to the shared directory. Likewise checking the output of the web interface (state, feedback) was done by simply opening the shared folder and checking the files. The following outlines the requirements for the web interface to pass testing.

- The web interface must allow students to choose an experiment

- The web interface must store student feedback in the correct shared folder.
- The web interface must store feedback for each exercise in a separate file.
- Each file containing feedback must be properly named to differentiate between exercises (e.g. exercise1.txt)
- The web interface must allow for easy navigation between is web pages.
- The web interface must correctly read the state of the lab from the 'state' file in the shared folder.
- The web interface must accurately show the current state of the lab.
- Each experiment must show the correct server requirements.
- The web interface must correctly set the new state of the lab based on the experiment requirements.
- The web interface must accurately output the new state of the lab into the correct shared folder.
- The content on the web interface must be laid out clearly and easy to comprehend.
- The experiments' instructions must be clear and easy to follow.

5.2.2 Physical Machines

Testing the physical machines was simply done by checking that the required software was install, configured and functioning properly.

- The hardware and software must meet the resource requirements.
- The physical machines must have SSH running and configured correctly to allow the lab program to connect to any physical machine.
- SSH must be configured to only allow key-based authentication.
- The PC running the lab program must contain all the SSH authentication keys of the other physical machines.
- The software of the physical machines must be easily upgradable.

- The physical machine running the lab program must be connected to the configurable switch through a serial cable.

5.2.3 Virtual Machines

Testing the virtual machines was simply done by checking that the required software was install, configured functioning properly.

- The virtual machines must have the software required to perform the experiments.
- The virtual machines must be configured properly in order to perform the experiments.
- The virtual machines must to discard changes made to in when restarted.
- The virtual machines must be capable of being updated.

5.3 Integration/System Testing

This test covered the lab program, virtual machines, DHCP server, and web interface together. Unfortunately the lab was not integrated with USQ's remote access lab service so integration testing did no cover that part of the lab.

- Setting the state in the web interface must be picked up by the lab program, and the relevant machines must be started.
- The lab program must be able to correctly set the lab state
- The lab program must be able to parse the state information from the file output by the web interface
- The lab program must be able to open a SSH session to other virtual machines
- The lab program must be able to start and stop virtual machines on any physical machine
- The lab program must be able to start and stop a DHCP server on any machine.

- The lab program must be able to tell the web interface the current state of the lab.
- If the lab program crashes for any reason it must display an error message
- The lab program must output its current operation (checking state, starting stopping machines, etc).
- The lab program must be able to reset the configurable switch

5.4 Acceptance Testing

Acceptance testing was done by performing the actual experiments on the remote access lab.

5.4.1 Laboratory and Experiments

- The experiments must allow students to learn about computer networking.
- The experiments must be completable.
- The content of the experiments must be comparable to existing computer networking courses.
- The content of the experiments must be comparable to what is used in the industry.

5.5 Student Evaluation

Student evaluation is focused on:

- How usable is the web interface, can it be navigated easily?
- Are the experiments understandable/doable, what parts need clarification?
- What did the students learn from the exercises?
- What experiments would the student like to see extended.

- What experiments would the student like to see included.

Student feedback is important as the operating goal of the remote access lab is to provide a practical learning experience. Information gathered from the students help find parts of the lab which need modification and improvement. Unfortunately due to time constraints student evaluation was not conducted. However all other tests outlined above were completed.

Chapter 6

Conclusion

6.1 Overview

This chapter outlines the further work that can be continued on the remote access lab outside the scope of this project.

6.2 Further Work

Unfortunately the lab wasn't able to be integrated with the USQ's remote access labs or able to be tested with students. Future work for this project will prioritize these two tasks. Further work after that entails receiving feedback from students on the lab and further improving it based on that feedback. Other work would include extending the existing experiments, adding additional experiments and possibly additional hardware such as wifi adaptors.

6.3 Summary

Currently the University of Southern Queensland utilises practical courses in order to give students practical experience in computer networking. However external students have to travel to USQ within a limited time to attend this practical course as there has been no means of giving external students practical experience remotely. In order to provide a practical experience for computer networking to external students, a remote access lab which give students this practical experience was created.

This dissertation detailed the design and implementation of a remote access laboratory capable of allow students to learn about the concepts of computer networking. It is hoped that the laboratory detailed in this dissertation will be put into use as part of the curriculum at USQ. It is also hoped that this lab will be continue to be developed and improved in the future, providing a practical learning experience for future students.

References

Alves, G., Gericota, M., Silva, J. & Alves, J. (2007), Large and small scale networks of remote labs: a survey, *in* 'Advances on remote laboratories and e-learning experiences', University of Deusto, Bilbao, pp. 15–34.

Andreas Bhne1, N. F. & Wagner, B. (2002), *Self-directed Learning and Tutorial Assistance in a Remote Laboratory*, Interactive Computer Aided Learning Conference.

Australia, E. (2010), 'Code of Ethics', `http://www.engineersaustralia.org.au/sites/default/files/shado/About%20Us/Overview/Governance/codeofethics2010.pdf`.

Berzoukov, D. (2009), 'VNC The Essential Sysadmin Tool', `http://www.softpanorama.org/Xwindows/vnc.shtml`.

Bright, C., Lindsay, E., Lowe, D., Murray, S. & Lui, D. (2008), *Factors that impact learning outcomes in Remote Laboratories*, World Conference on Educational Multimedia, Hypermedia and Telecommunications (EDMEDIA) 2008.

Chang, R. (2004), *Teaching Computer Networking with the Help of Personal Computer Networks*, The Hong Kong Polytechnic University.

Chen, C. (2003), 'A Constructivist Approach to Teaching: Implications in Teaching Computer Networking', *Information Technology, Learning, and Performance Journal* **21**(2).

Gomes, L. & Bogosyan, S. (2009), 'Current Trends in Remote Laboratories', *IEEE Transactions on Industrial Electronics* **56**(12), 4744.

Gomes, L. & Garca-Zuba, J. (2007), Preface, *in* 'Advances on remote laboratories and e-learning experiences', University of Deusto, Bilbao, pp. 9–11.

Hsiao, P. (2009), *Virtual Remote Desktop - Implementation*, Tatung University.

Hutchinson, D. & Bekkering, E. (2009), ‘Using Remote Desktop Applications in Education’, *Information Systems Education Journal* **7**(13).

Microsoft (2012), ‘Remote Desktop Services’, `http://technet.microsoft.com/library/hh831447.aspx`.

Oracle (n.d.*a*), ‘VirtualBox - Chapter 5. Virtual Storage’, `http://www.virtualbox.org/manual/ch05.html`.

Oracle (n.d.*b*), ‘VirtualBox - Chapter 7. Remote Virtual Machines’, `http://www.virtualbox.org/manual/ch07.html`.

Rigbey, S. & Dark, M. (2006), *Designing a Flexible, Multipurpose Remote Lab for the IT Curriculum*, SIGITE ’06 Proceedings of the 7th conference on Information technology education.

Rosenblum, M. (2004), ‘The Reincarnation of Virtual Machines’, *Queue - Virtual Machines* **2**(5), 34–40.

Sarkar, N. (2006), ‘Teaching Computer Networking Fundamentals Using Practical Laboratory Exercises’, *IEEE Transactions on Education* **9**(2).

Yoo, S. & Hovis, S. (2007), *Remote Access Networking Laboratory*, Middle Tennessee State University.

Appendix A

Project Specification

University of Southern Queensland

FACULTY OF ENGINEERING AND SURVEYING

ENG4111/4112 Research Project
PROJECT SPECIFICATION

FOR:	Keith DICKMANN
TOPIC:	NETWORKING LABORATORY DESIGN AND INSTALLATION
SUPERVISORS:	Alexander Kist
PROJECT AIM:	To design and implement a Remote Access Laboratory (RAL), capable of allowing external students to learn the basics of computer networking, providing a hands-on practical learning experience while being completely remote.

PROGRAMME: (Issue C, 26 March 2013)

1. Undertake a basic requirements analysis and establish potential experiments and learning outcomes of the experiments.

2. Undertake a comprehensive literature review covering all aspects of this project including computer networking, network management, automatic system restoration, and remote interfacing.

3. Design a networked system consisting of a configurable switch and multiple virtual systems, that allows a user to manipulate the network configuration and the computers on the network, and can be reset to a known default state when needed.

4. Build and test the system.

5. Create a number of practical experiments that will enable students to comprehend networking concepts and learn about network configuration.

6. Trial the experiments and evaluate the practicality, ease of use, and reliability of the system and make modifications as necessary.

7. Undertake trials with students to evaluate the system and experiments.

8. Document the project and write an academic dissertation on the research.

As Time Permits:

9. Modify the system to include more hardware, and allow multiple users access at the same time.

10. Include additional experiments which introduces other commonly used networking hardware.

11. Include additional experiments which introduce commonly used system administration tools.

AGREED: ___________________(Student) ___________________(Supervisor)

____ / ____ / ____ ____ / ____ / ____

Appendix B

Risk Assessment

The risk assessment is split into three parts:

- Risk to the Student/User
- Risk to the Administrator/Developer
- Security Risks

B.1 Risks to the Student/User

Hazard Description Repetitive stress injury using keyboard/mouse

Number of People 1

Parts of the Body hands

Injury Level low

Likelyhood low

Actions to Reduce Risk force the user to rest

B.2 Risk to the Administrator/Developer

Hazard Description Electrocution via short/faulty wiring

Number of People 1

Parts of the Body hands

Injury Level high

Likelyhood low

Actions to Reduce Risk keep equipment away from water, keep equipment disconnected from mains power when working on it

---

Hazard Description Injury from dropping equipment

Number of People up to 3

Parts of the Body legs, feet

Injury Level low, medium

Likelyhood low

Actions to Reduce Risk wear suitable clothing, minimizing chance that equipment will fall

---

Hazard Description Cuts/abrasions from assembling/upgrading/maintaining computers

Number of People 1

Parts of the Body hands, fingers

Injury Level low

Likelyhood low

Actions to Reduce Risk

B.3 Security Risks

Risk Description User tries to compromise the lab from the configurable computers

Actions to Reduce Risk Isolate the configurable machines from the physical machines as much as possible

Risk Description User tries to compromise using other techniques

Actions to Reduce Risk harden the lab as much as possible. Remove unneeded programs, setup firewalls, etc.

Risk Description Unauthorised access

Actions to Reduce Risk Access will be restricted to USQ students who have been granted RAL access. Use USQ student authentication.

Risk Description User tries to disrupt/damage configurable computers

Actions to Reduce Risk Make the configurable computers virtual and have them reset

---

Risk Description User tries to disrupt/damage lab interface

Actions to Reduce Risk Isolate the lab interface from the rest of the lab as much as possible.

Appendix C

Supervisory Script Source Code

C.1 ral.lua

```
-- explode(seperator, string)
function explode(d,p)
    local t, ll
    t={}
    ll=0
    if(p == nil) then return {} end
    if(#p == 1) then return {p} end
    while true do
        l=string.find(p,d,ll,true) -- find the next d in the string
        if l~=nil then -- if "not not" found then..
            table.insert(t, string.sub(p,ll,l-1)) -- Save it in our array.
            ll=l+1 -- save just after where we found it for searching next time.
        else
            table.insert(t, string.sub(p,ll)) -- Save what's left in our array.
            break -- Break at end, as it should be, according to the lua manual.
        end
    end
    return t
end
--------------------------------------------------------------------------------


--check whether a file exists
function fileExists(name)
   local f=io.open(name,"r")
   if f~=nil then io.close(f) return true else return false end
end


--------------------------------------------------------------------------------
--check whether a virtual machine is running
function checkMachine(host, name)
    expectScript = [[
set timeout 2
```

```
spawn ssh user@]]..host.."\n"..[[
expect {
  "*\]" {
    send "ps aux | grep \[V\]irtualBox | grep ]]..name..[[\n"
    expect {
      "VirtualBox" {
        expect {
          "]]..name..[[" {
            send "exit\n"
            expect eof
            exit 0
          }
        }
      }
    }
  }
}
exit 1]]
    os.execute("echo '"..expectScript.."' > expectScript")
    return os.execute("expect expectScript && rm -rf expectScript")
    --return os.execute("expect -c '"..expectScript.."'")
end

--------------------------------------------------------------------------------
function vboxmanage(host, name, action)
    if action == "start"
    then
        arg = "startvm "..name
    else
        arg = "controlvm "..name.." poweroff"
    end
    expectScript = [[
    set timeout 5
    spawn ssh user@]]..host.."\n"..[[
    expect {
      "*\]" {
        send "DISPLAY=:0 vboxmanage ]]..arg..[[\n"
        expect "*\]"
        send "exit\n"
        expect eof
        exit 0
      }
    }
    exit 1]]
    return os.execute("expect -c '"..expectScript.."'")
end

--------------------------------------------------------------------------------
function checkdhcpd(host)
    expectScript = [[
set timeout 2
spawn ssh user@]]..host.."\n"..[[
expect {
  "*\]" {
    send "ps aux | grep \[d\]hcpd\n"
    expect {
      "dhcpd" {
        send "exit\n"
        expect eof
        exit 0
      }
    }
  }
}
exit 1]]
    os.execute("echo '"..expectScript.."' > expectScript")
    return os.execute("expect expectScript && rm -rf expectScript")
    --return os.execute("expect -c '"..expectScript.."'")
end
```

```
--------------------------------------------------------------------------------
function dhcpd(host, action)
    expectScript = [[
set timeout 5
spawn ssh user@]]..host.."\n"..[[
expect {
  "*\]" {
    send "su -c \"/etc/rc.d/init.d/dhcpd ]]..action..[[\"\n"
    expect {
      "Password:" {
        send "raladmin\n"
        expect "*\]"
        send "exit\n"
        expect eof
        exit 0
      }
    }
  }
}
exit 1]]
    return os.execute("expect -c '"..expectScript.."'")
end

--[[---------------------------------------------------------------------------------
function checkExercise1()
    print("checking exercise 1")
    checkString = ""
    if os.execute("ifconfig eth1 192.168.0.50") ~= 0
    then
        os.execute("echo 'Error: unable to set ip address to eth1 while checking exercise
    end
    local r = os.execute("nmap -O --max-os-tries 1 192.168.0.101 | grep Linux")
    checkString = checkString..tostring(r).." Linux Machine with an IP of 192.168.0.101;"
    local r = os.execute("nmap -O --max-os-tries 1 192.168.0.102 | grep Linux")
    checkString = checkString..tostring(r).." Linux Machine with an IP of 192.168.0.102;"
    local r = os.execute("nmap -O --max-os-tries 1 192.168.0.103 | grep Windows")
    checkString = checkString..tostring(r).." Windows Machine with an IP of 192.168.0.103
    local r = os.execute("nmap -O --max-os-tries 1 192.168.0.104 | grep Windows")
    checkString = checkString..tostring(r).." Windows Machine with an IP of 192.168.0.104
    os.execute("echo '"..checkString.."' > "..checkDir.."exercise1")
end--]]
--------------------------------------------------------------------------------
--program body

--predefined variables
VMMACHINES = {}
VMMACHINES[1] = {"127.0.0.1", "CentOS_6-1", "STOPPED"}
VMMACHINES[2] = {"127.0.0.1", "CentOS_6-2", "STOPPED"}
VMMACHINES[3] = {"10.1.0.228", "Windows_7-1", "STOPPED"}
VMMACHINES[4] = {"10.1.0.228", "Windows_7-2", "STOPPED"}
webip = "127.0.0.1"
webname = "WebServer"
quit = false
dhcpServer = "127.0.0.1"
labLanIp = "192.168.11.1"
tpScript = "tplink.ksc"
currentState = ""
switchTime = 0
--checkDir = "/home/user/shared/check/"
--checkFile = checkDir.."tocheck"
stateFile = "/home/user/shared/state/state"
statusFile = "/home/user/shared/state/status"
errorLog = "/home/user/shared/lab/error"
--stateFile = "/var/www/state"
numMachines = 4

--[[check for root user
if os.execute("whoami | grep root") ~= 0
then
    print("this script mus be run as root")
    os.exit()
end
--]]
os.execute("touch "..stateFile)
while 0 == 0
do
----check for new state
    print("----------checking for new state----------")
    if fileExists(stateFile) == true
    then
        file = io.open(stateFile, "r")
        state = file:read('*line')
        print("new state found: ", state)
        stateArray = explode(" ", state)
        file:close()
        os.execute("rm -f "..stateFile)
        print("\n")
        print("----------stopping current state----------")
--        os.execute("rm -rf "..checkDir.."*")
    ----stop dhcp server
        dhcpd(dhcpServer, "stop")
    ----stop virtual machines
        for i = 1, numMachines
        do
            vboxmanage(VMMACHINES[i][1], VMMACHINES[i][2], "stop")
        end
    ----reset switch
        if os.time() - switchTime > 35
        then
            os.execute("kermit "..tpScript)
            switchTime = os.time()
        else
            print("switch already in the process of resetting")
        end
        print("\n")
    ----start servers
        print("----------setting new state----------")
--        os.execute("ifconfig eth1 "..labLanIp)
        if table.getn(stateArray) ~= 0
        then
            for i = 1, table.getn(stateArray)
            do
            ----dhcp server
                if      stateArray[i] == "D" then dhcpd(dhcpServer, "start")
            ----virtual machines
                elseif  stateArray[i] == "1" then vboxmanage(VMMACHINES[1][1], VMMACHINES
                elseif  stateArray[i] == "2" then vboxmanage(VMMACHINES[2][1], VMMACHINES
                elseif  stateArray[i] == "3" then vboxmanage(VMMACHINES[3][1], VMMACHINES
                elseif  stateArray[i] == "4" then vboxmanage(VMMACHINES[4][1], VMMACHINES
                end
            end
        end
    end
----check web server
    print("----------checking web werver----------")
    if checkMachine(webip, webname) ~= 0
    then
         print("--web server not running, starting..")
         vboxmanage(webip, webname, "start")
    end
----check lab state
    print("----------checking current state----------")
    currentState = ""
    if table.getn(stateArray) ~= 0
    then
        for i = 1, table.getn(stateArray)
        do
            if stateArray[i] == "D"
            then
                print("----------checking DHCP server----------")
                if checkdhcpd(dhcpServer) ~= 0
                then
                    print("--dhcp not running, starting..")
                    dhcpd(dhcpServer, "start")
                end
                currentState = currentState.."D"
            end
            for j = 1, 4
            do
                if stateArray[i] == tostring(j)
                then
                    print("----------checking virtual machine "..tostring(j).."----------
                    if checkMachine(VMMACHINES[j][1], VMMACHINES[j][2]) ~= 0
                    then
                        print("--machine "..tostring(j).." not running, starting..")
                        vboxmanage(VMMACHINES[j][1], VMMACHINES[j][2], "start")
                    end
                    currentState = currentState..j
                end
            end
        end
    end
    os.execute("echo '"..currentState.."' > "..statusFile)
--[[--check exercise feedback
    print("----------checking exercise results----------")
    if fileExists(checkFile) == true
    then
        file = io.open(checkFile, "r")
        exercise = file:read('*line')
        file:close()
        os.execute("rm -f "..checkFile)
        local exercise = exercise.sub(exercise, 1, 2);
        if     exercise == "1" then checkExercise1();
        elseif exercise == "2" then checkExercise1();
elseif exercise == "5" then checkExercise1();
```

```
        elseif exercise == "7" then checkExercise1();
        elseif exercise == "8" then checkExercise1();
        elseif exercise == "9" then checkExercise1();
        elseif exercise == "10" then checkExercise1();
        end
    end
--]]
    os.execute('sleep 2s')
end
```

C.2 tplink.ksc

```
set line /dev/ttyS0
set flow-control none
set carrier-watch off
set speed 38400
lineout
input 5 TL-SG3216>
lineout enable
input 5 TL-SG3216#
lineout reset
input 5 (Y/N)
lineout y
exit
```

Appendix D

Experiment Instructions

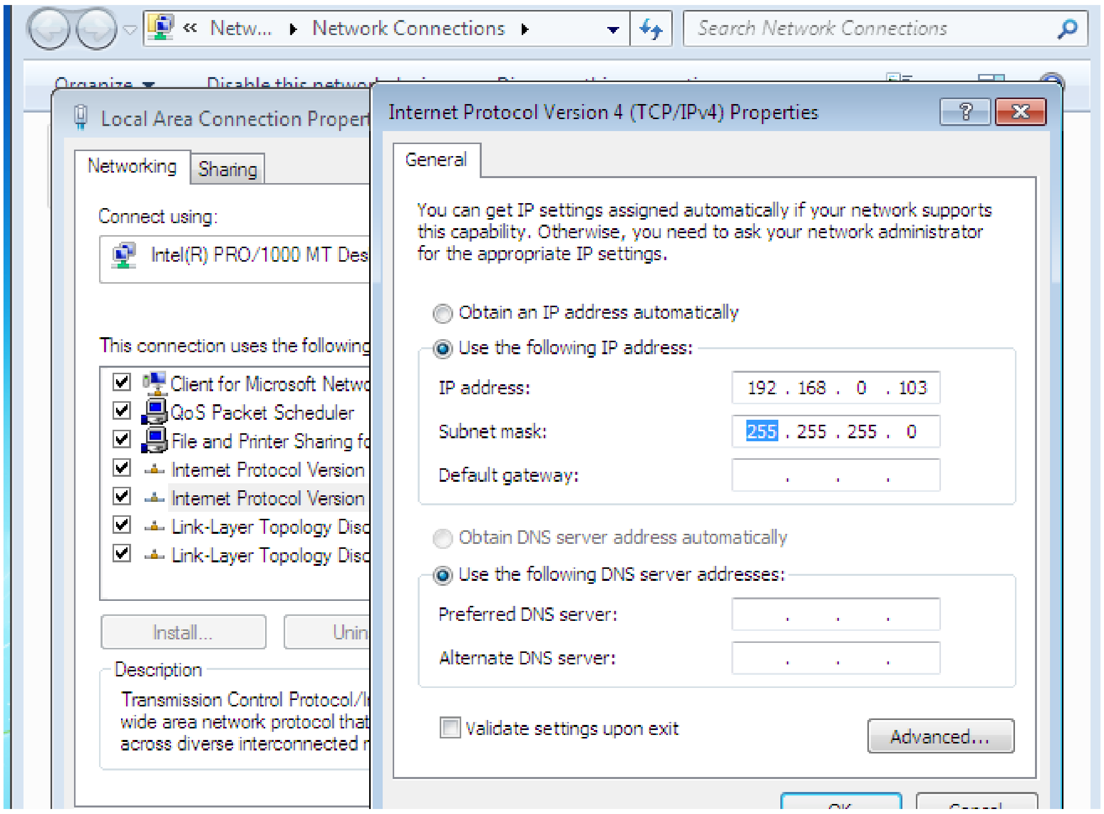
D.1 Exercise 1 - Static Ip Addresses

D.1.1 Introduction

In this Experiment you will create a small network of consisting of four computers. You will be assigning each computer an IP address manually and then you will test the network by making sure that each computer can connect to another.

D.1.2 Methodology

1 - To configure a Windows machine with a static IP first click on the network icon in the system tray then select 'Open Network and Sharing Centre' then 'Change Adapter Settings'. Alternatively type 'View Network Connections' in the search area of the start menu. Right-click the network interface you want to configure then select properties. Select Internet Protocol Version 4 (TCP/IPv4) then click properties.


« Netw... ▸ Network Connections ▸

Search Network Connections

Local Area Connection Propert

Networking | Sharing

Connect using:

Intel(R) PRO/1000 MT Des

This connection uses the following

- [✓] Client for Microsoft Netwo
- [✓] QoS Packet Scheduler
- [✓] File and Printer Sharing fo
- [✓] Internet Protocol Version
- [✓] Internet Protocol Version
- [✓] Link-Layer Topology Disc
- [✓] Link-Layer Topology Disc

Install... | Unin

Description

Transmission Control Protocol/I wide area network protocol that across diverse interconnected n

Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

( ) Obtain an IP address automatically
[✓ Use the following IP address:]

IP address:	192 . 168 . 0 . 103
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	. . .

( ) Obtain DNS server address automatically
[✓ Use the following DNS server addresses:]

Preferred DNS server:	. . .
Alternate DNS server:	. . .

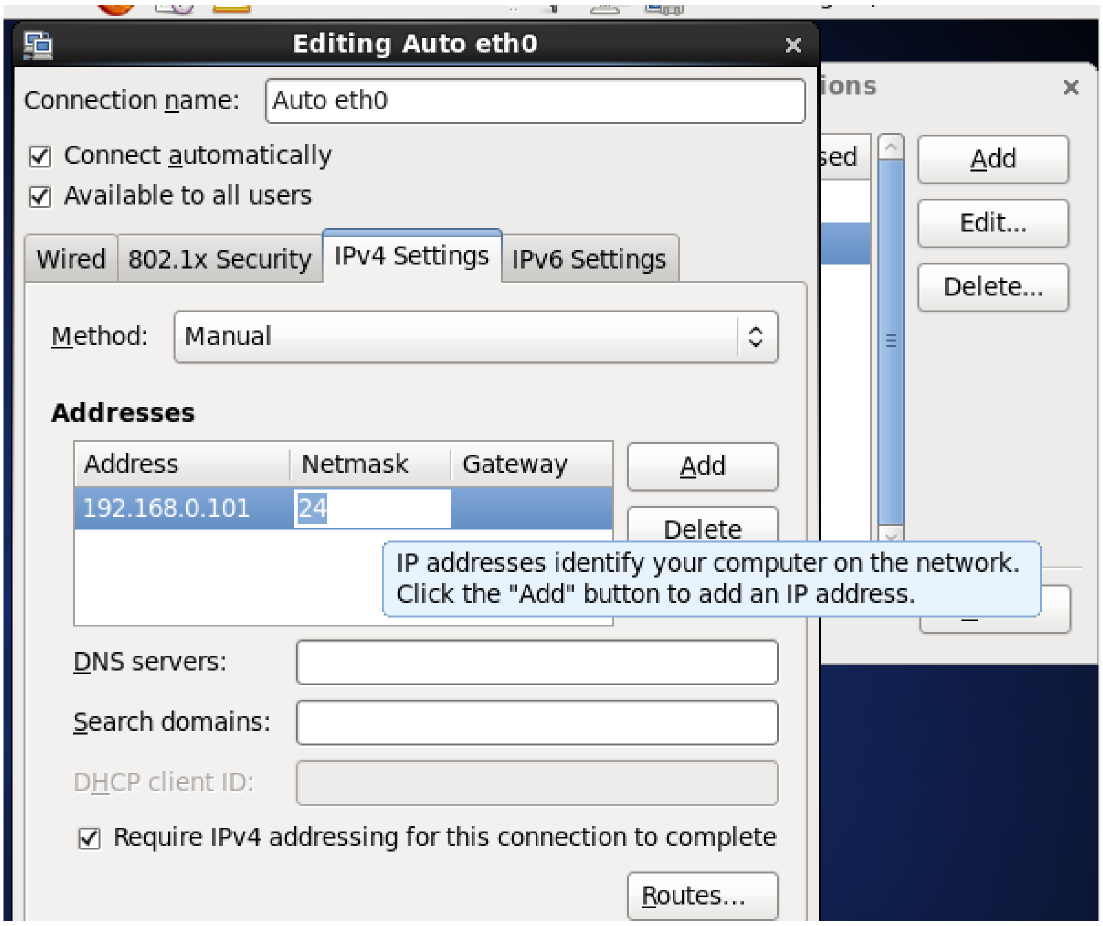
[ ] Validate settings upon exit

Advanced...

Figure D.1: Configuring a network address under Windows

2 - To Configure a Linux machine right-click the network icon in the system tray then select 'Edit Connections'. Select eth0 from the list then select Edit. Select the IPv4

Tab and change Method to Manual



IP addresses identify your computer on the network.
Click the "Add" button to add an IP address.

Figure D.2: Configuring a network address under Linux

3 - Configure the *first* ethernet interface of the machines to use the following IP addresses (leave the default gateway and DNS server blank for the moment). When applying these settings to a Linux machine you may be asked for a password. The password is `password`.

Linux 1 - Address: 192.168.0.101, Netmask: 225.225.225.0

Linux 2 - Address: 192.168.0.102, Netmask: 225.225.225.0

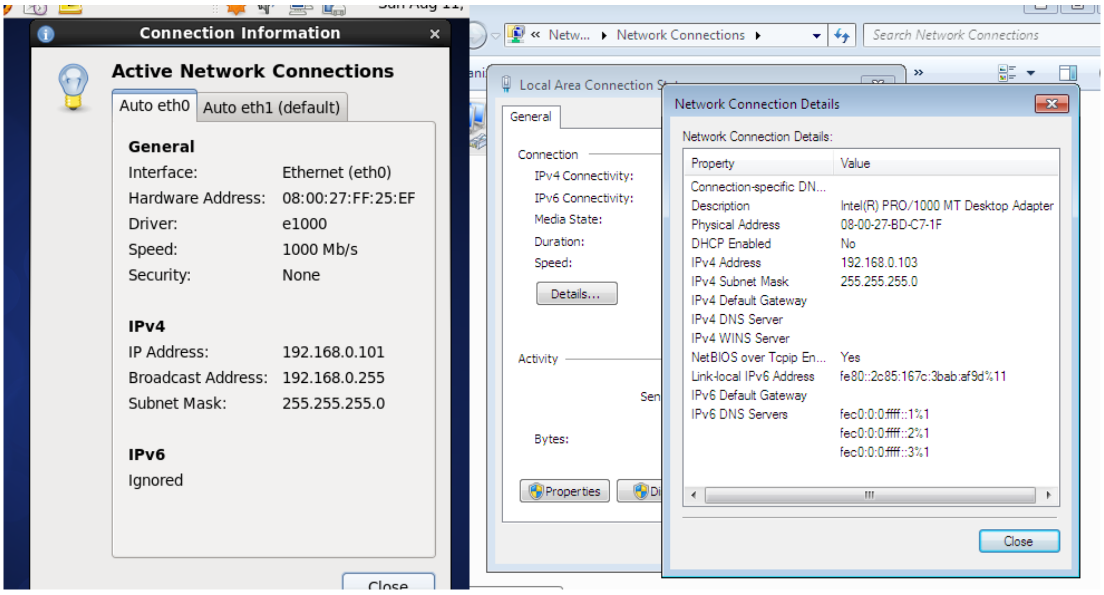
Windows 1 - Address: 192.168.0.103, Netmask: 225.225.225.0

Windows 2 - Address: 192.168.0.103, Netmask: 225.225.225.0

4 - Click OK/Apply to apply the new settings

5 - Confirm that each computers have the correct IP addresses. On Windows right-click

the network interface then select 'status', then click 'details'. On Linux right-click the network icon in the system tray then select 'Connection Information' then select the eth0 tab.



General

Interface:	Ethernet (eth0)
Hardware Address:	08:00:27:FF:25:EF
Driver:	e1000
Speed:	1000 Mb/s
Security:	None

IPv4

IP Address:	192.168.0.101
Broadcast Address:	192.168.0.255
Subnet Mask:	255.255.255.0

IPv6

Ignored

Network Connection Details:

Property	Value
Connection-specific DN...	
Description	Intel(R) PRO/1000 MT Desktop Adapter
Physical Address	08-00-27-BD-C7-1F
DHCP Enabled	No
IPv4 Address	192.168.0.103
IPv4 Subnet Mask	255.255.255.0
IPv4 Default Gateway	
IPv4 DNS Server	
IPv4 WINS Server	
NetBIOS over Tcpip En...	Yes
Link-local IPv6 Address	fe80::2c85:167c:3bab:af9d%11
IPv6 Default Gateway	
IPv6 DNS Servers	fec0:0:0:ffff::1%1
	fec0:0:0:ffff::2%1
	fec0:0:0:ffff::3%1

Figure D.3: Checking IP Settings

6 - Confirm that the computers can communicate with each other by running `ping <ip-address>`, where <ip-address>is the address of the computer you want to send the ICMP Packet to. Sending ICMP Packets with the `ping` is a great way to diagnose whether a computer on the network is running or not.

D.2 Exercise 2 - Multiple networks and Routing

D.2.1 Introduction

In this exercise you will create two networks and allow packets to be routed between them. Packets can only be sent to computers on the same subnet. If a host doesn't know an IP address, it either sends it to a specified host depending on the routing table, or drops it. The routing table is a table that stores information on where packets are meant to be sent depending on the packets destination. Each 'route' contains a destination (which can either be a network or a particular host) and a gateway (where to send the packet meant for the destination). For example if you have a packet that is meant to be sent to the IP address ¡code¿8.8.8.8¡/code¿, the operating system will look through the routing table to see if there is an the IP address matches any of the destination entries, if there is it will send the packet to the corresponding gateway.

D.2.2 Methodology

1 - Connect to Linux 1 and give it an IP address of 192.168.0.2 (either interface will be fine).

2 - Open the web browser and type 192.168.0.1 into the address bar. This will allow you to access the configurable switch. Select VLans from the menu on the side. Virtual LAN (VLAN) allows you to break up the ports on the configurable switch into "virtual switches". allowing you to create multiple networks on the same physical switch without having to rewire connections or buy extra switches.

The virtual machines are connect to the configurable switch. The Following shows what machines are connected to each port on the switch.

Linux 1 - eth0 - port 1
Linux 1 - eth1 - port 2
Linux 2 - eth0 - port 3
Linux 2 - eth1 - port 4
Windows 1 - Local Area Network 1 - port 5
Windows 1 - Local Area Network 2 - port 6

Windows 2 - Local Area Network 1 - port 7

Windows 2 - Local Area Network 2 - port 8

2 - Create two VLANs and put Linux 1 on the first one, Linux 2 on the second one, and Windows 1 on both VLANs by putting one ethernet port on the first VLAN.

3 - To use a windows machine as a router, a registry key has to be set. Type in regedit in the windows start menu to bring it up in the menu. Then click it to run it. Locate the key `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters`. Select `IPEnableRouter` and set it's value to 1. For future reference you can enable ip forwarding in Linux by setting `/proc/sys/net/ipv4/ip_forward` to 1. To enable it permanently change `/etc/sysctl.conf` and set `net.ipv4.ip_forward = 1`, or add it if it isn't there.

4 - The first method of routing we will use is by setting a default route (also called a default gateway). The routing table will have the entry network: `0.0.0.0`, gateway: `<:address>`. This means that all traffic (as denoted by the `0.0.0.0`) is sent to the gateway `<address>`. The default route is usually applied last in the routing table to allow other routing rules to be applied (i.e. the default route is only used when other routing rules do not match). Configure the ethernet interfaces of the machines to use the following IP addresses (leave the default gateway and DNS server blank for the moment).

Linux 1 - Address: 192.168.0.101, Netmask: 225.225.225.0, Gateway: 192.168.0.102

Linux 2 - Address: 192.168.5.101, Netmask: 225.225.225.0, Gateway: 192.168.5.102

Windows 1 - Address: 192.168.0.102, Netmask: 225.225.225.0

Windows 1 - Address: 192.168.5.102, Netmask: 225.225.225.0

As you can see Linux 1 will be on one network and Linux 2 will be on the other. Windows 1 will be connected to both networks and will act as a gateway between them. Make sure that each interface on Windows 1 is configured correctly (e.g the interface that is on the same network as Linux 1 is on the same subnet as Linux 1).

5 - Test that the packets are routed properly by trying to ping Linux 1 from Linux 2 (or vice-versa). If done correctly it should be successful.

6 - Now instead of setting a default route we are going to modify the routing table directly.

Open a terminal on one of the Linux machines and type in `route` to display the routing table. It should only show an entry for the default gateway but it may also show a route for the network that the machine is on.

¡div¿¡img src=exercise5-12.png alt="Sample output for netstat"/¿ ¡br/¿sample output for route¡/div¿

7 - Now edit the Linux machines' IP settings and remove the default gateway. You may have to delete the entry and re-add the IP settings to be able to save the configuration. You may also have to reconnect to the network in order to apply those settings. Just click the network icon in the system tray and click on the name of the network to reconnect it.

8 - Now open the terminal and log into root by typing `su` and using the password `password`. Add a new route to the routing table by running `route add net <destination> netmask <netmask> gw <gateway>`

Linux 1 - Destination: 192.168.5.0, Netmask: 225.225.225.0, Gateway: 192.168.0.102
Linux 2 - Destination: 192.168.0.0, Netmask: 225.225.225.0, Gateway: 192.168.5.102

9 - Test that the network works by again sending a packet from one Linux machine to another.

10 - OPTIONAL - If you want to play with routing more you can create more networks, change subnets, etc.

D.3 Exercise 3 - Packet Analysing

D.3.1 Introduction

In this exercise you will use a packet analyzer view the packets travelling through an ethernet interface. Wireshark is a powerful open-source network protocol analyzer, which we will be using in this exercise. Wireshark is already installed on the Windows machines.

D.3.2 Methodology

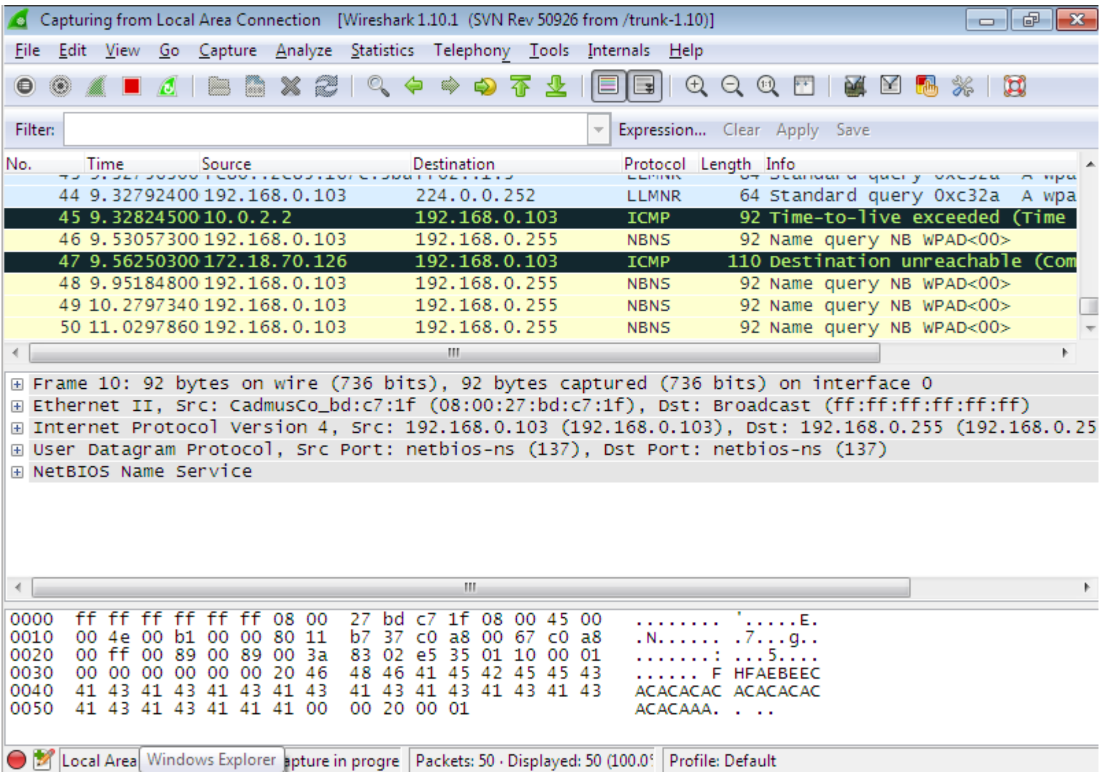
1 - Start Wireshark.

2 - Select capture from the menu bar and select interfaces. Then select an interface you want to capture packets from and start capturing.

3 - When Wireshark is capturing packets it will show the screen in three sections. The top section shows the packets as they are passing through the interface. If you click on a packet in the top section, all the details of that packet will be shown in the middle section, and the raw data of the selected packet will be shown in the bottom section.

4 - Try pinging another machine. The packets should come up in the packet analyser.

5 - Select a packet coming through and look through all its details taking note of information like source, destination, protocol, ports, etc. Also take note of the protocol layers.


Capturing from Local Area Connection [Wireshark 1.10.1 (SVN Rev 50926 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.    Time        Source          Destination     Protocol Length Info
    44 9.32792400 192.168.0.103    224.0.0.252      LLMNR       64 Standard query 0xc32a  A wpa
    45 9.32824500 10.0.2.2         192.168.0.103    ICMP        92 Time-to-live exceeded (Time
    46 9.53057300 192.168.0.103    192.168.0.255    NBNS        92 Name query NB WPAD<00>
    47 9.56250300 172.18.70.126    192.168.0.103    ICMP       110 Destination unreachable (Com
    48 9.95184800 192.168.0.103    192.168.0.255    NBNS        92 Name query NB WPAD<00>
    49 10.2797340 192.168.0.103    192.168.0.255    NBNS        92 Name query NB WPAD<00>
    50 11.0297860 192.168.0.103    192.168.0.255    NBNS        92 Name query NB WPAD<00>
```

```
⊞ Frame 10: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
⊞ Ethernet II, Src: CadmusCo_bd:c7:1f (08:00:27:bd:c7:1f), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊞ Internet Protocol Version 4, Src: 192.168.0.103 (192.168.0.103), Dst: 192.168.0.255 (192.168.0.25
⊞ User Datagram Protocol, Src Port: netbios-ns (137), Dst Port: netbios-ns (137)
⊞ NetBIOS Name Service
```

```
0000  ff ff ff ff ff ff 08 00  27 bd c7 1f 08 00 45 00   ........ '.....E.
0010  00 4e 00 b1 00 00 80 11  b7 37 c0 a8 00 67 c0 a8   .N...... .7...g..
0020  00 ff 00 89 00 89 00 3a  83 02 e5 35 01 10 00 01   .......: ...5....
0030  00 00 00 00 00 00 20 46  48 46 41 45 42 45 45 43   ...... F HFAEBEEC
0040  41 43 41 43 41 43 41 43  41 43 41 43 41 43 41 43   ACACACAC ACACACAC
0050  41 43 41 43 41 41 41 00  00 20 00 01               ACACAAA. . ..
```

Figure D.4: Sample output for wireshark

D.4 Exercise 4 - Introduction to the Linux Operating System

D.4.1 Introduction

This exercise (...well, not really an exercise) will give you an introduction to the Linux filesystem as well as the Terminal and some commonly used terminal commands. Linux is vastly different from Windows. Windows is designed for ordinary users, as such many aspects of the operating system are hidden. Unix (upon which Linux is based) is the opposite, with most if not all of the operating system exposed allowing the user a large amount of control over the system.

D.4.2 The Linux Filesystem

Unlike Windows filesystems which start with a drive letter. Linux uses a single heirarchal file structure starting at /, also called the root directory. *All* files and folders (including those on separate drives/partitions) are descendants of this root directory.

Instead of assigning drive letters to partitions, Linux *mounts* the partitions to a location in the existing filesystem. You can visualise it as attaching the top level directory of the drive/partition to an existing folder on the Linux filesystem. So when you access that folder you are actually accessing the drive/partition.

The Filesystem Hierarchy Standard (FHS) defines the directory structure of Linux systems. Most Linux distributions follow the FHS but some may have slight deviations. The following is the directory structure according to the FHS.

/ The root directory. This is the top level directory of the entire filesystem.

/bin Contains Essential programs for all users.

/boot Contains the Linux kernel and other files required at boot time.

/dev All files here are representations of the system hardware. The Linux philosophy is that everything is a file. Reading and writing to files in this directory will read and write to the hardware directly.

/etc Contains all the system configuration files.

/home Contains all the user files. Each user gets their own folder here called their home directory.

/lib Libraries essential to the programs in /bin. Some 64-bit distributions also include a /lib64 directory

/media Contains mount points for removable media (although partitions can be mounted to any folder, it is common practice to mount external media to folders within /media)

/mnt Contains mount points for temporary filesystems.

/opt Used for optional software. This directory is used less and less in favour of /usr/local.

/proc A virtual filesystem which represents the internals of the Linux kernel.

/root Home directory for the root user.

/sbin Contains essential system binaries.

/srv Contains files to be used by services.

/tmp Contains temporary files which are usually deleted at boot time.

/usr Contains read-only user data, user applications (/usr/bin), user libraries (/usr/lib), documentation and source code.

/var Contains files that are expected to continually change such as log files, spooling data, ...

D.4.3 Linux Permissions

Unix Permissions

Traditional Unix (and by extension Linux) permissions come in three classes: Owner, Group and Other. Within each class there are three distinct permissions: read, write and execute. This comes to a total of nine different types of permissions that *all* files and directories have associated with them. Lets do an example:

1 - Open a terminal (`Applications->System Tools->Terminal`). Note that the terminal will start in the users home directory (`/home/<user-name>/`, also called `/` for the current user).

2 - Create a file called `test` in the current directory by typing `touch test`.

3 - now list all the files in the current directory and their permissions by typing `ls -l`. You should get something similar to:

```
 total 32
drwxr-xr-x.  2 user user 4096 Jun 24 01:57 Desktop
drwxr-xr-x.  2 user user 4096 Jun 24 01:57 Documents
drwxr-xr-x.  2 user user 4096 Jun 24 01:57 Downloads
drwxr-xr-x.  2 user user 4096 Jun 24 01:57 Music
drwxr-xr-x.  2 user user 4096 Jun 24 01:57 Pictures
drwxr-xr-x.  2 user user 4096 Jun 24 01:57 Public
drwxr-xr-x.  2 user user 4096 Jun 24 01:57 Templates
-rw-rw-r--.  1 user user 0 Jun 25 23:29 test
```

```
drwxr-xr-x.  2 user user 4096 Jun 24 01:57 Videos
```

For a full explanation of the output see http://en.wikipedia.org/wiki/Ls

The letters to the left (minus the very first character, which is set to `d` to indicate a directorie) shows the permissions that are set for the file. Our test file shows the permissions: `rw-rw-r--`. The first three character show permissions for the owner of the file, the next three show permissions for the group associated with that file, and the last three are permissions for everybody else. As you can see our test file has read/write access for both the user and group, and read-only access for everyone else.

Take note that directories have their executable permissions set. This allows you to browse through the directories.

Superuser: root

Most operating systems have a special user accout called the superuser which has *complete, unrestricted* access to the entire system regardless of permissions. Windows systems have a user called `Administrator` which is usually disabled by default, with UAC providing elevated priveleges. The Linux superuser is called `root` which may or may not be enabled depending on the distribution of Linux. Distributions with `root` disabled have a program called `sudo` which can elevate priveleges. On the lab Linux machines `root` is enabled and can be logged in to using the terminal.

Note: Make sure that you understand Linux permissions. On most Linux systems (including the lab machines) a large majority of the operating system is owned by `root`, and since you log into the machines as a user named `user`, the operating system will not allow you to modify those files. Keep this in mind as in later exercises you will have to log in as `root` and modify some system files.

D.4.4 Useful Linux Commands

Here are some basic linux commands that are very useful. If you want to know more about the command you can run `man <command>` in a terminal to view the command's manual.

`cd <directior>` Change the current directory. Typing `cd ../` will change to the parent of the current directory. Please note that Linux file and path names are *case-sensitive*

`ls` Lists all files and folders in the current directory

`pwd` Show the current directory path

`man <command>` Shows the manual page for the given command. Very useful if you want to know how to use a particular command

`cp <source> <destination>` Copy a file/directory ¡source¿ to ¡destination¿. (Note. you must use the recursive option ¡code¿-r¡/code¿ when copying folders)

`mkdir <directory>` Create a new directory

`rm <file1> <file2>...` Remove files. Use the `-r` option to remove folders. Do NOT run this on the root directory / as it could potentially wipe the operating system and all mounted drives

`cat <file1> <file2>...` Concatinate files and display the result. Useful for just displaying the contents of a particular file

`locate <file>` Show the location of a file.

`grep <string> <file>` Find a string within a file

`ps` List current running processes. Running `ps` by itself only shows processes within the current terminal. To show all running process run —textttps aux

`top` Show the top running processes ordered by cpu usage (default)

`touch <file>` Updates the timestamp of a file. if the file doesn't exist it is created

`df` Shows disk usage

`free` Shows RAM usage

`less` shows the contents of a file allowing you to scroll through the file

`vi` A command line text editor

`whoami` Show the current user

GUI programs can also be run from the terminal. One of the most useful programs is `gedit` which is the text editor for gnome (the desktop GUI on the lab's Linux machines). This is handy for when we need to edit system configuration files. To edit a system file, open a terminal, log in to root using `su`, navigate to the directory containing the file you want to edit using `cd <directory>`, and open the file using `gedit <filename>`.

Linux makes extensive use of piping, which allows the output of one program to be used as input for another program. For example `ps -aux | grep openssh` will get the list of currently running processes, which will be input to `grep` which will search for the string `openssh`. Program output can also be piped to files, e.g `ps -aux > processlist.txt` will get the list of currently running processes and write it to `processlist.txt`

Practice the above commands as some of them will be useful in later exercises.

D.5 Exercise 5 - Windows and Linux Networking Tools

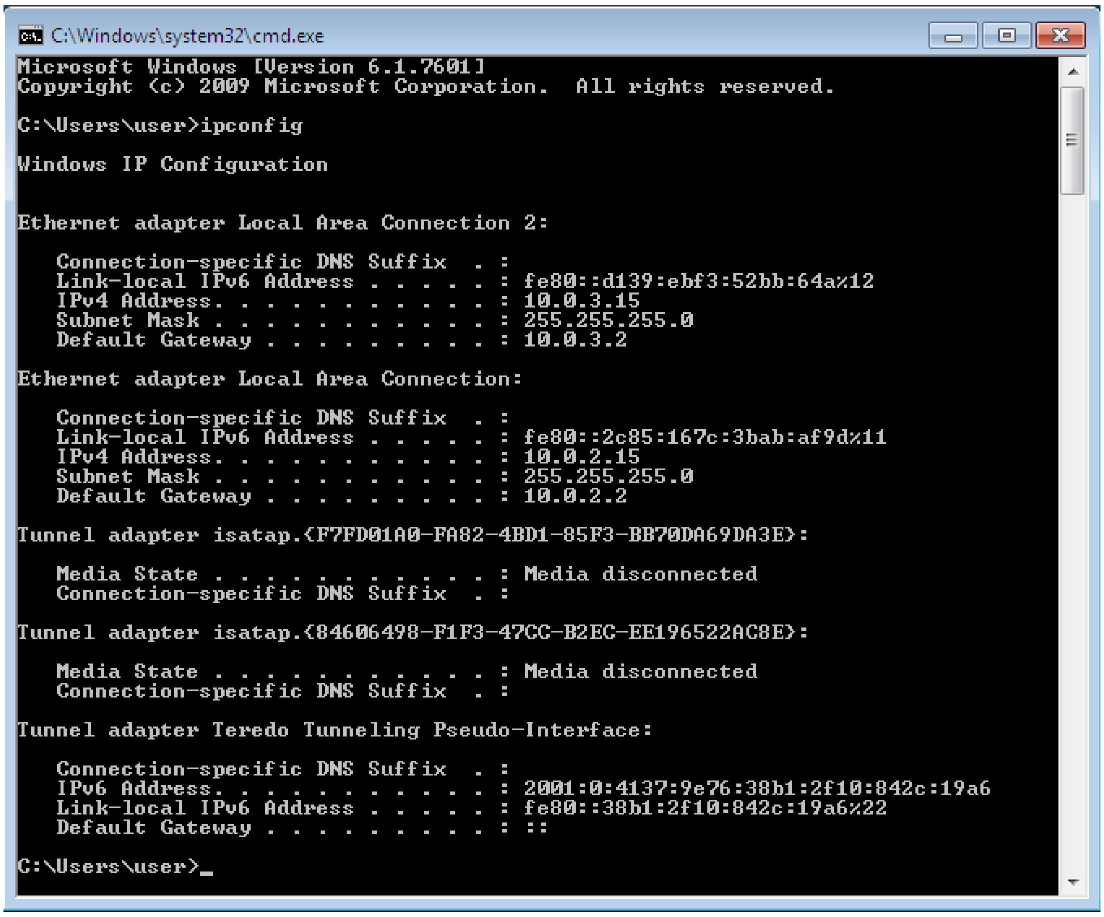
D.5.1 Introduction

In this exercise you will learn how to use some commonly used networking tools under both Windows and Linux. These tools are used to show information about ethernet ports, information about the network the computer is connected to, as well as troubleshoot any problems in the network. Under Linux you can look up the manual page for each command.

D.5.2 Windows Networking Tools

ipconfig

Ipconfig will display the system's network configuration. Run `ipconfig /all` to display more detailed information.



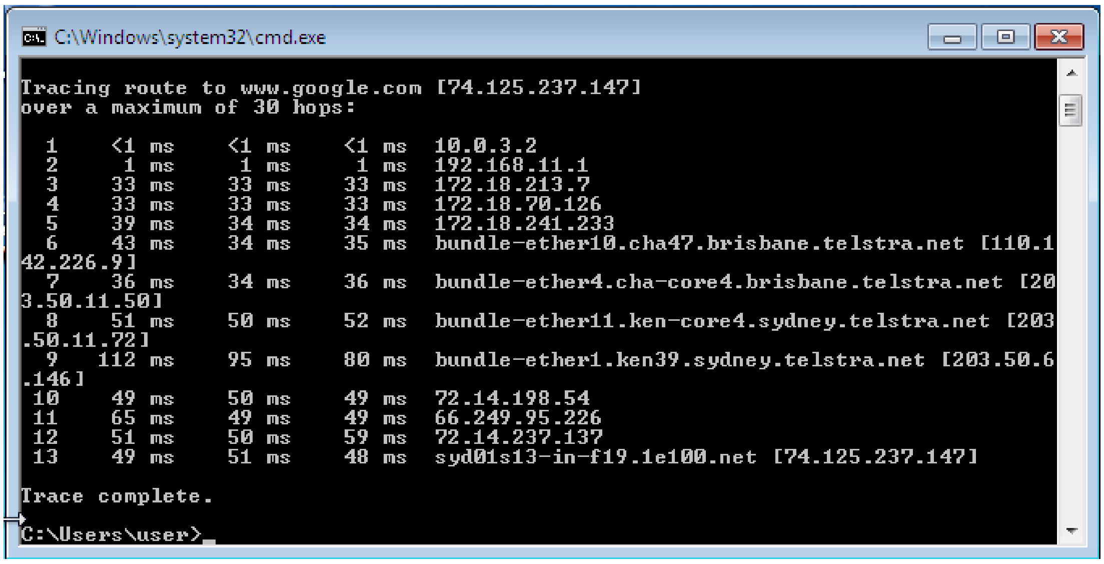
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\user>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::d139:ebf3:52bb:64a%12
   IPv4 Address. . . . . . . . . . . : 10.0.3.15
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.0.3.2

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::2c85:167c:3bab:af9d%11
   IPv4 Address. . . . . . . . . . . : 10.0.2.15
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.0.2.2

Tunnel adapter isatap.{F7FD01A0-FA82-4BD1-85F3-BB70DA69DA3E}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter isatap.{84606498-F1F3-47CC-B2EC-EE196522AC8E}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Teredo Tunneling Pseudo-Interface:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:0:4137:9e76:38b1:2f10:842c:19a6
   Link-local IPv6 Address . . . . . : fe80::38b1:2f10:842c:19a6%22
   Default Gateway . . . . . . . . . : ::

C:\Users\user>_
```

Figure D.5: sample output for ipconfig

tracert

Usage: `tracert <host>`

Show the path that a packet travels to a given host.



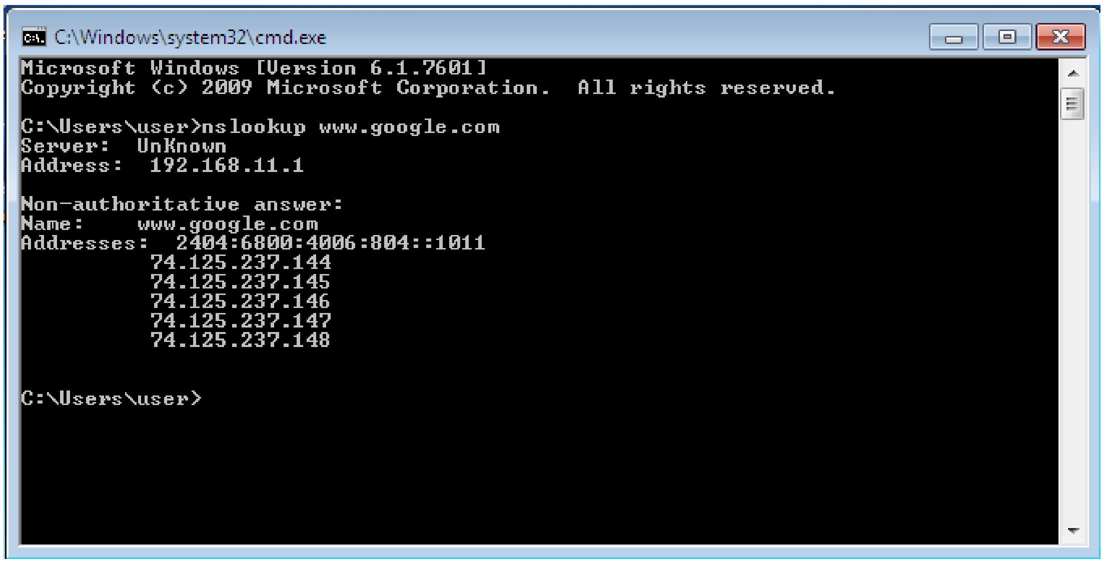
```
Tracing route to www.google.com [74.125.237.147]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  10.0.3.2
  2     1 ms     1 ms     1 ms  192.168.11.1
  3    33 ms    33 ms    33 ms  172.18.213.7
  4    33 ms    33 ms    33 ms  172.18.70.126
  5    39 ms    34 ms    34 ms  172.18.241.233
  6    43 ms    34 ms    35 ms  bundle-ether10.cha47.brisbane.telstra.net [110.1
42.226.9]
  7    36 ms    34 ms    36 ms  bundle-ether4.cha-core4.brisbane.telstra.net [20
3.50.11.50]
  8    51 ms    50 ms    52 ms  bundle-ether11.ken-core4.sydney.telstra.net [203
.50.11.72]
  9   112 ms    95 ms    80 ms  bundle-ether1.ken39.sydney.telstra.net [203.50.6
.146]
 10    49 ms    50 ms    49 ms  72.14.198.54
 11    65 ms    49 ms    49 ms  66.249.95.226
 12    51 ms    50 ms    59 ms  72.14.237.137
 13    49 ms    51 ms    48 ms  syd01s13-in-f19.1e100.net [74.125.237.147]

Trace complete.

C:\Users\user>_
```

Figure D.6: sample output for tracert

nslookup

Usage: `nslookup <hostname>`

Determine the IP address of a given host.



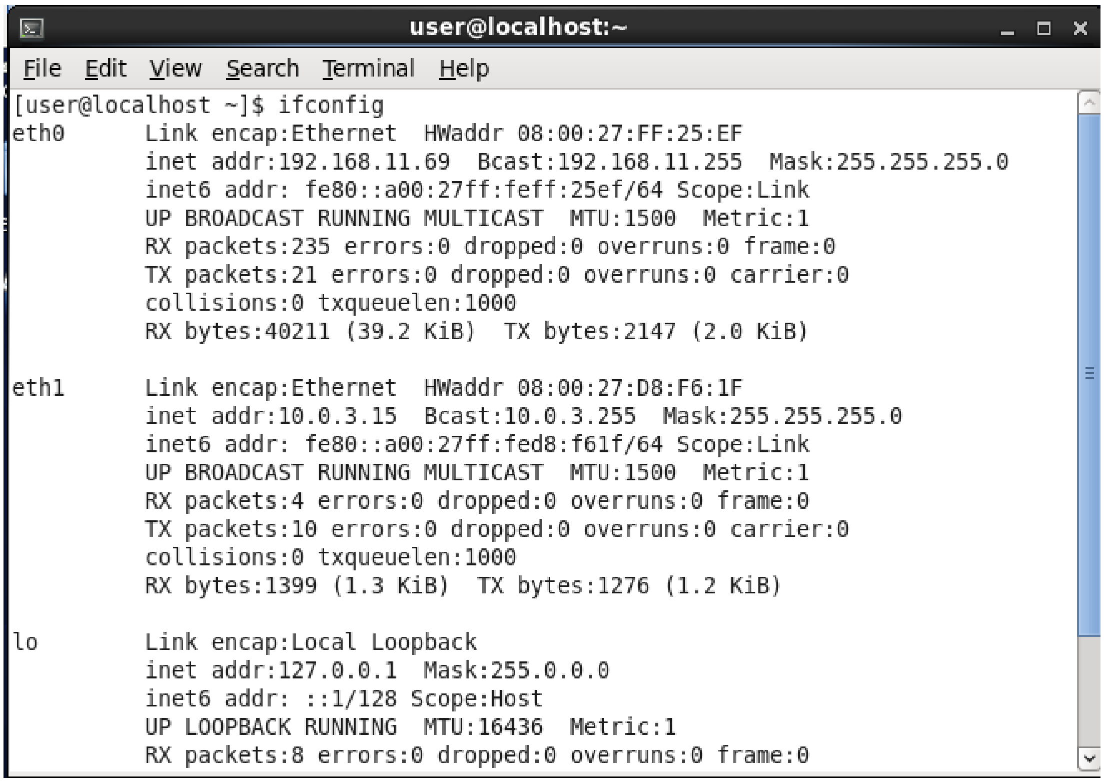
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\user>nslookup www.google.com
Server:  UnKnown
Address:  192.168.11.1

Non-authoritative answer:
Name:    www.google.com
Addresses:  2404:6800:4006:804::1011
          74.125.237.144
          74.125.237.145
          74.125.237.146
          74.125.237.147
          74.125.237.148


C:\Users\user>
```

Figure D.7: sample output for nslookup

D.5.3 Linux Networking Tools

ifconfig

Ifconfig will display the system's network configuration. Run `ifconfig <interface> <ipaddress>` to assign an ip address to an interface.



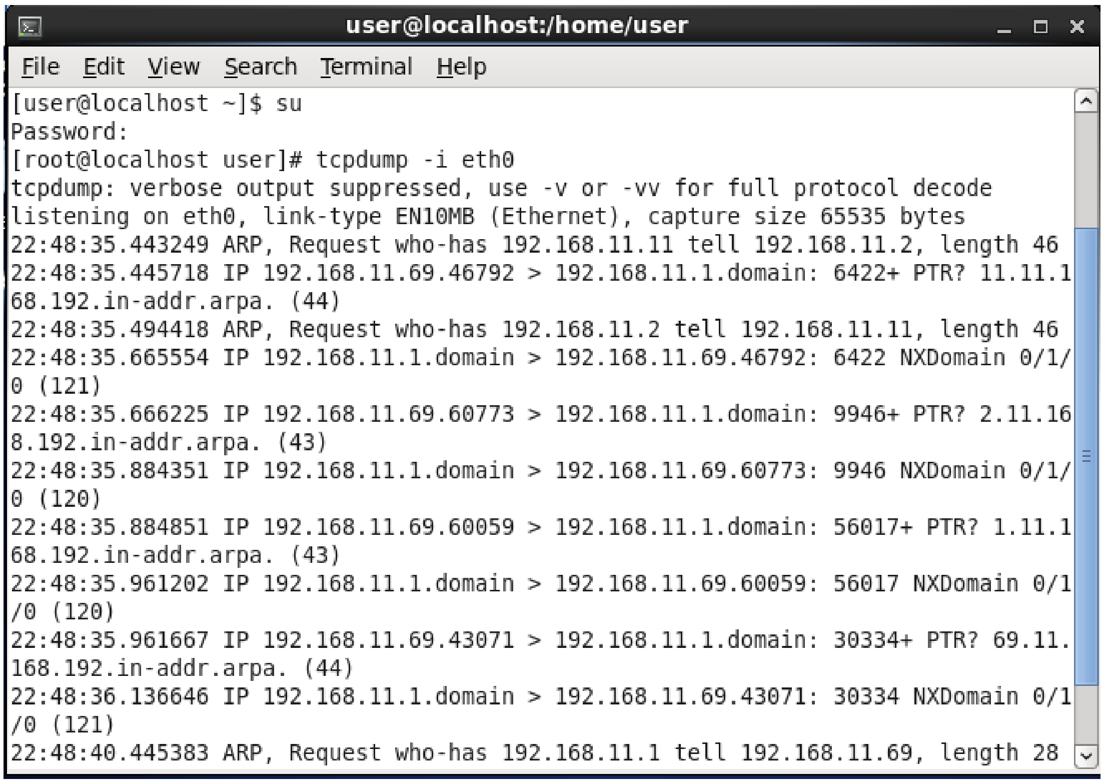
```
[user@localhost ~]$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:FF:25:EF
          inet addr:192.168.11.69  Bcast:192.168.11.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:feff:25ef/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:235 errors:0 dropped:0 overruns:0 frame:0
          TX packets:21 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:40211 (39.2 KiB)  TX bytes:2147 (2.0 KiB)

eth1      Link encap:Ethernet  HWaddr 08:00:27:D8:F6:1F
          inet addr:10.0.3.15  Bcast:10.0.3.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fed8:f61f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1399 (1.3 KiB)  TX bytes:1276 (1.2 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
```

Figure D.8: ¿sample output for ifconfig

tcpdump

Usage: `tcpdump -i <interface>`

Capture packets from a giveen interface and dump packet information to screen. Add the -v and -vv options for more detail. This utility *must* be run as root.


user@localhost:/home/user
File Edit View Search Terminal Help

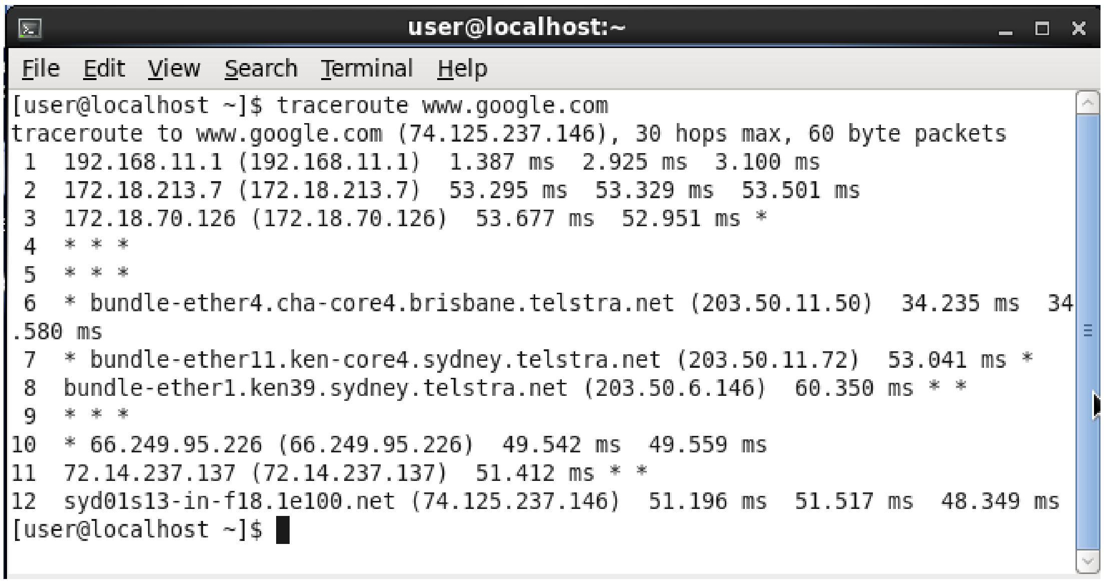
```
[user@localhost ~]$ su
Password:
[root@localhost user]# tcpdump -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
22:48:35.443249 ARP, Request who-has 192.168.11.11 tell 192.168.11.2, length 46
22:48:35.445718 IP 192.168.11.69.46792 > 192.168.11.1.domain: 6422+ PTR? 11.11.1
68.192.in-addr.arpa. (44)
22:48:35.494418 ARP, Request who-has 192.168.11.2 tell 192.168.11.11, length 46
22:48:35.665554 IP 192.168.11.1.domain > 192.168.11.69.46792: 6422 NXDomain 0/1/
0 (121)
22:48:35.666225 IP 192.168.11.69.60773 > 192.168.11.1.domain: 9946+ PTR? 2.11.16
8.192.in-addr.arpa. (43)
22:48:35.884351 IP 192.168.11.1.domain > 192.168.11.69.60773: 9946 NXDomain 0/1/
0 (120)
22:48:35.884851 IP 192.168.11.69.60059 > 192.168.11.1.domain: 56017+ PTR? 1.11.1
68.192.in-addr.arpa. (43)
22:48:35.961202 IP 192.168.11.1.domain > 192.168.11.69.60059: 56017 NXDomain 0/1
/0 (120)
22:48:35.961667 IP 192.168.11.69.43071 > 192.168.11.1.domain: 30334+ PTR? 69.11.
168.192.in-addr.arpa. (44)
22:48:36.136646 IP 192.168.11.1.domain > 192.168.11.69.43071: 30334 NXDomain 0/1
/0 (121)
22:48:40.445383 ARP, Request who-has 192.168.11.1 tell 192.168.11.69, length 28
```

Figure D.9: sample output for tcpdump

traceroute

Usage: `traceroute <host>`

Show the path that a packet travels to a given host.



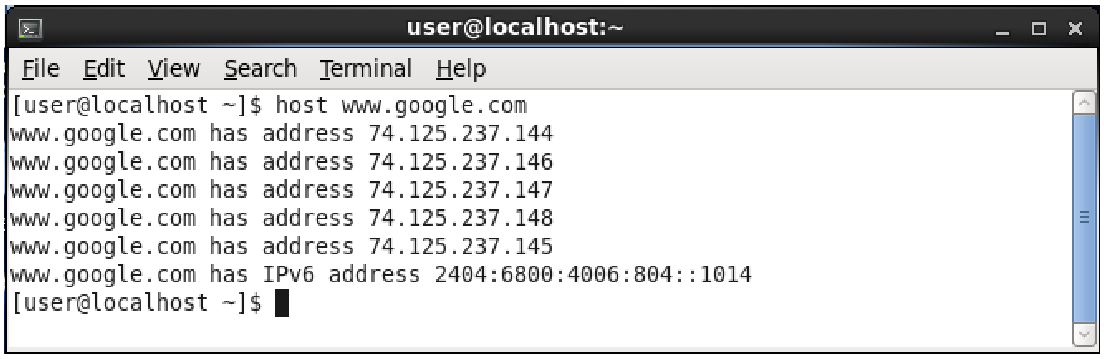
```
[user@localhost ~]$ traceroute www.google.com
traceroute to www.google.com (74.125.237.146), 30 hops max, 60 byte packets
 1  192.168.11.1 (192.168.11.1)  1.387 ms  2.925 ms  3.100 ms
 2  172.18.213.7 (172.18.213.7)  53.295 ms  53.329 ms  53.501 ms
 3  172.18.70.126 (172.18.70.126)  53.677 ms  52.951 ms *
 4  * * *
 5  * * *
 6  * bundle-ether4.cha-core4.brisbane.telstra.net (203.50.11.50)  34.235 ms  34
.580 ms
 7  * bundle-ether11.ken-core4.sydney.telstra.net (203.50.11.72)  53.041 ms *
 8  bundle-ether1.ken39.sydney.telstra.net (203.50.6.146)  60.350 ms * *
 9  * * *
10  * 66.249.95.226 (66.249.95.226)  49.542 ms  49.559 ms
11  72.14.237.137 (72.14.237.137)  51.412 ms * *
12  syd01s13-in-f18.1e100.net (74.125.237.146)  51.196 ms  51.517 ms  48.349 ms
[user@localhost ~]$ 
```

Figure D.10: sample output for traceroute

host

Usage: `host <hostname>`

Determine the IP address of a given host.



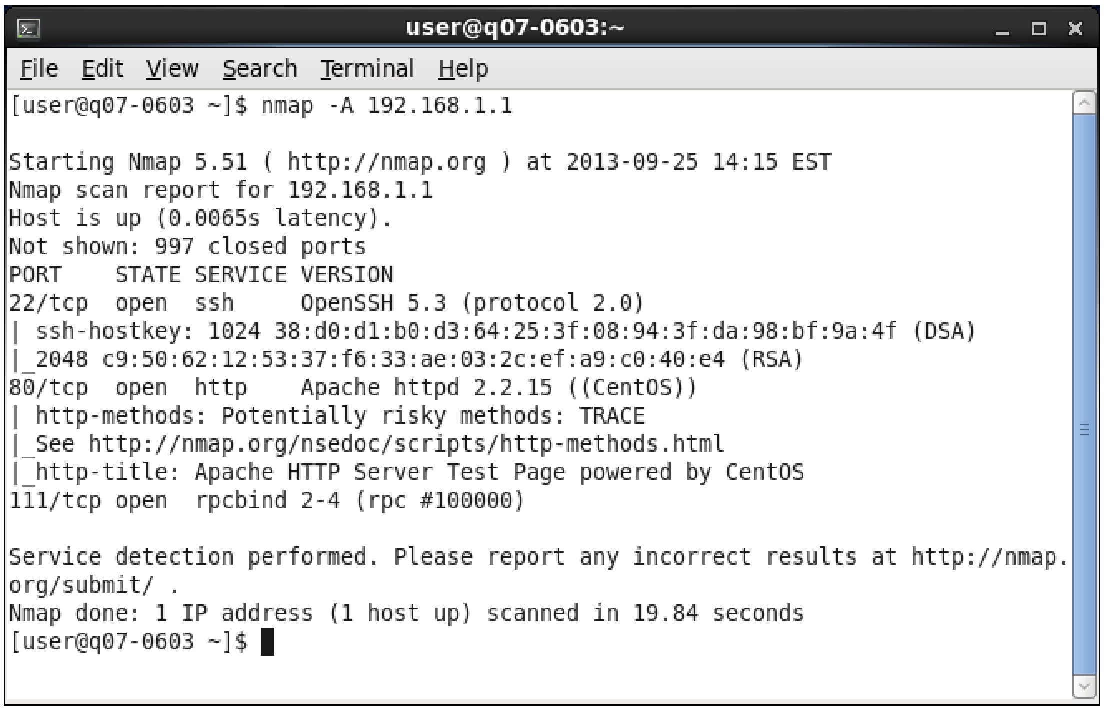
```
[user@localhost ~]$ host www.google.com
www.google.com has address 74.125.237.144
www.google.com has address 74.125.237.146
www.google.com has address 74.125.237.147
www.google.com has address 74.125.237.148
www.google.com has address 74.125.237.145
www.google.com has IPv6 address 2404:6800:4006:804::1014
[user@localhost ~]$
```

Figure D.11: sample output for host

nmap

Usage: too many to list, see the manual page.

Network Scanner/Mapper. Nmap sends specially crafted packets out onto the network in order to gather information about it. It can discover hosts on the network, scan for open ports on a host, and even detect the operating system.



```
[user@q07-0603 ~]$ nmap -A 192.168.1.1

Starting Nmap 5.51 ( http://nmap.org ) at 2013-09-25 14:15 EST
Nmap scan report for 192.168.1.1
Host is up (0.0065s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 5.3 (protocol 2.0)
| ssh-hostkey: 1024 38:d0:d1:b0:d3:64:25:3f:08:94:3f:da:98:bf:9a:4f (DSA)
|_2048 c9:50:62:12:53:37:f6:33:ae:03:2c:ef:a9:c0:40:e4 (RSA)
80/tcp  open  http    Apache httpd 2.2.15 ((CentOS))
| http-methods: Potentially risky methods: TRACE
|_See http://nmap.org/nsedoc/scripts/http-methods.html
|_http-title: Apache HTTP Server Test Page powered by CentOS
111/tcp open  rpcbind 2-4 (rpc #100000)

Service detection performed. Please report any incorrect results at http://nmap.
org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.84 seconds
[user@q07-0603 ~]$
```

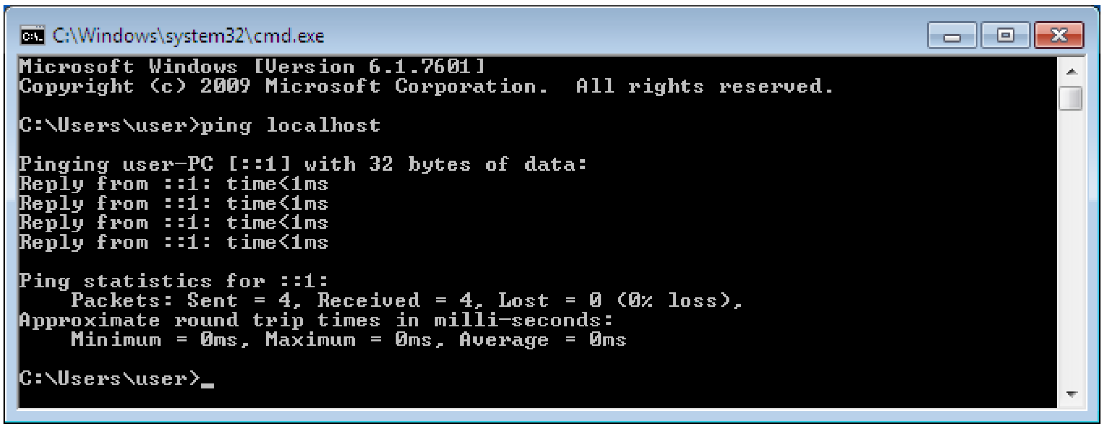
Figure D.12: sample output for nmap

D.5.4 Common Tools

ping

Usage: `ping <host>`

ping will send an icmp packet to a given host/address

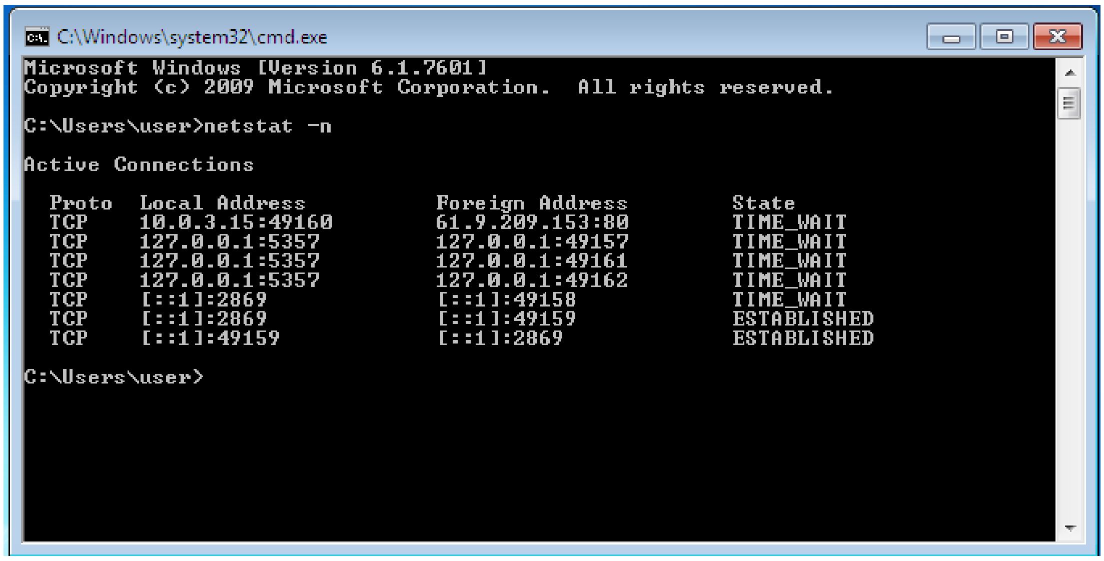


```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\user>ping localhost

Pinging user-PC [::1] with 32 bytes of data:
Reply from ::1: time<1ms
Reply from ::1: time<1ms
Reply from ::1: time<1ms
Reply from ::1: time<1ms

Ping statistics for ::1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\user>_
```

Figure D.13: sample output for ping

netstat

Display network statistics. For a full list of options see http://en.wikipedia.org/wiki/Netstat



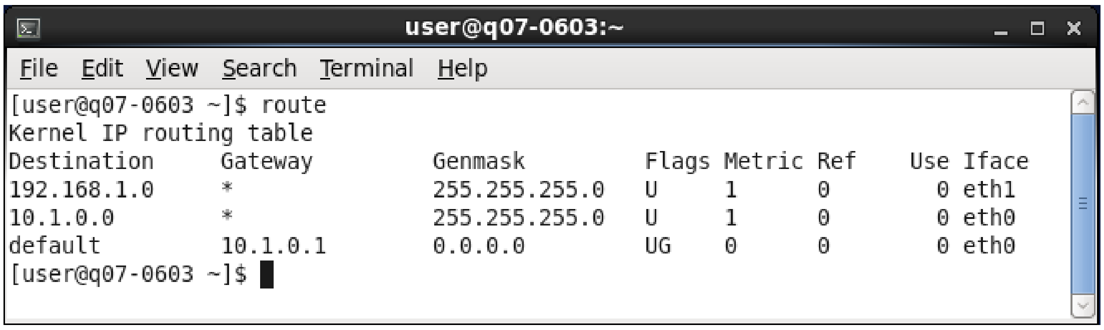
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\user>netstat -n

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    10.0.3.15:49160        61.9.209.153:80        TIME_WAIT
  TCP    127.0.0.1:5357         127.0.0.1:49157        TIME_WAIT
  TCP    127.0.0.1:5357         127.0.0.1:49161        TIME_WAIT
  TCP    127.0.0.1:5357         127.0.0.1:49162        TIME_WAIT
  TCP    [::1]:2869             [::1]:49158            TIME_WAIT
  TCP    [::1]:2869             [::1]:49159            ESTABLISHED
  TCP    [::1]:49159            [::1]:2869             ESTABLISHED

C:\Users\user>
```

Figure D.14: sample output for netstat

route

View/Modify Routing Table

Running route without any arguments will display the routing table under Linux, while running `route print`will do the same in Windows. You can add to the routing table by running `route add <destination> mask <mask> <gatewary>` under Windows and `route add -host <destination> netmask <mask> gw <gateway>` under Linux.



```
[user@q07-0603 ~]$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     1      0        0 eth1
10.1.0.0        *               255.255.255.0   U     1      0        0 eth0
default         10.1.0.1        0.0.0.0         UG    0      0        0 eth0
[user@q07-0603 ~]$
```

Figure D.15: sample output for route

D.6 Exercise 6 - DHCP: Dynamic Host Configuration Protocol

D.6.1 Introduction

A DHCP server allows the dynamic allocation of IP addresses to clients without an IP address without having to configure those computers manually.

When a new computer is connected to the network it broadcasts (sends to address 255.255.255.255) a DHCPDISCOVER message onto the network which is picked up by any DHCP servers on the network. the DHCP server(s) then reserves an IP address and broadcasts a DHCPOFFER message containing the reserved ip address and the hardware address of the new computer.

The new computer receives the offer(s), accepts *one* offer and broadcasts a DHCPREQUEST message requesting the IP address of that particular offer. This also tells all DHCP servers on the network which offer the new computer accepted (the other DHCP servers withdraw their offers). The DHCP server whose offer was offer was excepted then replies with a DHCPACK message acknowledging the request and containing the lease time and any other information the new computer needs. The new computer receives this and configures its network interface accordingly.

In this Experiment you will create a DHCP server on one of the machines. This Machine you will give a static IP address. You will then configure the DHCP server to give a range of IP addresses to DHCP enabled computers. You will then enable DHCP on the other computers. You will have successfully completed this exercise when the other 3 computers are automatically assigned IP addresses.

D.6.2 Methodology

Note: You will have to log in to the root user `su` in order to modify system files. The root Password is `password`. It is recommended you use `gedit` text editor to edit system files.

1 - Connect to Linux machine 1 as it will run our DHCP server

2 - Examine `man dhcpd` for configuration details.

3 - Edit the configuration file `/etc/dhcpd/dhcpd.conf` and add the following:

```
 default-lease-time 60;
max-lease-time 60;
ddns-update-style interim;
option subnet-mask 255.255.255.0;

subnet 192.168.1.0 netmask 255.255.255.0
{
interface eth0;
range 192.168.1.150 192.168.1.200;
}
```

Explanation of parameters:

default lease time Default time than an allocated IP is valid in seconds

max-lease-time Maximum time than an allocated IP is valid in seconds

ddns-update-style = update Update the DNS server whenever a lease is updated (not used here)

option subnet-mask sets the subnet mask

interface The network interface for the DHCP server to operate on. In this case the server is restricted to eth0.

subnet, netmask, range This defines the network the DHCP operates on, and the range of ip addresses that are handed out to clients

Note: Parameters outside the subnet definition are considered global and apply to all subnet definitions while parameters inside the subnet definition are local to that definition.

Note: If you want it is possible to allocate a fixed ip address to a certain hardware address. e.g:

```
 host Q6-4101
{
hardware ethernet 76:75:89:32:67:3C;
fixed address 192.168.11.300;
}
```

This will make the dhcp server assign an IP address of 192.168.11.300 to the machine with the hardware address of 76:75:89:32:67:3C.

4 - Configure the first interface (eth0) so Linux-1 to have a static ip of 192.168.1.1. This will be the network interface that the DHCP server will be listening on (as shown in the configuration above).

5 - Start the DHCP server by running ¡code¿service dhcpd start¡/code¿. If `[FAILED]` is shown double check you configuration and make sure the machine has a static IP. You can also check `/var/log/messages` for any dhcpd errors. You can do this by running `tail -n 20 /var/log/messages` (must be run as root).

6 - Windows-1 and Linux-2 will serve as our DHCP clients so you must configure them to use DHCP. You can do this in the same place where you assigned static IPs, but this time you select 'obtain an ip address automatically' in Windows and change Method to 'Automatic (DHCP)' in Linux.

7 - Ensure that the client machines are all assigned IP addresses. these address should be 192.168.1.150, 192.168.1.151 and 192.168.1.152, or something similar. Windows strangely tends to assign a (usually) useless ip address to an interface that cannot connect to a network. If this is the case you may have to disable the interface and re-enable it (right-click).

You will have complete this exercise when all interfaces on all machines (except the interface the DHCP is listening on) are assigned an IP address by the DHCP server.

D.7 Exercise 7 - Windows File Sharing: NetBIOS Protocol

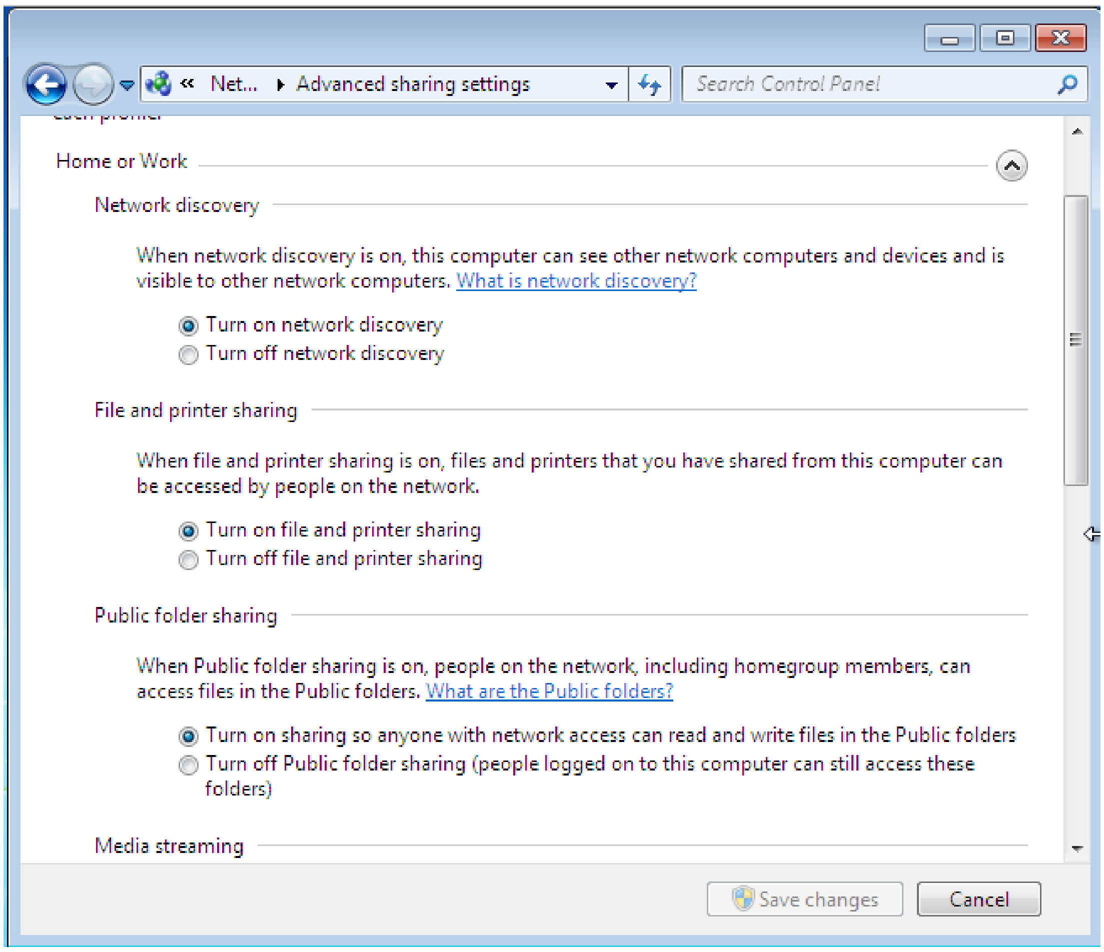
D.7.1 Introduction

In this exercise you will share a folder on one Windows machine, and access it on another Windows machine. You will have successfully completed this exercise when you are able to share files from one Windows machine to another.

D.7.2 Methodology

1 - Click start and then right-click computer then select properties. Ensure that the workgroup is named WORKGROUP.

2 - Ensure that the machine has file sharing turned on. To do this open the Network and Sharing Centre and select 'Change advanced sharing settings'. Make sure that 'Turn on file and printer sharing', 'Turn on network discovery', 'Turn off password protected sharing' are selected for public networks. Do this for both Windows-1 and Windows-2.



Home or Work

Network discovery

When network discovery is on, this computer can see other network computers and devices and is visible to other network computers. What is network discovery?

[✓ Turn on network discovery]
Turn off network discovery

File and printer sharing

When file and printer sharing is on, files and printers that you have shared from this computer can be accessed by people on the network.

[✓ Turn on file and printer sharing]
Turn off file and printer sharing

Public folder sharing

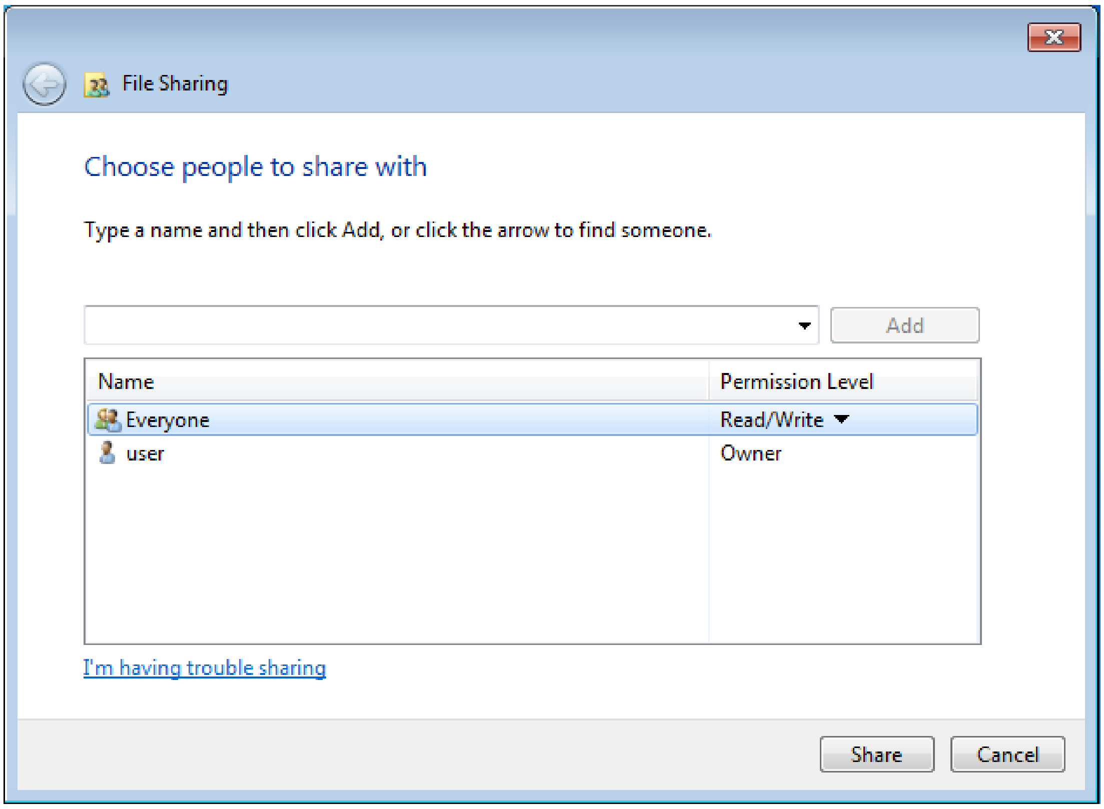
When Public folder sharing is on, people on the network, including homegroup members, can access files in the Public folders. What are the Public folders?

[✓ Turn on sharing so anyone with network access can read and write files in the Public folders]
Turn off Public folder sharing (people logged on to this computer can still access these folders)

Media streaming

Figure D.16: Enable File and Folder Sharing

3 - Create a new folder on the desktop. To share this folder right-click it and select properties. Select the sharing tab and click 'share'. You can choose which users you can share the folder with. For this exercise add 'Everyone' from the drop-down box and click add then change the permission level for 'Everyone to 'Read/Write'. Click Share then Done.



Choose people to share with

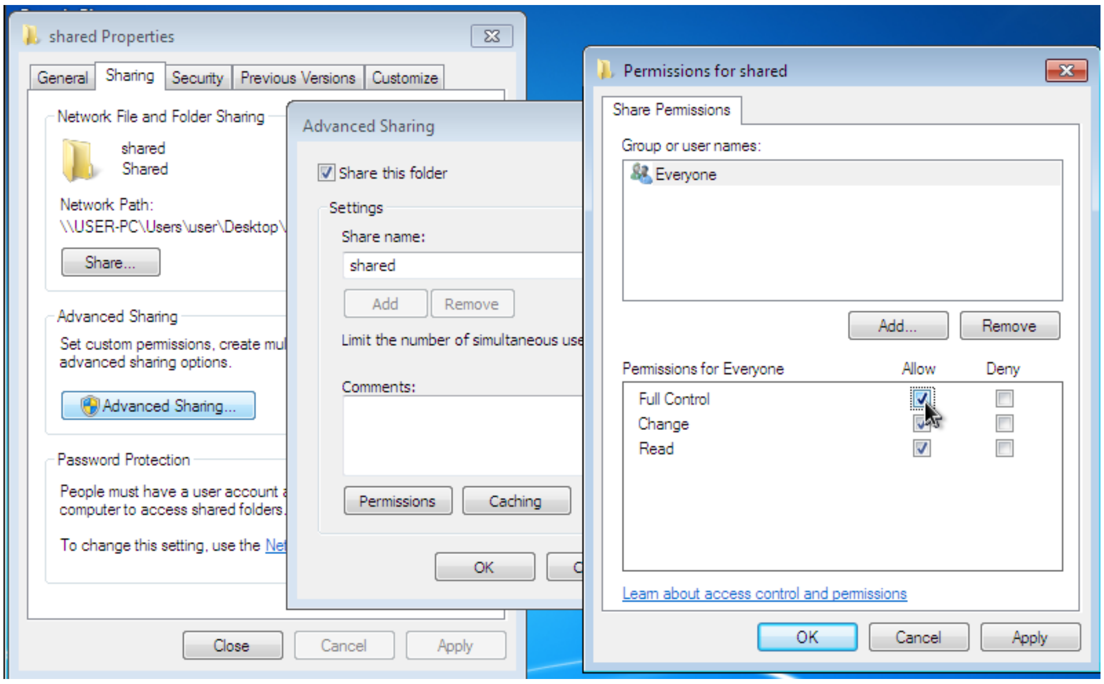
Type a name and then click Add, or click the arrow to find someone.

Name	Permission Level
Everyone	Read/Write
user	Owner

I'm having trouble sharing

Figure D.17: Selecting users who are allowed access to shared folder

Also Click 'Advanced Sharing' and make sure 'Share this folder' is checked. Click permissions and make sure that 'Everyone' has full control.



Network File and Folder Sharing

shared
Shared

Network Path:
\\USER-PC\Users\user\Desktop\

Advanced Sharing

Set custom permissions, create mul
advanced sharing options.

Password Protection

People must have a user account a
computer to access shared folders.

To change this setting, use the Net

[✓] Share this folder

Settings

Share name:
shared

Limit the number of simultaneous use

Comments:

Group or user names:
Everyone

Permissions for Everyone	Allow	Deny
Full Control	✓	
Change	✓	
Read	✓	

Learn about access control and permissions

Figure D.18: Advanced sharing options

If you're asked to turn of file and folder sharing select yes.

4 - On the other Windows machine open Windows explorer and select 'Network' from the side menu. From here you should be able to navigate to the shared folder. If you cannot see the folder try typing \\`<ip-address>`\ into the address bar where `<ip-address>` is the IP address of the machine with the shared folder (you will need to find the IP address yourself).Test that you can read and write to the folder by creating a new file. If you cannot read or write to the folder check the sharing permissions.

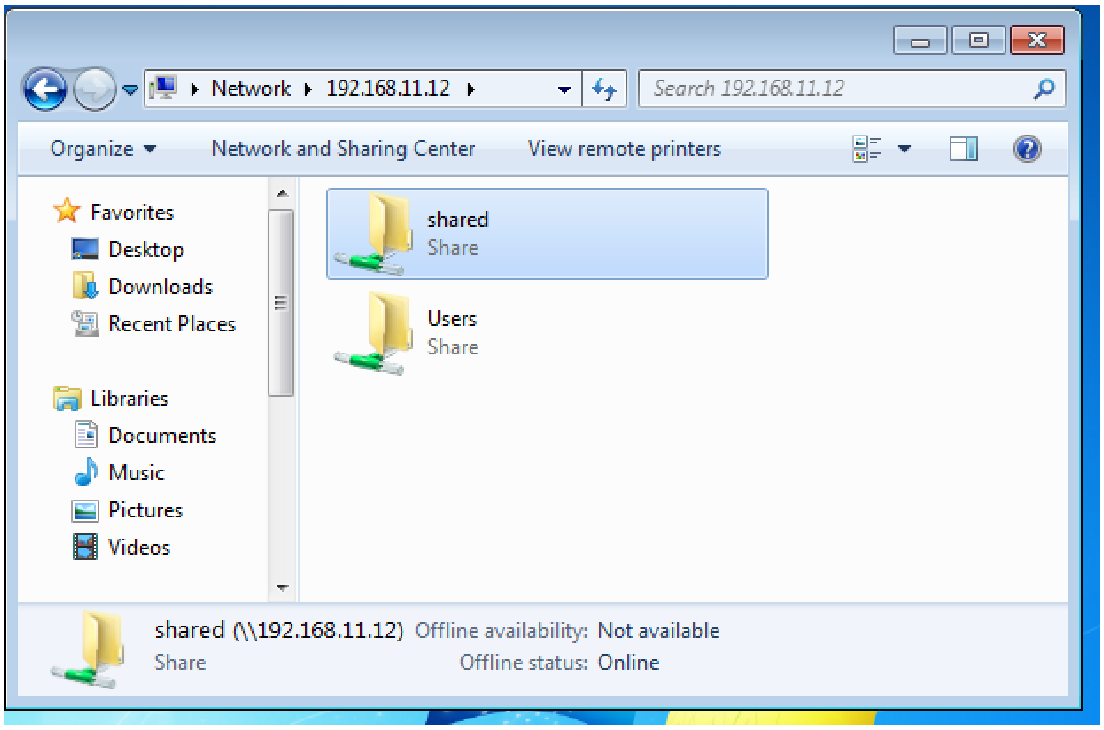


Figure D.19: Connecting to shared folder

5 - Right-click the share and select 'Map Network Drive', and assign the share a drive letter. You should now be able to open the shared folder from My Computer. Right-click the network drive and select disconnect to remove the drive.

The following does the same as the above steps except using the command line. 6 - Run `net view<` to list windows computers on the network

7 - Run `net use /? | more` to learn the syntax on how to assign a drive letter to a share ¡p/¿ 8 - Assign a drive letter to the share by running `net use <drive-letter>:` \\`<computer-name>`\`<share-name>`, where `<drive-letter>` is the drive letter you want to assign, `<computer-name>` is the name or IP address of the machine containing

the shared folder, and `<share-name>` is the name of the share. The shared folder should now appear in My Computer as a network drive.

9 - List current shares by running `net use`

10 - Disconnect the share by running `net use <drive-letter>/delete`

D.8 Exercise 8 - Unix-Windows File Sharing - Samba

D.8.1 Introduction

Samba allows Unix files to be accessible on Windows using the SMB (Server Message Block) protocol, also known as CIFS (Common Internet FileSystem). In this exercise you will share a folder on a Linux machine, and access it on a windows machine. You will have successfully completed this exercise when you are able to share files from the Linux machine to a Windows machine.

D.8.2 Methodology

Note: You will have to login as root in order to modify system files. the password for the root user is `password`

1 - Connect to Linux-1, which will run our samba server.

2 - Examine `man smb.conf` for configuration details.

3 - Rename `/etc/samba/smb.conf` to `/etc/samba/smb.conf.backup` using `mv /etc/samba/smb.conf /etc/samba/smb.conf.backup` and create a new blank smb.conf by running `touch /etc/samba/smb.conf`

4 - Edit `/etc/samba/smb.conf` using gedit and add the following:

```
 [global]
workgroup = WORKGROUP
Server String = I am a Samba Server
security = user
Map to guest = Bad User
username map = /etc/samba/smbusers
```

Explanation of parameters

workgroup Windows NT Domain Name/Workgroup Name

Server String server description

security = user security mode samba runs in, setting it to user mode makes samba authenticate incoming connections

Map to guest = Bad User makes samba treat any unauthenticated users as a guest user

username map The file listing the users that samba uses for authentication

5 - make sure that `nobody = guest` exists in `/etc/samba/smbusers`. If `smbusers` doesn't exist then create it and add the line to it. `nobody = guest` will link the samba guest account to the Linux account 'nobody'.

6 - Create folder `/home/public`. This will be the folder you will be sharing. Make sure that it has its permissions for other set to read, write and executable by running `chmod o+rwx /home/public` (**important!**)

7 - Add the following share definition to `/etc/samba/smb.conf`:

```
 [share]
comment = Public Folder
path = /home/public
writeable = yes
browsable = yes
public = yes
only guest = yes
```

Explanation of parameters

comment description of the share definition

path path to the folder to be shared

writeable whether or not connected users can write to the shared folder

browsable whether or not connected users can navigate the shared folder

public whether the guest account is allowed access (guest ok = yes/no can be used here instead)

only guest whether or not only the guest account is allowed access

8 - Start the server by running `service smb start` (as root). You can stop the server by running `service smb stop` and check its status by running `service smb status`

9 - Check that the server works. On a windows machine open windows explorer and enter \\`<ip-address>`\ into the address bar, where `<ip-address>` is the IP address of the machine that the server is running on. To find the ip address type `ifconfig` on the machine running the server.

10 - If any "cannot connect" or "permission denied" errors occur, check the status of the server in `/var/log/samba/log.smbd`. Also double check `/etc/samba/smb.conf` for any errors. Also check the permissions of the shared folder on the server. the folder and all file inside should have read, write and execute permissions. If not then change the permissions using the chmod command.

D.9 Exercise 9 - Unix File Sharing - NFS: Network Filesystem

D.9.1 Introduction

In this exercise you will share a folder on a Linux machine, and access it on another Linux machine. You will have successfully completed this exercise when you are able to share files from one Linux machine to another.

D.9.2 Methodology

Note: You will have to log in to the root user `su` in order to modify system files. The root Password is `password`. It is recommended you use `gedit` text editor to edit system files.

1 - Connect to Linux machine 1

2 - create folder `/home/public` by running `mkdir -p /home/public`. This will be the folder you will be sharing.

3 - start the NFS server by running `service nfs start`

4 - In order to make a folder accessible to a client it must be "exported" by the server. To do this we must make an entry into `/etc/exports`. For the full set of options check the exports man page `man exports`.

5 - Edit `/etc/exports` and add this line:

```
/home/public 192.168.11.*(rw,sync,no_root_squash,no_subtree_check)
```

Explanation of parameters

/home/public folder to be shared

192.168.11.* specifies ip addresses that are allowed to connect (in this case the entire 192.168.11.xxx subnet)

rw allow both read and write requests

sync reply to new requests only after previous changes have been completed on the server system

no_root_squash allows the client's root account to be mapped to server's root account (by default requests made by the client's root account are mapped to `nobody` on the server, an action called root squashing)

no_subtree_check disable checking whether the file/folder is a descendant of the exported directory

6 - `exportfs` is the program that maintains the export table for the nfs server. Now we have to update the export table with the new entry we put into ¡code¿/etc/exports¡/code¿. Run `exportfs -a` to do this.

7 - On the other Linux machine create a mount point. The concept of mount points is that a partition on a physical drive (or in our case a location on a remote computer) us bound to a folder on the filesystem. For example if you mount sda1 (first partition of physical drive sda) to folder `/media/drive/` the contents of that partition is accessible from that folder. Windows uses the same concept except it uses specialised mount points (drive letters).

So lets create a mount point `/mnt/remotedir/`, (run `mkdir -p /mnt/remotedir`), and then mount the shared folder on the remote system to the mount point that we just created. Run `mount <ip-address>:/home/public /mnt/remotedir`.

8 - You should now be able to access the shared folder by accessing `/mnt/remote/dir`. Take note that Linux permissions play a big role here. By default the permissions of `/mnt/remotedir` only allow write access for root. To allow a normal user to write to files in the shared folder you need to change the permissions of the mount point (`/mnt/remotedir`). You can give everyone write permissions by running `chmod o+rwx /mnt/remotedir`.

D.10 Experiment 10 - Apache Web Server

D.10.1 Introduction

In this exercise you will learn how to set up a web page using the apache web server.

D.10.2 Methodology

Note: You will have to log in to the root user `su` in order to modify system files. The root Password is `password`. It is recommended you use `gedit` text editor to edit system files.

Starting a web server

1 - First create a test html file called `index.html` in `/var/www/html`. This is the default document root for the apache web server in CentOS.

2 - Type the following into your `index.html` file:

```
 <!-- doctype html -->

<html>
<body>
<h1 align=center>
<font color="Blue"> Hello World </font>
</h1>
</body>>
</html>
```

You can add anything you want to this html file.

3 - Now test whether the server actually works. Run `apachectl start` in the terminal to start the server. Open a web browse and type `localhost` or `http://127.0.0.1` in the address bar. The web page you created should come up.

4 - Now try to connect from another machine by opening a web browser and typing the IP address of the server into the address bar. You may have to adjust the firewall on the server to allow http connections (port 80).

Access Permissions

5 - Now we are going to restrict access to the web server. Search the main apache configuration file `/etc/httpd/conf/httpd.conf` for the entry

```
<Directory "/var/www/html">
```

Find the section dealing with access mermissions:

```
 Order allow,deny
Allow from all
```

This rule says that the "allow" rules are processed first, followed by the "deny" rules. In this case everybody is allowed permission to view the web page.

6 - Change this section to only allow access from a particular machine:

```
 Order deny,allow
deny from all
allow from <ip-address>
```

where `<ip-address>` is the IP address of the machine you want to allow access to the web server.

7 - Restart the web server (`apachectl restart`). Verify that the only the machine specified in the above rule is the only one allowed access to the web page.

password protection

8 - Now we are going to configure the web server to require a username and password for access to the web page. First we have to create a password file which will contain

the usernames and passwords for the web server. The `htpasswd` program does this for us.

```
htpasswd -cm /etc/httpd/conf/htpasswd.conf fred
```

will create a new password file called `htpasswd.conf` in `/etc/httpd/conf/`, prompt for a password, create an entry for the user `fred` in that password file and encrypt the password with the md5 hash algorithm and store it alongside the username.

Passwords are always encrypted with a hash algorithm. This ensures that passwords cannot be decrypted given the encrypted password. The example above will yield an entry similar to:

```
fred:$apr1$Iw12IIOq$PX67jCzpU3uJlGdzZdvgw1
```

You can also check the manual page for `htpasswd` (`man htpasswd`) for more details on the program.

9 - Now reopen the `http.conf` file and again search for the section:

```
<Directory "/var/www/html">
```

and search for the line:

```
AllowOverride None
```

Change this to:

```
AllowOverride AuthConfig
```

10 - Now go back to the document root (`/var/www/html/`) and create a new file called `.htaccess` containing:

```
 AuthName "Realm Name"
AuthType basic
AuthUserFile "/etc/httpd/conf/htpasswd.conf"
require valid-user
```

11 - Restart the server, and check that the browser requests a password for all we pages

in the directory.

Appendix E

Web Interface Source Code

E.1 Makefile

```
all:
mkdir -p ../project
cp index.php ../project/
m4 state.m4 > ../project/state.php
m4 feedback.m4 > ../project/feedback.php
m4 exercises.m4 > ../project/exercises.php
m4 exercise1.m4 > ../project/exercise1.php
cp exercise1-* ../project
m4 exercise2.m4 > ../project/exercise2.php
m4 exercise3.m4 > ../project/exercise3.php
cp exercise3-* ../project
m4 exercise4.m4 > ../project/exercise4.php
m4 exercise5.m4 > ../project/exercise5.php
cp exercise5-* ../project
m4 exercise6.m4 > ../project/exercise6.php
m4 exercise7.m4 > ../project/exercise7.php
cp exercise7-* ../project
m4 exercise8.m4 > ../project/exercise8.php
m4 exercise9.m4 > ../project/exercise9.php
m4 exercise10.m4 > ../project/exercise10.php
cp server-o* ../project/

clean:
rm -rf ../project
```

E.2 defines.m4

```
define(‘XML_HEADER’, ‘<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
           "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">’)dnl
define(‘STYLE’, ‘    <style>
    div{
        display:inline-block;
    }
    body{
        margin-left: auto;
        margin-right: auto;
        max-width: 70em;
    }
    </style>')dnl
define(`FUNCTIONS', `<?PHP
function getState()
{
    $statein = "";
    $statein = @file_get_contents("/media/sf_state/status");
    return $statein;
}

function displayState($statein)
{
    $statusD = "server-offline.png";
    $status1 = "server-offline.png";
    $status2 = "server-offline.png";
    $status3 = "server-offline.png";
    $status4 = "server-offline.png";
    if (strpos($statein,"D") !== FALSE)
        $statusD = "server-online.png";
    if (strpos($statein,"1") !== FALSE)
        $status1 = "server-online.png";
    if (strpos($statein,"2") !== FALSE)
        $status2 = "server-online.png";
    if (strpos($statein,"3") !== FALSE)
        $status3 = "server-online.png";
    if (strpos($statein,"4") !== FALSE)
        $status4 = "server-online.png";
    echo "<div><img src=\"$statusD\" alt=\"DHCP\"/><br/>DHCP</div>";
    echo "<div><img src=\"$status1\" alt=\"Linux-1\"/><br/>Linux-1</div>";
    echo "<div><img src=\"$status2\" alt=\"Linux-2\"/><br/>Linux-2</div>";
    echo "<div><img src=\"$status3\" alt=\"Windows-1\"/><br/>Windows-1</div>";
    echo "<div><img src=\"$status4\" alt=\"Windows-2\"/><br/>Windows-2</div>";
    echo "<br/>";
}

function checkState($state)
{
    $statein = getState();
    $referer = basename( $_SERVER["PHP_SELF"] );
    $stateD = false;
    $state1 = false;
    $state2 = false;
    $state3 = false;
    $state4 = false;
    if (preg_match("/D/", $statein) === preg_match("/D/", $state))
        $stateD = true;
    if (preg_match("/1/", $statein) === preg_match("/1/", $state))
        $state1 = true;
    if (preg_match("/2/", $statein) === preg_match("/2/", $state))
        $state2 = true;
    if (preg_match("/3/", $statein) === preg_match("/3/", $state))
        $state3 = true;
    if (preg_match("/4/", $statein) === preg_match("/4/", $state))
        $state4 = true;
    if ($stateD === false
     || $state1 === false
     || $state2 === false
     || $state3 === false
     || $state4 === false)
    {
        echo "The current lab state does not match the exercise requirements.<br/>";
        $stateLink = "<a href=\"./state.php?state=$state&amp;referer=$referer\">Here</a>"
        echo "Click $stateLink to set the lab state.";
    }
    else
        echo "The current lab state matches the exercise requirements";
}
?>')dnl
```

E.3 feedback.m4

```
include(`defines.m4')dnl
XML_HEADER
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<?php
$exercise = "";
$complete1 = "";
$complete2 = "";
$objectives1 = "";
$objectives2 = "";
$content1 = "";
$content2 = "";
$comments = "";
if (array_key_exists("exercise",$_GET))
{
    $exercise = $_GET["exercise"];
}
if (array_key_exists("complete1", $_POST))
{
    $complete1 = $_POST["complete1"];
}
if (array_key_exists("complete2", $_POST))
{
    $complete2 = $_POST["complete2"];
}
if (array_key_exists("objectives1", $_POST))
```

```
{
    $objectives1 = $_POST["objectives1"];
}
if (array_key_exists("objectives2", $_POST))
{
    $objectives2 = $_POST["objectives2"];
}
if (array_key_exists("content1", $_POST))
{
    $content1 = $_POST["content1"];
}
if (array_key_exists("content2", $_POST))
{
    $content2 = $_POST["content2"];
}
if (array_key_exists("comments", $_POST))
{
    $comments = $_POST["comments"];
}
if (!empty($_POST))
{
    shell_exec("echo \"$complete1, $complete2, $objectives1, $objectives2, $content1, $co
    header("Location: exercises.php");
}
?>
<head>
  <title>Feedback - Exercise <?php printf($exercise)?></title>
</head>
<body>
<h1> Feedback - Exercise <?php printf($exercise)?></h1>

<form name="feedback" action="feedback.php?exercise=<?php printf($exercise)?>" method="po
Were you able to complete the exercise?<br/>
<input type="radio" name="complete1" value="yes"/>yes
<input type="radio" name="complete1" value="no"/>no
```

```
<p/>
If you were unable to complete the exercise, please provide information on what went wron
<textarea rows="10" cols="100" name="complete2"/></textarea>
<p/>
Do you feel you understood/achieved the objectives of the exercise?<br/>
<input type="radio" name="objectives1" value="1"/>1
<input type="radio" name="objectives1" value="2"/>2
<input type="radio" name="objectives1" value="3"/>3
<input type="radio" name="objectives1" value="4"/>4
<input type="radio" name="objectives1" value="2"/>5
<p/>
If you did not understand/achieve the objectives of the exercise, what aspects of the exe
<textarea rows="10" cols="100" name="objectives2"/></textarea>
<p/>
Do you feel the exercise was enough to achieve the objectives?<br/>
<input type="radio" name="content1" value="1"/>1
<input type="radio" name="content1" value="2"/>2
<input type="radio" name="content1" value="3"/>3
<input type="radio" name="content1" value="4"/>4
<input type="radio" name="content1" value="2"/>5
<p/>
What aspects of the exercise would you implement differently?<br/>
<textarea rows="10" cols="100" name="content2"/></textarea>
<p/>
Other comments.<br/>
<textarea rows="10" cols="100" name="comments"/></textarea>
<p/>
<input type="submit" value="Submit"> <INPUT type="reset"/>
</form>
</body>
</html>
```

E.4 state.m4

```
include(‘defines.m4’)dnl

<?php
$statein = "";
$stateout = "";
if (array_key_exists("state",$_GET))
{
    $statein = $_GET["state"];
}
if (strpos($statein,"D") !== FALSE)
{
    $stateout .= " D";
}
if (strpos($statein,"1") !== FALSE)
{
    $stateout .= " 1";
}
if (strpos($statein,"2") !== FALSE)
{
    $stateout .= " 2";
}
if (strpos($statein,"3") !== FALSE)
{
    $stateout .= " 3";
}
if (strpos($statein,"4") !== FALSE)
{
    $stateout .= " 4";
}
shell_exec("echo \"$stateout\" > /media/sf_state/state");
shell_exec("echo \"$stateout\" > /media/sf_state/status");
if (array_key_exists("referer",$_GET))
```

```
{
    $referer = $_GET["referer"];
    header("Location: $referer");
}
?>
```

E.5 exercises.m4

```
include(‘defines.m4’)dnl
XML_HEADER
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <title>Exercises</title>
STYLE
</head>
<body>
<h1>Networking Remote Access Laboratory</h1>
<h2>Introduction</h2>
Welcome to the Networking Remote Access Laboratory. This Laboratory is designed to allow

<h3>This lab is experimental. There will be bugs.</h3>

<h2>Server Status</h2>
Refresh the Page to update the server status.<br/>
<strong>Note:</strong> Linux Machines - Username: user, Password: password
<p/>
<?php
$statein = getState();
displayState($statein);
$referer = basename( $_SERVER["PHP_SELF"] );
$stateLink = "<a href=\"./state.php?referer=$referer\">Here</a>";
echo "Click $stateLink to shutdown the lab machines.";
?>
```

```
<h2>Exercises</h2>
  <a href="./exercise1.php">Exercise 1</a> - Static IP Addresses
  <a href="./feedback.php?exercise=1">Feedback</a><br/>
  <a href="./exercise2.php">Exercise 2</a> - Routing Between Networks
  <a href="./feedback.php?exercise=2">Feedback</a><br/>
  <a href="./exercise3.php">Exercise 3</a> - Packet analysis using Wireshark
  <a href="./feedback.php?exercise=3">Feedback</a><br/>
  <a href="./exercise4.php">Exercise 4</a> - Introduction to the Linux Operating System
  <a href="./feedback.php?exercise=4">Feedback</a><br/>
  <a href="./exercise5.php">Exercise 5</a> - Networking CLI Tools
  <a href="./feedback.php?exercise=5">Feedback</a><br/>
  <a href="./exercise6.php">Exercise 6</a> - DHCP Server
  <a href="./feedback.php?exercise=6">Feedback</a><br/>
  <a href="./exercise7.php">Exercise 7</a> - Windows File Sharing - NetBIOS protocol
  <a href="./feedback.php?exercise=7">Feedback</a><br/>
  <a href="./exercise8.php">Exercise 8</a> - Unix-Windows File Sharing - Samba
  <a href="./feedback.php?exercise=8">Feedback</a><br/>
  <a href="./exercise9.php">Exercise 9</a> - Unix File Sharing - NFS: Network File System
  <a href="./feedback.php?exercise=9">Feedback</a><br/>
  <a href="./exercise10.php">Exercise 10</a> - Apache Web Server
  <a href="./feedback.php?exercise=10">Feedback</a><br/>

</body>
</html>
FUNCTIONS
```

E.6 exercise1.m4

```
include(‘defines.m4’)dnl
XML_HEADER
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <title>Exercise 1 - Static IP Addresses</title>
  STYLE
```

```
</head>
<body>
  <h1>Exercise 1 - Static IP Addresses</h1>
<a href="./exercises.php">Exercise List</a>
  <h2>Required Lab Machines</h2>
<?PHP
displayState("1234");
checkState("1234");
?>
  <h2>Introduction</h2>
  <br/>
In this Experiment you will create a small network of consisting of four
computers. You will be assigning each computer an IP address manually and
then you will test the network by making sure that each computer can
connect to another.

  <h2>Methodology</h2>
1 - To configure a Windows machine with a static IP first click on the
network icon in the system tray then select 'Open Network and Sharing
Centre' then 'Change Adapter Settings'. Alternatively type 'View Network
Connections' in the search area of the start menu. Right-click the
network interface you want to configure then select properties. Select
'Internet Protocol Version 4 (TCP/IPv4)' then click properties.
<p/>
<div><img src=exercise1-1.png alt="Configuring a network address under Windows"/>
<br/>Configuring a network address under Windows</div>
<p/>
2 - To Configure a Linux machine right-click the network icon in the
system tray then select 'Edit Connections'. Select eth0 from the list
then select 'Edit'. Select the IPv4 Tab and change Method to Manual
<p/>
<div><img src=exercise1-2.png alt="Configuring a network address under Linux"/>
<br/>Configuring a network address under Linux</div>
<p/>
3 - Configure the <em>first</em> ethernet interface of the machines to
```

```
use the following IP addresses (leave the default gateway and dns server
blank for the moment). When applying these settings to a Linux machine
you may be asked for a password. The password is <code>password</code>.
<br/>
Linux 1 - Address: 192.168.0.101, Netmask: 225.225.225.0<br/>
Linux 2 - Address: 192.168.0.102, Netmask: 225.225.225.0<br/>
Windows 1 - Address: 192.168.0.103, Netmask: 225.225.225.0<br/>
Windows 2 - Address: 192.168.0.103, Netmask: 225.225.225.0<br/>
<p/>
4 - Click OK/Apply to apply the new settings
<p/>
5 - Confirm that each computers have the correct IP addresses. On Windows
right-click the network interface then select 'status', then click
'details'. On Linux right-click the network icon in the system tray then
select 'Connection Information' then select the 'eth0' tab.
<p/>
<div><img src=exercise1-3.png alt="Checking IP Settings"/>
<br/>Checking IP Settings</div>
<p/>
6 - Confirm that the computers can communicate with each other by running
<code>ping &lt;ip-address&gt;</code>, where <code>&lt;ip-address&gt;
</code> is the address of the computer you want to send the ICMP Packet
to. Sending ICMP Packets with the <code>ping</code> is a great way to
diagnose whether a computer on the network is running or not.

<!--<h2>Results</h2>
<a href="check.php?exercise=1">check</a> whether you have successful completed this exerc

</body>
</html>
FUNCTIONS
```

E.7 exercise2.m4

```
include(‘defines.m4’)dnl
XML_HEADER
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <title>Exercise 1 - Static IP Addresses</title>
  STYLE
</head>
<body>
  <h1>Exercise 1 - Static IP Addresses</h1>
<a href="./exercises.php">Exercise List</a>
  <h2>Required Lab Machines</h2>
<?PHP
displayState("1234");
checkState("1234");
?>
  <h2>Introduction</h2>
  <br/>
In this Experiment you will create a small network of consisting of four
computers. You will be assigning each computer an IP address manually and
then you will test the network by making sure that each computer can
connect to another.

  <h2>Methodology</h2>
1 - To configure a Windows machine with a static IP first click on the
network icon in the system tray then select 'Open Network and Sharing
Centre' then 'Change Adapter Settings'. Alternatively type 'View Network
Connections' in the search area of the start menu. Right-click the
network interface you want to configure then select properties. Select
'Internet Protocol Version 4 (TCP/IPv4)' then click properties.
<p/>
<div><img src=exercise1-1.png alt="Configuring a network address under Windows"/>
<br/>Configuring a network address under Windows</div>
```

```
<p/>
2 - To Configure a Linux machine right-click the network icon in the
system tray then select ’Edit Connections’. Select eth0 from the list
then select ’Edit’. Select the IPv4 Tab and change Method to Manual
<p/>
<div><img src=exercise1-2.png alt="Configuring a network address under Linux"/>
<br/>Configuring a network address under Linux</div>
<p/>
3 - Configure the <em>first</em> ethernet interface of the machines to
use the following IP addresses (leave the default gateway and dns server
blank for the moment). When applying these settings to a Linux machine
you may be asked for a password. The password is <code>password</code>.
<br/>
Linux 1 - Address: 192.168.0.101, Netmask: 225.225.225.0<br/>
Linux 2 - Address: 192.168.0.102, Netmask: 225.225.225.0<br/>
Windows 1 - Address: 192.168.0.103, Netmask: 225.225.225.0<br/>
Windows 2 - Address: 192.168.0.103, Netmask: 225.225.225.0<br/>
<p/>
4 - Click OK/Apply to apply the new settings
<p/>
5 - Confirm that each computers have the correct IP addresses. On Windows
right-click the network interface then select ’status’, then click
’details’. On Linux right-click the network icon in the system tray then
select ’Connection Information’ then select the ’eth0’ tab.
<p/>
<div><img src=exercise1-3.png alt="Checking IP Settings"/>
<br/>Checking IP Settings</div>
<p/>
6 - Confirm that the computers can communicate with each other by running
<code>ping &lt;ip-address&gt;</code>, where <code>&lt;ip-address&gt;
</code> is the address of the computer you want to send the ICMP Packet
to. Sending ICMP Packets with the <code>ping</code> is a great way to
diagnose whether a computer on the network is running or not.

<!--<h2>Results</h2>
```

```
<a href="check.php?exercise=1">check</a> whether you have successful completed this exerc

</body>
</html>
FUNCTIONS
```

E.8 exercise3.m4

```
include(‘defines.m4’)dnl
XML_HEADER
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <title>Exercise 1 - Static IP Addresses</title>
  STYLE
</head>
<body>
  <h1>Exercise 1 - Static IP Addresses</h1>
<a href="./exercises.php">Exercise List</a>
  <h2>Required Lab Machines</h2>
<?PHP
displayState("1234");
checkState("1234");
?>
  <h2>Introduction</h2>
  <br/>
In this Experiment you will create a small network of consisting of four
computers. You will be assigning each computer an IP address manually and
then you will test the network by making sure that each computer can
connect to another.

  <h2>Methodology</h2>
1 - To configure a Windows machine with a static IP first click on the
network icon in the system tray then select 'Open Network and Sharing
Centre' then 'Change Adapter Settings'. Alternatively type 'View Network
```

```
Connections' in the search area of the start menu. Right-click the
network interface you want to configure then select properties. Select
'Internet Protocol Version 4 (TCP/IPv4)' then click properties.
<p/>
<div><img src=exercise1-1.png alt="Configuring a network address under Windows"/>
<br/>Configuring a network address under Windows</div>
<p/>
2 - To Configure a Linux machine right-click the network icon in the
system tray then select 'Edit Connections'. Select eth0 from the list
then select 'Edit'. Select the IPv4 Tab and change Method to Manual
<p/>
<div><img src=exercise1-2.png alt="Configuring a network address under Linux"/>
<br/>Configuring a network address under Linux</div>
<p/>
3 - Configure the <em>first</em> ethernet interface of the machines to
use the following IP addresses (leave the default gateway and dns server
blank for the moment). When applying these settings to a Linux machine
you may be asked for a password. The password is <code>password</code>.
<br/>
Linux 1 - Address: 192.168.0.101, Netmask: 225.225.225.0<br/>
Linux 2 - Address: 192.168.0.102, Netmask: 225.225.225.0<br/>
Windows 1 - Address: 192.168.0.103, Netmask: 225.225.225.0<br/>
Windows 2 - Address: 192.168.0.103, Netmask: 225.225.225.0<br/>
<p/>
4 - Click OK/Apply to apply the new settings
<p/>
5 - Confirm that each computers have the correct IP addresses. On Windows
right-click the network interface then select 'status', then click
'details'. On Linux right-click the network icon in the system tray then
select 'Connection Information' then select the 'eth0' tab.
<p/>
<div><img src=exercise1-3.png alt="Checking IP Settings"/>
<br/>Checking IP Settings</div>
<p/>
6 - Confirm that the computers can communicate with each other by running
```

```
<code>ping &lt;ip-address&gt;</code>, where <code>&lt;ip-address&gt;
</code> is the address of the computer you want to send the ICMP Packet
to. Sending ICMP Packets with the <code>ping</code> is a great way to
diagnose whether a computer on the network is running or not.

<!--<h2>Results</h2>
<a href="check.php?exercise=1">check</a> whether you have successful completed this exerc

</body>
</html>
FUNCTIONS
```

E.9 exercise4.m4

```
include(‘defines.m4’)dnl
XML_HEADER
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <title>
    Exercise 4 - Introduction to the Linux Operating System
  </title>
    STYLE
</head>
<body>
  <h1>Exercise 4 - Introduction to the Linux Operating System</h1>
<a href="./exercises.php">Exercise List</a>
  <h2>Required Lab Machines</h2>
<?PHP
displayState("1");
checkState("1");
?>
  <h2>Introduction</h2>
  <br/>
This exercise (...well, not really an exercise) will give you an
```

```
introduction to the Linux filesystem as well as the Terminal and some
commonly used terminal commands. Linux is vastly different from
Windows. Windows is designed for ordinary users, as such many aspects
of the operating system are hidden. Unix (upon which Linux is based)
is the opposite, with most if not all of the operating system exposed
allowing the user a large amount of control over the system.

  <h2>The Linux Filesystem</h2>
Unlike Windows filesystems which start with a drive letter. Linux uses
a single hierarchical file structure starting at /, also called the root
directory. <em>All</em> files and folders (including those on separate
drives/partitions) are descendentslol coul of this root directory. Instead of
assigning drive letters to partitions, Linux <em>mounts</em> the
partitions to a location in the existing filesystem. You can visualise
it as attaching the top level directory of the drive/partition to an
existing folder on the Linux filesystem. So when you access that folder
you are actually accessing the drive/partition.
<p/>
The Filesystem Hierarchy Standard (FHS) defines the directory structure
of Linux systems. Most Linux distributions follow the FHS but some may
have slight deviations. The following is the directory structure
according to the FHS.
<p/>
<dl>
  <dt>/</dt>
  <dd>The root directory. This is the top level directory of the
  entire filesystem.</dd>
  <dt>/bin</dt>
  <dd>Contains Essential programs for all users.</dd>
  <dt>/boot</dt>
  <dd>Contains the Linux kernel and other files required at boot time.
  </dd>
  <dt>/dev</dt>
  <dd>All files here are representations of the system hardware. The
  Linux philosophy is that everything is a file. Reading and writing
```

```
to files in this directory will read and write to the hardware
directly.</dd>
<dt>/etc</dt>
<dd>Contains all the system configuration files.</dd>
<dt>/home</dt>
<dd>Contains all the user files. Each user gets their own folder
here called their home directory.</dd>
<dt>/lib</dt>
<dd>Libraries essential to the programs in /bin. Some 64-bit
distributions also include a /lib64 directory</dd>
<dt>/media</dt>
<dd>Contains mount points for removable media (although partitions
can be mounted to any folder, it is common practice to mount
external media to folders within /media).</dd>
<dt>/mnt</dt>
<dd>Contains mount points for temporary filesystems</dd>
<dt>/opt</dt>
<dd>Used for optional software. This directory is used less and less
in favour of /usr/local</dd>
<dt>/proc</dt>
<dd>A virtual filesystem which represents the internals of the
Linux kernel.</dd>
<dt>/root</dt>
<dd>Home directory for the root user</dd>
<dt>/sbin</dt>
<dd>Contains essential system binaries</dd>
<dt>/srv</dt>
<dd>Contains files to be used by services</dd>
<dt>/tmp</dt>
<dd>Contains temporary files which are usually deleted at boot time
</dd>
<dt>/usr</dt>
<dd>Contains read-only user data, user applications (/usr/bin), user
libraries (/usr/lib), documentation and source code</dd>
<dt>/var</dt>
```

```
  <dd>Contains files that are expected to continually change such as
  log files, spooling data, ...</dd>
</dl>
<p/>
  <h2>Linux Permissions</h2>
  <h3>Unix Permissions</h3>
Traditional Unix permissions come in three classes: Owner, Group and
Other. Within each class there are three distinct permissions: read,
write and execute. This comes to a total of nine different types of
permissions that <em>all</em> files and directories have associated
with them. Lets do an example:
<p/>
1 - Open a terminal (<code>Applications->System Tools->Terminal
</code>). Note that the terminal will start in the users home
directory (<code>/home/&lt;user-name&gt;/</code>, also called <code>
~/</code> for the current user).
<p/>
2 - Create a file called <code>test</code> in the current directory by
typing <code>touch test</code>.
<p/>
3 - now list all the files in the current directory and their
permissions by typing <code>ls -l</code>. You should get something
similar to:
<pre>
total 32
drwxr-xr-x. 2 user user 4096 Jun 24 01:57 Desktop
drwxr-xr-x. 2 user user 4096 Jun 24 01:57 Documents
drwxr-xr-x. 2 user user 4096 Jun 24 01:57 Downloads
drwxr-xr-x. 2 user user 4096 Jun 24 01:57 Music
drwxr-xr-x. 2 user user 4096 Jun 24 01:57 Pictures
drwxr-xr-x. 2 user user 4096 Jun 24 01:57 Public
drwxr-xr-x. 2 user user 4096 Jun 24 01:57 Templates
-rw-rw-r--. 1 user user    0 Jun 25 23:29 test
drwxr-xr-x. 2 user user 4096 Jun 24 01:57 Videos
</pre>
```

```
For a full explanation of the output see http://en.wikipedia.org/wiki/Ls
<p/>
The letters to the left (minus the very first character, which is set to
<code>d</code> to indicate a directory) shows the
permissions that are set for the file. Our test file shows the
permissions: <code>rw-rw-r--</code>. The first three character show
permissions for the owner of the file, the next three show permissions
for the group associated with that file, and the last three are
permissions for everybody else. As you can see our test file has
read/write access for both the user and group, and read-only access
for everyone else.
<p/>
Take note that directories have their executable permissions set. This
allows you to browse through the directories.
  <h3>Superuser: root</h3>
Most operating systems have a special user accout called the superuser
which has <em>complete, unrestricted</em> access to the entire system
regardless of permissions. Windows systems have a user called
<code>Administrator</code> which is usually disabled by default, with
UAC providing elevated priveleges. The Linux superuser is called
<code>root</code> which may or may not be enabled depending on the
distribution of Linux. Distributions with <code>root</code> disabled
have a program called <code>sudo</code> which can elevate priveleges.
On the lab Linux machines <code>root</code> is enabled and can be
logged in to using the terminal.
<p/>
<strong>Note:</strong> Make sure that you understand Linux permissions.
On most Linux systems (including the lab machines) a large majority of
the operating system is owned by <code>root</code>, and since you log
into the machines as a user named <code>user</code>, the operating
system will not allow you to modify those files. Keep this in mind as
in later exercises you will have to log in as <code>root</code> and
modify some system files.
<p/>
  <h2>Useful Linux Commands</h2>
```

```
Here are some basic linux commands that are very useful. If you want
to know more about the command you can run <code>man &lt;command&gt;
</code> in a terminal to view the command's manual.
<dl>
  <dt>cd &lt;directory&gt;</dt>
  <dd>Change the current directory. Typing <code>cd ../</code> will
  change to the parent of the current directory. Please note that
  Linux file and path names are <em>case-sensitive</em></dd>

  <dt>ls</dt>
  <dd>Lists all files and folders in the current directory</dd>

  <dt>pwd</dt>
  <dd>Show the current directory path</dd>

  <dt>man &lt;command&gt;</dt>
  <dd>Shows the manual page for the given command. Very useful if you
  want to know how to use a particular command</dd>

  <dt>cp &lt;source&gt; &lt;destination&gt;</dt>
  <dd>Copy a file/directory &lt;source&gt; to &lt;destination&gt;.
  (Note. you must use the recursive option <code>-r</code> when copying
  folders)</dd>

  <dt>mkdir &lt;directory&gt;</dt>
  <dd>Create a new directory</dd>

  <dt>rm &lt;file1&gt; &lt;file2&gt;...</dt>
  <dd>Remove files. Use the <code>-r</code> option to remove folders.
  Do NOT run this on the root directory <code>/</code> as it could
  potentially wipe the operating system and all mounted drives</dd>

  <dt>cat &lt;file1&gt; &lt;file2&gt;...</dt>
  <dd>Concatinate files and display the result. Useful for just
  displaying the contents of a particular file</dd>
  <dt>cat &lt;file1&gt; &lt;file2&gt;...</dt>
  <dd>Concatinate files and display the result. Useful for just
  displaying the contents of a particular file</dd>

  <dt>locate &lt;file&gt;</dt>
  <dd>Show the location of a file</dd>
  <dt>grep &lt;string&gt; &lt;file&gt;</dt>
  <dd>Find a string within a file</dd>
  <dt>ps</dt>
  <dd>List current running processes. Running <code>ps</code> by
  itself only shows processes within the current terminal. To show all
  running process run <code>ps aux</code></dd>

  <dt>top</dt>
  <dd>Show the top running processes ordered by cpu usage (default)</dd>

  <dt>touch &lt;file&gt;</dt>
  <dd>Updates the timestamp of a file. if the file doesn't exist it is
  created</dd>
  <dt>df</dt>
  <dd>Shows disk usage</dd>

  <dt>free</dt>
  <dd>Shows RAM usage</dd>

  <dt>less</dt>
  <dd>shows the contents of a file allowing you to scroll through the
file</dd>
  <dt>vi</dt>
  <dd>A command line text editor</dd>
  <dt>whoami</dt>
  <dd>Show the current user</dd>
</dl>
GUI programs can also be run from the terminal. One of the most useful
```

```
programs is <code>gedit</code> which is the text editor for gnome (the
desktop GUI on the lab’s Linux machines). This is handy for when we
need to edit system configuration files. To
edit a system file, open a terminal, log in to root using <code>su
</code>, navigate to the directory containing the file you want to
edit using <code>cd &lt;directory&gt;</code>, and open the file using
<code>gedit &lt;filename&gt;</code>.
<p/>
Linux makes extensive use of piping, which allows the output of one
program to be used as input for another program. For example <code>
ps -aux | grep openssh</code> will get the list of currently running
processes, which will be input to <code>grep</code> which will search
for the string <code>openssh</code>. Program output can also be piped
to files, e.g <code>ps -aux > processlist.txt</code> will get the list
of currently running processes and write it to <code>processlist.txt</code>
<p/>
Practice the above commands as some of them will be useful in later
exercises.
</body>
</html>
FUNCTIONS
```

E.10 exercise5.m4

```
include(‘defines.m4’)dnl
XML_HEADER
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <title>Exercise 1 - Static IP Addresses</title>
  STYLE
</head>
<body>
  <h1>Exercise 1 - Static IP Addresses</h1>
<a href="./exercises.php">Exercise List</a>
```

```
  <h2>Required Lab Machines</h2>
<?PHP
displayState("1234");
checkState("1234");
?>
  <h2>Introduction</h2>
  <br/>
In this Experiment you will create a small network of consisting of four
computers. You will be assigning each computer an IP address manually and
then you will test the network by making sure that each computer can
connect to another.

  <h2>Methodology</h2>
1 - To configure a Windows machine with a static IP first click on the
network icon in the system tray then select ’Open Network and Sharing
Centre’ then ’Change Adapter Settings’. Alternatively type ’View Network
Connections’ in the search area of the start menu. Right-click the
network interface you want to configure then select properties. Select
’Internet Protocol Version 4 (TCP/IPv4)’ then click properties.
<p/>
<div><img src=exercise1-1.png alt="Configuring a network address under Windows"/>
<br/>Configuring a network address under Windows</div>
<p/>
2 - To Configure a Linux machine right-click the network icon in the
system tray then select ’Edit Connections’. Select eth0 from the list
then select ’Edit’. Select the IPv4 Tab and change Method to Manual
<p/>
<div><img src=exercise1-2.png alt="Configuring a network address under Linux"/>
<br/>Configuring a network address under Linux</div>
<p/>
3 - Configure the <em>first</em> ethernet interface of the machines to
use the following IP addresses (leave the default gateway and dns server
blank for the moment). When applying these settings to a Linux machine
you may be asked for a password. The password is <code>password</code>.
<br/>
```

```
Linux 1 - Address: 192.168.0.101, Netmask: 225.225.225.0<br/>
Linux 2 - Address: 192.168.0.102, Netmask: 225.225.225.0<br/>
Windows 1 - Address: 192.168.0.103, Netmask: 225.225.225.0<br/>
Windows 2 - Address: 192.168.0.103, Netmask: 225.225.225.0<br/>
<p/>
4 - Click OK/Apply to apply the new settings
<p/>
5 - Confirm that each computers have the correct IP addresses. On Windows
right-click the network interface then select ’status’, then click
’details’. On Linux right-click the network icon in the system tray then
select ’Connection Information’ then select the ’eth0’ tab.
<p/>
<div><img src=exercise1-3.png alt="Checking IP Settings"/>
<br/>Checking IP Settings</div>
<p/>
6 - Confirm that the computers can communicate with each other by running
<code>ping &lt;ip-address&gt;</code>, where <code>&lt;ip-address&gt;
</code> is the address of the computer you want to send the ICMP Packet
to. Sending ICMP Packets with the <code>ping</code> is a great way to
diagnose whether a computer on the network is running or not.

<!--<h2>Results</h2>
<a href="check.php?exercise=1">check</a> whether you have successful completed this exerc

</body>
</html>
FUNCTIONS
```

E.11 exercise6.m4

```
include(‘defines.m4’)dnl
XML_HEADER
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
```

```
  <title>Exercise 6 - DHCP: Dynamic Host Configuration Protocol</title>
  STYLE
</head>
<body>
  <h1>Exercise 6 - DHCP: Dynamic Host Configuration Protocol</h1>
<a href="./exercises.php">Exercise List</a>
  <h2>Required Lab Machines</h2>
<?PHP
displayState("123");
checkState("123");
?>
  <h2>Introduction</h2>
  <br/>
A DHCP server allows the dynamic allocation of IP addresses to clients
without an IP address without having to configure those computers
manually.
<p/>
When a new computer is connected to the network it broadcasts
(sends to address 255.255.255.255) a DHCPDISCOVER message onto the
network which is picked up by any DHCP servers on the network. the DHCP
server(s) then reserves an IP address and broadcasts a DHCPOFFER message
containing the reserved ip address and the hardware address of the new
computer.
<p/>
The new computer recieves the offer(s), accepts
<strong>one</strong> offer and broadcasts a DHCPREQUEST message
requesting the IP address of that particular offer. This also tells
all DHCP servers on the network which offer the new computer accepted
(the other DHCP servers withdraw their offers). The DHCP server whose
offer was offer was excepted then replies with a DHCPACK message
acknowledging the request and containing the lease time and any other
information the new computer needs. The new computer recieves this
and configures its network interface accordingly.
<p/>
In this Experiment you will create a DHCP server on one of the
```

```
machines. This Machine you will give a static ip
address. You will then configure the dhcp server to give a range of
ip addresses to dhcp enabled computers. You will then enable dhcp on the
other computers. You will have successfully completed this exercise
when the other 3 computers are automatically assigned ip addresses.


  <h2>Methodology</h2>
<strong>Note:</strong> You will have to log in to the root user <code>
su</code> in order to modify system files. The root Password is <code>
password</code>. It is recommended you use <code>gedit</code> text
editor to edit system files.
<p/>
1 - Connect to Linux machine 1 as it will run our DHCP server
<p/>
2 - Examine <code>man dhcpd</code> for configuration details.
<p/>
3 - Edit the configuration file <code>/etc/dhcpd/dhcpd.conf</code> and
add the following:
<p/>
<code>
default-lease-time 60;<br/>
max-lease-time 60;<br/>
ddns-update-style interim;<br/>
option subnet-mask 255.255.255.0<br/><br/>
subnet 192.168.1.0 netmask 255.255.255.0<br/>
{<br/>
&nbsp;&nbsp;&nbsp;&nbsp;interface eth0;<br>
&nbsp;&nbsp;&nbsp;&nbsp;range 192.168.1.150 192.168.1.200;<br/>
}<br/>
</code>
<h4>Explanation of parameters</h4>
<dl>
  <dt>default lease time</dt>
  <dd>Default time than an allocated ip is valid in seconds</dd>
  <dt>max-lease-time</dt>
  <dd>Maximum time than an allocated ip is valid in seconds</dd>
  <dt>ddns-update-style = update</dt>
  <dd>Update the DNS server whenever a lease is updated (not used
  here)</dd>
  <dt>option subnet-mask</dt>
  <dd>sets the subnet mask</dd>
  <dt>interface</dt>
  <dd>The network interface for the DHCP server to operate on.
  In this case the server is restricted to eth0.</dd>
  <dt>subnet, netmask, range</dt>
  <dd>This defines the network the dhcp operates on, and the range
  of ip addresses that are handed out to clients</dd>
</dl>
<p/>
<strong>Note:</strong> Parameters outside the subnet definition are
considered global andapply to all subnet definitions while parameters
inside the subnet definition are local to that definition.
<p/>
<strong>Note:</strong> If you want it is possible to allocate a fixed
ip address to a certain hardware address. e.g:
<p/>
<code>
host Q6-4101<br/>
{<br/>
&nbsp;&nbsp;&nbsp;&nbsp;hardware ethernet 76:75:89:32:67:3C;<br/>
&nbsp;&nbsp;&nbsp;&nbsp;fixed address 192.168.11.300;<br/>
}<br/>
</code>
<p/>
This will make the dhcp server assign an IP address of 192.168.11.300
to the machine with the hardware address of 76:75:89:32:67:3C.
<p/>
4 - Configure the first interface (eth0) so Linux-1 to have a static
ip of 192.168.1.1. This will be the network interface that the DHCP
server will be listening on (as shown in the configuration above).
```

```
<p/>
5 - start the dhcp server by running <code>service dhcpd start</code>.
If <code>[FAILED]</code> is shown double check you configuration and
make sure the machine has a static ip. You can also check
<code>/var/log/messages</code> for any dhcpd errors. You can do this
by running <code>tail -n 20 /var/log/messages</code> (must be run as
root).
<p/>
6 - Windows-1 and Linux-2 will serve as our DHCP clients so you must
configure them to use DHCP. You can do this in the
same place where you assigned static ips, but this time you select
'obtain an ip address automatically' in Windows and change Method to
'Automatic (DHCP)' in Linux.
<p/>
7 - Ensure that the client machines are all assigned ip addresses.
these address should be 192.168.1.150, 192.168.1.151 and 192.168.1.152,
or something similar. Windows strangely tends to assign a (usually)
useless ip address to an interface that cannot connect to a network.
If this is the case you may have to disable the interface and renable
it (right-click).
<p/>
You will have complete this exercise when all interfaces on all machines
(except the interface the DHCP is listening on) are assigned an IP address
by the DHCP server.

</body>
</html>
FUNCTIONS
```

E.12 exercise7.m4

```
include(`defines.m4')dnl
XML_HEADER
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
```

```
<head>
  <title>Exercise 7 - Windows File Sharing: NetBIOS Protocol</title>
STYLE
</head>
<body>
  <h1>Exercise 7 - Windows File Sharing: NetBIOS Protocol</h1>
<a href="./exercises.php">Exercise List</a>
  <h2>Required Lab Machines</h2>
<?PHP
displayState("D34");
checkState("D34");
?>
  <h2>Introduction</h2>
  <br/>
In this exercise you will share a folder on one Windows machine, and
access it on another Windows machine. You will have successfully
completed this exercise when you are able to share files from one
Windows machine to another.
  <h2>Methodology</h2>
1 - Click start and then right-click computer then select properties.
Ensure that the workgroup is named WORKGROUP.
<p/>
2 - Ensure that the machine has file sharing turned on. To do this open
the Network and Sharing Centre and select 'Change advanced sharing
settings'. Make sure that 'Turn on file and printer sharing', 'Turn on
network discovery', 'Turn off password protected sharing' are selected
for public networks. Do this for both Windows-1 and Windows-2.
<p/>
<div><img src=exercise7-1.png alt="Enable Sharing"/>
<br/>Enable Sharing</div>
<p/>
3 - Create a new folder on the desktop. To share this folder right-
click it and select properties. Select the sharing tab and click
'share'. You can choose which users you can share the folder with. For
this exercise add 'Everyone' from the drop-down box and click add then
```

```
change the permission level for 'Everyone to 'Read/Write'. Click Share
then Done.
<p/>
<div><img src=exercise7-2.png alt="Share Folder"/><br/>Share Folder</div>
<p/>
Also Click 'Advanced Sharing' and make sure 'Share this
folder' is checked. Click permissions and make sure that 'Everyone'
has full control.
<p/>
<div><img src=exercise7-3.png alt="Share Folder"/><br/>Share Folder</div>
<p/>
If you're asked to turn of file and folder sharing select yes.
<p/>
4 - On the other Windows machine open Windows explorer and select
'Network' from the side menu. From here you should be able to navigate
to the shared folder. If you cannot see the folder try typing
<code>&#92;&#92&lt;ip-address&gt;&#92;</code> into the address bar where
<code>&lt;ip-address&gt;</code> is the ip address of the machine with the
shared folder (you will need to find the IP address yourself).Test that
you can read and write to the folder
by creating a new file. If you cannot read or write to the folder check
the sharing permissions.
<p/>
<div><img src=exercise7-4.png alt="Connecting to shared folder"/><br/>
Connecting to shared folder</div>
<p/>
5 - Right-click the share and select 'Map Network Drive', and assign
the share a drive letter. You should now be able to open the shared
folder from My Computer. Right-click the network drive and select
disconnect to remove the drive.
<p/>
<h4>The following does the same as the above steps except using the command
line.</h4>
6 - Run <code>net view</code> to list windows computers on the network
<p/>
```

```
7 - Run <code>net use /? | more</code> to learn the syntax on how to
assign a drive letter to a share
<p/>
8 - Assign a drive lette to the share by running <code>net use
&lt;drive-letter&gt;: \\&lt;computer-name&gt;\&lt;share-name&gt;</code>,
where <code>&lt;drive-letter&gt;</code> is the drive letter you want
to assign, <code>&lt;computer-name&gt;</code> is the name or IP address
of the machine containing the shared folder, and
<code>&lt;share-name&gt;</code> is the name of the share.
The shared folder should now apper in My Computer as a network drive.
<p/>
9 - List current shares by running <code>net use</code>
<p/>
10 - Disconnect the share by running <code>net use
&lt;drive-letter&gt;/delete</code>
</body></html>
FUNCTIONS
```

E.13 exercise8.m4

```
include(‘defines.m4’)dnl
XML_HEADER
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <title>Exercise 8 - Unix-Windows File Sharing - Samba</title>
STYLE
</head>
<body>
  <h1>Exercise 8 - Unix-Windows File Sharing - Samba</h1>
<a href="./exercises.php">Exercise List</a>
  <h2>Required Lab Machines</h2>
<?PHP
displayState("D13");
checkState("D13");
```

```
?>
  <h2>Introduction</h2>
Samba allows Unix files to be accessable on Windows using the SMB
(Server Message Block) protocol, also known as CIFS (Common
Internet FileSystem). In this exercise you will share a folder on a
Linux machine, and access it on a windows machine. You will have
successfully completed this exercise when you are able to share files
from the Linux machine to a Windows machine.

  <h2>Methodology</h2>
<strong>Note:</strong> You will have to log in to the root user <code>
su</code> in order to modify system files. The root Password is <code>
password</code>. It is recommended you use <code>gedit</code> text
editor to edit system files.
<p/>
1 - Connect to Linux-1, which will run our samba server.
<p/>
2 - Examine <code>man smb.conf</code> for configuration details.
<p/>
3 - Rename <code>/etc/samba/smb.conf</code> to
<code>/etc/samba/smb.conf.backup</code> using <code>mv /etc/samba/smb.conf
/etc/samba/smb.conf.backup</code>and create a new blank smb.conf
by running <code>touch /etc/samba/smb.conf</code>
<p/>
4 - Edit <code>/etc/samba/smb.conf</code> using gedit and add
the follwing:<p/>
<code>
[global]<br/>
&nbsp;&nbsp;&nbsp;&nbsp;workgroup = WORKGROUP<br/>
&nbsp;&nbsp;&nbsp;&nbsp;Server String = I am a Samba Server<br/>
&nbsp;&nbsp;&nbsp;&nbsp;security = user<br/>
&nbsp;&nbsp;&nbsp;&nbsp;Map to guest = Bad User<br/>
&nbsp;&nbsp;&nbsp;&nbsp;username map = /etc/samba/smbusers<br/>
</code>
<p/>
```

```
<h4>Explanation of parameters</h4>
<dl>
  <dt>workgroup</dt>
  <dd>Windows NT Domain Name/Workgroup Name</dd>
  <dt>Server String</dt>
  <dd>server description</dd>
  <dt>security = user</dt>
  <dd>security mode samba runs in, setting it to user mode makes samba
  authenticate incoming connections</dd>
  <dt>Map to guest = Bad User</dt>
  <dd>makes samba treat any unauthenticated users as a guest user</dd>
  <dt>username map</dt>
  <dd>The file listing the users that samba uses for authentication</dd>
</dl>
<p/>

5 - make sure that <code>nobody = guest</code> exists in
<code>/etc/samba/smbusers</code>. If <code>smbusers</code> doesnt exist
then create it and add the line to it. <code>nobody = guest</code> will
link the samba guest account to the Linux account 'nobody'.
<p/>
6 - Create folder <code>/home/public</code>. This will be the folder
you will be sharing. Make sure that it has its permissions for other set
to read, write and executable by running <code>chmod o+rwx /home/public</code>
(<strong>important!</strong>)
<p/>
7 - Add the following share definition to <code>/etc/samba/smb.conf
</code>:
<p/>
<code>
[share]<br/>
&nbsp;&nbsp;&nbsp;&nbsp;comment = Public Folder<br/>
&nbsp;&nbsp;&nbsp;&nbsp;path = /home/public<br/>
&nbsp;&nbsp;&nbsp;&nbsp;writable = yes<br/>
&nbsp;&nbsp;&nbsp;&nbsp;browsable = yes<br/>
```

```
&nbsp;&nbsp;&nbsp;&nbsp;public = yes<br/>
&nbsp;&nbsp;&nbsp;&nbsp;only guest = yes<br/>
</code>
<p/>
<h4>Explanation of parameters</h4>
<dl>
  <dt>comment</dt>
  <dd>description of the share definition</dd>
  <dt>path</dt>
  <dd>path to the folder to be shared</dd>
  <dt>writable</dt>
  <dd>whether or not connected users can write to the shared folder</dd>
  <dt>browsable</dt>
  <dd>whether or not connected users can navigate the shared folder</dd>
  <dt>public</dt>
  <dd>whether the guest account is allowed acces (guest ok = yes/no can
  be used here instead)</dd>
  <dt>only guest</dt>
  <dd>whether or not only the guest account is allowed access</dd>
</dl>
<p/>
8 - Start the server by running <code>service smb start</code>. You
can stop the server by running <code>service smb stop</code> and check
its status by running <code>service smb status</code>
<p/>
9 - Check that the server works. On a windows machine open windows
explorer and enter <code>\\&lt;ip-address&gt;\</code> into the address
bar, where <code>&lt;ip-address&gt;</code> is the ip address of the
machine that the server is running on. To find the ip address type
<code>ifconfig</code> on the maching running the server.
<p/>
10 - If any "cannot connect" or "permission denied" errors occur, check
the status of the server in <code>/var/log/samba/log.smbd</code>. Also
double check <code>/etc/samba/smb.conf</code> for any errors. Also check
the permissions of the shared folder on the server. the folder and all
```

```
file inside should have read, write and execute permissions. If not
then change the permissions using the chmod command.
</body></html>
FUNCTIONS
```

E.14 exercise9.m4

```
include(‘defines.m4’)dnl
XML_HEADER
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <title>Exercise 9 - Unix File Sharing - NFS: Network Filesystem
  </title>
STYLE
</head>
<body>
  <h1>Exercise 9 - Unix File Sharing - NFS: Network Filesystem</h1>
<a href="./exercises.php">Exercise List</a>
  <h2>Required Lab Machines</h2>
<?PHP
displayState("D12");
checkState("D12");
?>
  <h2>Introduction</h2>
  <br/>
In this exercise you will share a folder on a Linux machine, and access
it on another Linux machine. You will have successfully completed this
exercise when you are able to share files from one Linux machine to
another.
  <h2>Methodology</h2>
<strong>Note:</strong> You will have to log in to the root user <code>
su</code> in order to modify system files. The root Password is <code>
password</code>. It is recommended you use <code>gedit</code> text
editor to edit system files.
```

```
<p/>
1 - Connect to Linux machine 1
<p/>
2 - create folder <code>/home/public</code> by running <code>mkdir -p
/home/public</code>. This will be the folder you will be sharing.
<p/>
3 - start the NFS server by running <code>service nfs start</code>
<p/>
4 - In order to make a folder accessible to a client it must be
"exported" by the server. To do this we must make an entry into <code>
/etc/exports</code>. For the full set of options check the exports man
page <code>man exports</code>.
<p/>
5 - Edit <code>/etc/exports</code> and add this line:
<p/>
<code>
/home/public 192.168.11.*(rw,sync,no_root_squash,no_subtree_check)
</code>
<p/>
<h4>Explanation of parameters</h4>
<dl>
  <dt>/home/public</dt>
  <dd>folder to be shared</dd>
  <dt>192.168.11.*</dt>
  <dd>specifies ip addresses that are allowed to connect (in this case
  the entire 192.168.11.xxx subnet)</dd>
  <dt>rw</dt>
  <dd>allow both read and write requests</dd>
  <dt>sync</dt>
  <dd>reply to new requests only after previous changes have been completed
  on the server system</dd>
  <dt>no_root_squash</dt>
  <dd>allows the client's root account to be mapped to server's root
  account (by default requests made by the client's root account are
  mapped to <code>nobody</code> on the server, an action called root
  squashing)</dd>
  <dt>no_subtree_check</dt>
  <dd>disable checking whether the file/folder is a descendant of the
  exported directory</dd>
</dl>
<p/>
6 - <code>exportfs</code> is the program that maintains the export
table for the nfs server. Now we have to update the export table with
the new entry we put into <code>/etc/exports</code>. Run <code>exportfs
-a</code> to do this.
<p/>
7 - On the other Linux machine create a mount point. The concept of
mount points is that a partition on a physical drive (or in our case a
location on a remote computer) us bound to a folder on the filesystem.
For example if you mount sda1 (first partition of physical drive sda)
to folder <code>/media/drive/</code> the contents of that partition is
accesible from that folder. Windows uses the same concept except it
uses specialised mount points (drive letters).
<p/>
So lets create a mount point <code>/mnt/remotedir/</code>, (run
<code>mkdir -p /mnt/remotedir</code>), and then mount the shared folder
on the remote system to the mount point that we just created. Run
<code>mount &lt;ip-address&gt;:/home/public /mnt/remotedir</code>.
<p/>
8 - You should now be able to access the shared folder by accessing
 <code>/mnt/remote/dir</code>. Take note that Linux permissions play a
big role here. By default the permissions of <code>/mnt/remotedir</code>
only allow write access for root. To allow a normal user to write to
files in the shared folder you need to change the permissions of the mount
point (<code>/mnt/remotedir</code>). You can give everyone write permisions
by running <code>chmod o+rwx /mnt/remotedir</code>.
</body></html>
FUNCTIONS
```

E.15 exercise10.m4

```
include(‘defines.m4’)dnl
XML_HEADER
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <title>Exercise 1 - Static IP Addresses</title>
  STYLE
</head>
<body>
  <h1>Exercise 1 - Static IP Addresses</h1>
<a href="./exercises.php">Exercise List</a>
  <h2>Required Lab Machines</h2>
<?PHP
displayState("1234");
checkState("1234");
?>
  <h2>Introduction</h2>
  <br/>
In this Experiment you will create a small network of consisting of four
computers. You will be assigning each computer an IP address manually and
then you will test the network by making sure that each computer can
connect to another.

  <h2>Methodology</h2>
1 - To configure a Windows machine with a static IP first click on the
network icon in the system tray then select 'Open Network and Sharing
Centre' then 'Change Adapter Settings'. Alternatively type 'View Network
Connections' in the search area of the start menu. Right-click the
network interface you want to configure then select properties. Select
'Internet Protocol Version 4 (TCP/IPv4)' then click properties.
<p/>
<div><img src=exercise1-1.png alt="Configuring a network address under Windows"/>
<br/>Configuring a network address under Windows</div>
```

```
<p/>
2 - To Configure a Linux machine right-click the network icon in the
system tray then select 'Edit Connections'. Select eth0 from the list
then select 'Edit'. Select the IPv4 Tab and change Method to Manual
<p/>
<div><img src=exercise1-2.png alt="Configuring a network address under Linux"/>
<br/>Configuring a network address under Linux</div>
<p/>
3 - Configure the <em>first</em> ethernet interface of the machines to
use the following IP addresses (leave the default gateway and dns server
blank for the moment). When applying these settings to a Linux machine
you may be asked for a password. The password is <code>password</code>.
<br/>
Linux 1 - Address: 192.168.0.101, Netmask: 225.225.225.0<br/>
Linux 2 - Address: 192.168.0.102, Netmask: 225.225.225.0<br/>
Windows 1 - Address: 192.168.0.103, Netmask: 225.225.225.0<br/>
Windows 2 - Address: 192.168.0.103, Netmask: 225.225.225.0<br/>
<p/>
4 - Click OK/Apply to apply the new settings
<p/>
5 - Confirm that each computers have the correct IP addresses. On Windows
right-click the network interface then select 'status', then click
'details'. On Linux right-click the network icon in the system tray then
select 'Connection Information' then select the 'eth0' tab.
<p/>
<div><img src=exercise1-3.png alt="Checking IP Settings"/>
<br/>Checking IP Settings</div>
<p/>
6 - Confirm that the computers can communicate with each other by running
<code>ping &lt;ip-address&gt;</code>, where <code>&lt;ip-address&gt;
</code> is the address of the computer you want to send the ICMP Packet
to. Sending ICMP Packets with the <code>ping</code> is a great way to
diagnose whether a computer on the network is running or not.

<!--<h2>Results</h2>
```

```
<a href="check.php?exercise=1">check</a> whether you have successful completed this exerc


</body>

</html>

FUNCTIONS
```