

Creation and Evaluation of an Educational Framework for use in Network Teaching

David I. L. McLuskie

Submitted in partial fulfilment of the requirements of
Napier University
for the degree of
Master of Science in Advanced Networking

School of Computing
October 2008

Authorship declaration

I, David McLuskie, confirm that this dissertation and the work presented in it are my own achievement.

1. Where I have consulted the published work of others this is always clearly attributed;
2. Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;
3. I have acknowledged all main sources of help;
4. If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;
5. I have read and understand the penalties associated with Academic Misconduct.
6. I also confirm that I have obtained **informed consent** from all people I have involved in the work in this dissertation following the School's ethical guidelines

Signed:

Date: 26/09/08

Matriculation no: 07008868

Data Protection declaration

Under the 1998 Data Protection Act we cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please sign your name against *one* of the options below to state your preference.

The University may make this dissertation, with indicative grade, available to others.

The University may make this dissertation available to others, but the grade may not be disclosed.

David I.L. McLuskie, MSc Advanced Networking 2008

The University may not make this dissertation available to others.

Abstract

Teaching and assessing students on the practical side of networking can be achieved through the use of simulators. However network simulators are limited in what they can do, since the device being simulated is not fully functional and the generation of the exercises typically results in the same specification being presented to the student [1, 2]. When the student has finished an exercise they are typically just presented with an outline grade with little indication of areas of weakness or strength.

This thesis investigates how the Bloom [3] and SOLO [4] learning taxonomies can be used to specify and grade network challenges, while using the idea of *fading worked* examples [5] to design the challenges to lower the cognitive load on the student. It also proposes a framework that can be used to generate network challenges specifications that changes every time the student attempts then. The challenge can then be solved using an emulation package (Dynamips) while a bolt-on package (GNS3) is used to provide the graphical user interface. Once the student has finished a challenge, it will then be graded and feedback presented indicating what was correct and incorrect.

The thesis includes a novel assessment method in grading the academic level of these challenges, based on key academic cognitive levels, such as with Knowledge, Comprehension, Application, Analysis, Synthesis, and Evaluation. There are three example challenges, which are assessed using Bloom's taxonomy. The challenges include: basic router configuration; EIGRP configuration between two routers; and a redistribution network with a three routers system running EIGRP and OSPF. The thesis outlines the coverage of most of the Bloom layer, apart from the synthesis and evaluation levels, which are covered in the conclusions.

From the results of a questionnaire, two of the most positive aspects of using the framework was that a fully feature IOS command line interface was available for the students to use, and also once they had mastered a skill they did not have to start from scratch in subsequent exercises, thus not reusing skills that had already mastered. However one of the negative aspects noticed from the questionnaire was the number of complex steps that was required to be followed to setup the challenge.

The thesis shows the novel results that MAC OS with a Windows XP virtual image produces an average 51% CPU footprint when running the most advanced challenge network topology using the BGP routing protocol, whereas while using the Windows XP and Vista the processor impact is only 1.5-2.5%. It also shows that there is little impact on the disk and memory components. Further investigation was carried out to determine how the Windows XP virtual image would cope with additional routers. It was proven that the processor utilisation was nearly linear, approximately 7% increase per router, going from 2 to 3 to 4, but adding 5 and 6 routers showed that the increase was no longer linear as might have been expected.

Table of Contents

Acknowledgements	10
1 Introduction	11
1.1 Context	11
1.2 Aims and objectives.....	12
1.3 Background	12
1.4 Thesis structure.....	13
2 Literature Review.....	14
2.1 Introduction	14
2.2 Simulation.....	14
2.2.1 Simulators.....	14
2.2.2 Simulation Models.....	15
2.2.3 Network Evaluation.....	15
2.2.4 Emulab.....	16
2.2.5 Planetlab	17
2.2.6 Labs	17
2.3 Virtualisation.....	19
2.3.1 Goals of Virtualisation	19
2.3.2 Types of Virtualisation.....	19
2.3.3 Costs.....	20
2.3.4 Optimisation	20
2.3.5 Uses	20
2.3.6 Benefits.....	21
2.3.7 Education	22
2.3.8 Setup	22
2.3.9 Future Trends	22
2.4 Education	23
2.4.1 Cognitive Architecture	23
2.4.2 Learning Taxonomy.....	24
3 Design.....	29
3.1 Introduction.....	29
3.2 Challenge Generator Program Design	29
3.2.1 Compose Challenge.....	30
3.2.2 Select Challenge	30
3.2.3 Generate Challenge Data	30
3.2.4 Generate/Query Help Bank.....	30
3.2.5 Display Challenge Specification.....	30
3.2.6 Run Emulator	30
3.2.7 Mark Challenge.....	31
3.2.8 Provide Feedback.....	31
3.3 Data Flows.....	31
3.4 Challenges.....	32
3.4.1 Cisco Labs	32
3.4.2 Challenge Generation.....	33
3.5 Questionnaire.....	39
3.6 Footprint Evaluation.....	39
3.7 Conclusions	41
4 Implementation	42
4.1 Introduction	42
4.2 Framework Implementation.....	42

4.2.1	Router Emulation	42
4.2.2	Compose Challenge.....	47
4.2.3	Generate Challenge Data	53
4.2.4	Display Challenge Specification.....	54
4.2.5	Mark Challenge / Provide Feedback.....	54
4.3	Footprint Evaluation Implementation	59
4.4	Questionnaire Implementation	61
4.5	Conclusions	61
5	Evaluation	62
5.1	Introduction	62
5.2	Performance Evaluation	62
5.3	Questionnaire Evaluation	65
5.4	Conclusions	66
6	Conclusions.....	67
6.1	Introduction	67
6.2	Conclusions	67
6.2.1	Performance Issues	67
6.2.2	Bloom Model.....	67
6.3	Aim and Objectives	68
6.3.1	Objective 1.....	68
6.3.2	Objective 2.....	69
6.3.3	Objective 3.....	72
6.4	Future Work.....	72
6.4.1	Evaluation.....	72
6.4.2	Challenge Creation.....	72
6.4.3	Bloom Challenge Level.....	73
6.4.4	Challenge Grading	73
6.4.5	Challenge Implementation	74
Appendix A - Challenge 2 Startup-Configs.....		76
Appendix A.1 - Router R0		76
Appendix A.2 Router R1.....		78
Appendix B - Challenge 3 Startup-Configs.....		80
Appendix B.1 – Paris Router		80
Appendix B.2 – London Router		82
Appendix B.3 – Edinburgh Router		84
Appendix C - Sample Dynagen File.....		86
Appendix D – Challenge Generator Code Listing.....		88
Appendix E – Performance Monitor Control Program.....		115
Appendix F – Project Plan		123
Appendix G – Example Project Diary.....		124

List of Figures

Figure 3-1 Program Design Block Diagram.....	29
Figure 3-2 Create Challenge Data Flow	31
Figure 3-3 Data Flow Diagram	32
Figure 3-4 Challenge 1 Network Topology.....	34
Figure 3-5 Challenge 2 Network Topology.....	35
Figure 3-6 Challenge 2 Diagnostic Screenshot	36
Figure 3-7 Challenge 3 Network Topology.....	37
Figure 4-1 GNS3 Screenshot.....	43
Figure 4-2 Dynamips in Task Manger	44
Figure 4-3 GNS3 Network Topology.....	45
Figure 4-4 Dynamips Hypervisor Killed	45
Figure 4-5 Edinburgh Router Loosing Connection.....	46
Figure 4-6 Paris Router Ping Response	46
Figure 4-7 London Router Ping Response	47
Figure 4-8 Challenge 2 Network Topology.....	48
Figure 4-9 Exporting Challenge2 Configs.....	49
Figure 4-10 Challenge 2 startup-config files.....	49
Figure 4-11 Screenshot of Experiment Running	61
Figure 5-1 Chart of Average Processor Utilisation.....	64

List of Tables

Table 2-1 Bloom Levels.....	25
Table 2-2 Revised Bloom	26
Table 2-3 SOLO Levels.....	27
Table 3-1 Bloom rating for Challenge 1	35
Table 3-2 Challenge 2 Bloom Rating	37
Table 3-3 Challenge 3 Bloom Rating	38
Table 3-4 Windows Performance Counters.....	40
Table 4-1 Native Machine Specification.....	59
Table 4-2 Emulation Machine Specification	59
Table 4-3 Software Versions	60
Table 4-4 Performance Counters	60
Table 5-1 Average Disk Transfers/second.....	62
Table 5-2 Average Memory\Pages /second	62
Table 5-3 Average Processor Utilisation.....	62
Table 5-4 Optimal Values for Performance Counters	63
Table 5-5 Average Processor Utilisation.....	63
Table 5-6 Questionnaire Responses	65

List of Code Snippets

Code Snippet 4-1 Router Class Definition.....	50
Code Snippet 4-2 First Router Class Constructor.....	50
Code Snippet 4-3 Class A IP Address.....	51
Code Snippet 4-4 Router Class Second Constructor	52
Code Snippet 4-5 Router Class Query Members	52
Code Snippet 4-6 Loopback Class Definition.....	52
Code Snippet 4-7 Loopback Constructor Class.....	53
Code Snippet 4-8 Challenge Initial Definition.....	53
Code Snippet 4-9 Process Member Function	54
Code Snippet 4-10 displaySpec Member Function.....	54
Code Snippet 4-11 Full Process Member Function	55
Code Snippet 4-12 checkConfigs Member Function	56
Code Snippet 4-13 Verifying Loopback Interfaces	56
Code Snippet 4-14 Verifying IP Address	57
Code Snippet 4-15 EIGRP Verification.....	57
Code Snippet 4-16 Verifying Connected Interfaces	58
Code Snippet 4-17 Advertising Loopback Interfaces into EIGRP	58

Acknowledgements

My first thanks goes to Professor Bill Buchanan, my supervisor for this work, for all the help and assistance he has provided.

Additional thanks goes to Dr. Ahmed Al-Dubai for acting as internal supervisor for this thesis.

Finally I have to thank all the guinea pigs that volunteered to evaluate the framework

1 Introduction

1.1 Context

One of the main challenges with teaching networking courses is trying to maintain the student's attention during exercises. The reasons that the students do not maintain their attention is either because they have already repeated the exercise before, or they are attempting a new exercise, but they have to repeat skills that they have already mastered before they get to the new skills that they have to master [5].

A possible example of a user becoming bored could be seen when they are carrying out a simulated exercise in the Cisco Academy [6]. Each time an exercise is started the user is forced to configure the basics first, like the IP address of the interfaces, routing, and so on, and this can take some time before they get to the new material. If they are presented with several exercises in a row then having to repeat this process each time can get quite frustrating and boring, and this can have the knock-on effect of the user either skipping or forgetting the point of the exercise, which has the end result of them not achieving the desired learning outcomes.

There are various software programs available for educational purposes that can test a learners understanding of performing network configurations, but each program has weaknesses. One of the main weaknesses is that they often do not have a full command set and some commands are not even implemented, at all[1, 2]. This means that the learner is restricted to the commands that they can execute. Also the network configuration task normally has a fixed topology [1], which is not ideal as it only prepares the student for that configuration, and it does not test the adaptability of the student.

If a user is unable to use real network equipment to practice on, then they typically have two alternative choices to work with, and they are to use either a simulator or an emulator. Even though both options perform the same function they work in different ways.

A simulator pretends to be the device and lets the user interact as if they are sitting in front of the network device (e.g. boson netsims [7] or Networksims [8]). If a command is entered, the simulator pretends that the command is being executed without the actual processing being carried out. An emulator is a program that usually emulates a device in its entirety, and when a command is entered the emulator will process the command as it would if the actual hardware was being used (e.g. using VMware [9] or dynamips [10])

One of the other problems with simulators is that when the student has completed an exercise no detailed feedback is presented to them, only a simple pass or fail is typically returned [11]. There is typically no scoring mechanism based upon how long they took to complete the exercise, or how many times the user had to access the help feature to look up a command.

1.2 Aims and objectives

The overall aim of this project is to develop and evaluate an educational framework that could be used to enhance education learning in universities for network courses. To achieve the overall aim for this project the following objectives had to be met:

1. Investigate the current learning taxonomies that are used in an educational environment and investigate how these can be applied in an emulated network environment to provide an enhanced learning tool for students.
2. Integrate a network emulation module that will automatically generate a dynamic network scenario for the user to configure and then automatically grade the user based upon how they solved the problem.
3. Monitor the performance of the overall solution to verify that the solution developed can be run on the user's machines without any adverse affects that would spoil the user experience.

1.3 Background

There are various vendor certifications that have gained prominence recently including Microsoft (MCP/MCSE), Cisco Systems (CCNA/CCNP/CCIE) and Oracle (OCP) certifications [12-14]. In networking the most highly sought after is the CCIE qualification from Cisco and this is because it seen as the gold standard of qualifications since it is the hardest one to achieve [15].

The Cisco Academy program is a repository of courses, written by Cisco, that are used to teach the material for Cisco certifications and they use a mixture of written, simulation and exercises to present the material to the student. The Cisco Academy program is used by some Universities as part of their networking courses but some of them have trouble integrating the written material.

The quality of Cisco's material, though, can be a hit or miss affair for some courses and can even change chapter by chapter. One of the problems with the written material is that it does not take into consideration how different students process and learn information, also the student can become overwhelmed with the amount of information being presented to them and the result being that they are not learning to the best of their ability.

When studying for a vendor certification one of the major barriers is obtaining the equipment to practice on. With the Microsoft and Oracle certifications, this usually meant obtaining a few computers, and for the Cisco certifications the

candidate requires to obtain the actual Cisco routers and switches. For a candidate this expenditure could end-up being quite costly [16].

Up until recently users had little option but to purchase the required hardware but as the computers have become more powerful an additional option was presented and that is using either simulation or emulation. For the Microsoft and Oracle certifications the candidates could use a program called VMware [9] on their host machine to load a virtual machine with which the required software could be loaded on. For the Cisco certification, simulation is the main tool that is used by the candidates to practice and this is used quite widely in the Cisco Academy program [6].

One of the problems with using the simulators is that the student is often asked to repeat long winded configuration steps, which they have already mastered before they get to the new material. The other problems is that the student is often not using a fully featured IOS command line interface, and this means that they can not use short cut commands or select previous commands using the arrow keys to help save time. Having simulators for practicing the commands is an academic advantage, as it can reinforce some of the concepts that the student is reading. However the downside is that if the student attempts the exercise again, then the same specification is often used and the end result could be that the student is remembering the commands to solve that problem and not engaging in actual learning.

1.4 Thesis structure

Chapter 2 provides a review of the literature in three key areas. The first area deals with that of network simulation and how it is used for research and education. The second area examines virtualisation and how it is used in both education and industry. The third and final area examines how learning takes place, the most popular educational taxonomies in use and how best to teach students.

Chapter 3 details the design of a proposed framework for generating challenges that have variables that change each time the challenge is attempted. A block diagram is presented that shows how the all the process's for the framework is put together and this is then followed by a data flow diagram that shows the flow of data between all the processes. The next part of the design chapter describes how the challenges were designed and is then followed on by describing how the overall solution will be evaluated.

Chapter 4 describes how the implementation was done for the framework and also how the overall solution was evaluated.

Chapter 5 discusses the evaluation of the results and describes any problems with suggestions for improvements.

Chapter 6 is split into three sections. The first section will deal with the main conclusions of the thesis, the second section will examine the aim and objectives and show how they have been met, along with any problems encountered and suggestions for improvements. The third section will discuss future improvements.

2 Literature Review

2.1 Introduction

This chapter presents a review of the existing literature in the areas of simulation, virtualisation and current learning trends. The first section will look into simulation and how it is used in research and teaching, the second section will examine virtualisation and how it is used in education, industry and also examine future trends. The final section will examine how best to present the lecture, so that the students learn to the best of their ability. Also the current learning taxonomies will be investigated and how they can be used to define the learning outcomes of a module and gauge exam answers.

2.2 Simulation

One of the major problems in research and teaching in networking is obtaining access to the required level of hardware. Obtaining access to the hardware can be troublesome because of several reasons, ranging from the university not having the money to purchase the equipment or not having sole access to the equipment i.e. other projects need to use the same equipment [1]. This is where simulation comes into play in that it can allow the user to set up and conduct experiments at a time of their choosing, without having to worry about any time constraints such as having to make way for other researchers.

If a user is unable to use real network equipment to practice on, then they typically have two alternative choices to work with, and they are to use either a simulator or an emulator. Even though both options perform the same function they work in different ways.

A simulator pretends to be the device and lets the user interact as if they are sitting in front of the network device (e.g. BOSON netsims [7] or Networksims [8]). If a command is entered, the simulator pretends that the command is being executed without the actual processing being carried out. An emulator is a program that usually emulates a device in its entirety, and when a command is entered the emulator will process the command as it would if the actual hardware was being used (e.g. using VMware [9] or dynamips [10]).

A simulator is good for practicing the actual commands, while an emulator is good for practicing the commands in a real time environment. One of the major differences between a simulator and emulator is that with the simulator the full range of commands may not be available to the user, while with an emulator the full command range is available.

2.2.1 Simulators

For network research there are various simulators available that can be used to model new protocols and evaluate network performance. Two of the most

popular simulation programs are called ns-2 [1, 17, 18], which will soon be surpassed by ns-3, and Opnet [1, 17, 18].

Both of these programs are modelling packages, where the network protocol is specified using a complex programming language. Both ns-2 and Opnet are ideal for researchers who are prepared and have the time to put the work into learning the modelling language. Students on the other hand may not have the required time to learn everything to make best use of these packages. It also has to be taken into consideration that networking students have little to no experience of programming and this will have the effect of making the learning curve even steeper for them.

One of the big disadvantages of ns-2 and Opnet is that they deal with the theoretical aspect of networks, and they do not introduce to the student how to configure and troubleshoot the networks. There is also the problem that ns-2 and Opnet do not work in real time. The time taken to perform the simulation on both platforms can take considerably longer than the actual "simulated time"[19, 20].

2.2.2 Simulation Models

The amount of time taken to do the simulation depends upon the simulation model that is being used. There are two ways of performing simulation, packet and flow level [19].

Packet level is the standard approach and allows for very highly accurate simulation using the full implementation of the TCP/UDP stack. However there is the problem in that the time taken to perform the simulation exceeds the time that is being simulated when large data sizes are being used. The highly accurate simulation is ideal for researchers studying protocols with small data sizes, however the time taken when large data packets are being used can be prohibitive.

To reduce the execution time of simulation an alternative approach was introduced called flow level simulation. The idea behind flow level simulation is that it does not attempt to model the entire TCP/UDP stack. Instead it uses theoretical models of the TCP flow but the problem with this approach is that the model makes assumptions that do not reflect in the real network, for example the TCP slow-start behaviour is ignored which has the end result of reducing accuracy. Flow level simulation has been proven to reduce simulation time, but at the cost of accuracy and it is up to the researcher to determine if this loss of accuracy is acceptable to their experimentation.

2.2.3 Network Evaluation

There are three ways to perform an evaluation of a network [21]:

1. Analytical Evaluation
2. Simulation
3. Experimentation (aka experimental evaluation / test bed)

In Analytical evaluation the network is modelled using a set of equations to “measure the performance and cost under input parameter variation” [21]. This evaluation only works with simple networks due to the fact that everything has to be expressed as equations and this can get quite complex and time consuming for large networks. As a result Analytical evaluation is not commonly used.

Simulation is the most popular form of evaluation as it allows a large complex network to be modelled quite easily using packages such as ns-2/3 and Opnet. The main problem with using this approach is that it is a “*pure*” form of evaluation. The simulation does not take into account the fact that the software implemented in network devices could contain bugs. These bugs can result in the simulation results not being able to be replicated using a real world network equipment.

The final form of evaluation is experimentation whereby the network is modelled using network hardware. This allows the proper evaluation of the network in a realistic environment with bugs and all. The main problem with this type of evaluation is that there is additional expense in obtaining the equipment and lab space required to run the experimentation, especially if it has to be run for some considerable length of time. Also the researcher will need to have the relevant experience to setup and configure the network.

Given the additional expense it can be seen why researchers prefer to use simulation to conduct experiments since it is a cheaper way of doing things and everything can be contained in a small amount of space. It is recommended though that researchers should not rely just on simulation alone, a combination of at least two of the network evaluation methods mentioned should be used to verify results.

To overcome the problem of the lack of accuracy with the flow level simulation it might be suggested that experimental evaluation should be used in conjunction with simulation. By following this approach it would allow for proper verification of the simulation results, but some researchers may see the duplication of effort as a waste and thus not perform this additional step.

So far the uses of simulators have been discussed along with their advantages and disadvantages. The next section will examine two popular emulators called Emulab and Planetlab.

2.2.4 Emulab

Emulab is a set of computers that are connected together by a switch [17, 18]. The user is able to specify a topology using a GUI interface that will then be configuring automatically on the hardware by the system. The links that connect the computers together can be configured to introduce delay, bandwidth and loss characteristics. On each computer the desired operating system can be specified and loaded as required (e.g. Windows XP, Windows Server or Linux).

Emulab can be adapted for the use that is required of it, for example an Emulab lab can be designed for operating system experimentation using blade servers to reduce the amount of space required to house the lab. Workstation or servers could be used but since space in Universities can be at a premium it is recommended to use blade servers where possible.

Alternatively for network experimentation Emulab can again be used mixed with Cisco routers [18] and blade servers. One of the main problems with this setup is finding space to store the Cisco routers, because unlike with the blade servers where space is optimised the Cisco routers could take up the same size as that is required to house a dedicated lab.

One of the advantages of using Emulab is that it allows the exploration of various concepts without causing risk to lab machines. Normally if a machine becomes unusable in a lab environment a technician would be required to restore the machine, which can take considerable time and effort. With Emulab if a machine gets corrupted then all that the user would have to do is reload the machine with a fresh copy of the operating system, thus saving time and effort.

Students sometimes require root access to explore various concepts and in a normal lab environment this will not be allowed, because of fear of the machine being made unusable. With Emulab this is not a problem because any changes to the operating system will be contained within Emulab and it can be restored quickly.

With falling hardware prices the cost of setting up an Emulab is not as expensive as might be expected, unless resources like Cisco routers are required and then it can get quite expensive [16]. One of the downsides to setting up an Emulab is obtaining and retaining the staff with the required experience to setup and maintain the environment.

2.2.5 Planetlab

Planetlab is a global network that spans the globe and can be used by researcher to emulate the Internet [18, 20, 22]. Planetlab works by having nodes, which is a server, spread throughout the globe. The node enables the building of large topologies that would not be possible to reproduce in a lab, and researchers can reserve time on these nodes to conduct experiments.

The connections between the nodes use real Internet links and as a result they are not configurable, this leads to the problem of finding useable nodes with the required bandwidth. The other problem with Planetlab is the competition with other researchers for the resources, as a result it is very hard to repeat the same experimentation with the same topology multiple of times.

Avoiding implementing new protocols directly on the Internet should be avoided as they may have undesirable effects that could play havoc with Internet infrastructure. This is why Planetlab is ideal for researchers to try out new ideas for the Internet without actually implementing them on the Internet.

2.2.6 Labs

Simulation is not just restricted for research, it can be used for teaching as well. One of the main problems with teaching specialist courses like networking or operating systems is that a dedicated lab is required to house all the necessary equipment [1]. For a dedicated training centre like QA training or Learning Tree this is fine as they have the income and lab personal, with the required experience, to support such labs. For a university, where space is at a premium, specialist labs can pose quite a problem due to the following issues [1, 17]: -

- The high setup costs of obtaining the infrastructure for the lab.
- The high maintenance cost of having to keep the lab equipment up to date with the latest developments.
- Trying to justify that the required space can be used profitably.
- Having to invest money in training staff up to the required level of experience to maintain the labs.

When a University starts out on a project like this they generally have to rely on obtaining funding or equipment donations from external sources [1].

There is a finite amount of lab equipment available to the students and that means that they will have to work in groups to complete the labs, this encourages group working but some students may prefer working on their own. When selecting groups there is the danger of a skill mismatch with the students, whereby the more experienced students will fly through the labs leaving the less skill students behind. Care must be taken to pair up students with the right skill experience [23].

Another problem with the specialist labs is that of security. Generally specialist labs will have equipment that is quite valuable, and if it went missing then it will be quite expensive to replace and it will also affect funding for future labs. As a result the specialist labs will generally have to be secured. Since some universities are not able to provide 24 hours security then this means that the specialist lab will have limited opening hours, and as a result students will have a restricted number of hours to complete the work.

When students are doing labs in class time they generally have an allotted amount of time to complete the labs, e.g. 2 hours. The initial step that the students have to undergo is the setup of the lab equipment, which can take some time and there is no guarantee that the configuration will be right. If the configuration is not correct then the students have to spend some time troubleshooting, which can be a good learning experience in itself but it can end up wasting a lot of the precious lab time. One of the other problems with the limited lab time is that it does not encourage the students to try and experiment with the configuration.

This is where simulation can help alleviate the problems mentioned. With simulation all the required configurations can be contained in one computer and the computer can also be used for other classes, thus reducing the need for dedicated lab space and also maximising the resources of the university.

This point is illustrated at Napier University in the e-security course. Napier does not have the money or space to house the equipment required for the Cisco portion of the course, to resolve this problem a program called Network sims [8] is used which simulate the required equipment and labs. As a result the simulated program can be installed on any computer in the university and accessed at any time. The student also does not have to wait for the resources to be made available before they can start studying, everything is available when they require and at any time of the day or night.

2.3 Virtualisation

Virtualisation has been around for a long time, but it is only recently with the advent of programs such as VMWARE [9] and XEN [10] and the increasing power of computers that it has gotten popular.

Virtualisation works by inserting a virtual machine monitor (VMM), also referred to as a hypervisor, on top of the actual hardware [24, 25]. The VMM is an abstract layer that behaves like the actual hardware of the computer. Once the VMM has been configured an operating system, called the guest operating system, can then be installed onto the VMM. The guest operating system can be any operating system that is supported by the VMM architecture, for example if the VMM was based on the x86 architecture then Linux, Windows 3.11, Windows XP and so on could be installed.

2.3.1 Goals of Virtualisation

The original three goals of virtualisation are as follows [26, 27]: -

1. Fidelity
2. Performance
3. Safety

Fidelity refers to the capability of the virtual machine monitor (VMM) or hypervisor to behave as if it is the actual machine while being able to run programs on the guest operating system without causing any problems.

Performance refers to the capability of the VMM or hypervisor to be able to execute the programs and match the performance as if it was being run on the actual machine. The performance may not match exactly due to virtualisation overheads but it should be as close as possible

Safety refers to the capability of the virtual machine to contain all of the programs and not affect any other operations being performed on the computer hosting the virtual machine. If one program in the virtual machine crashes then it should not have any effect on the programs being run on the host computer or any other VMM.

2.3.2 Types of Virtualisation

There are three types of virtualisation available [25, 27]: -

1. Full virtualisation
2. Paravirtualisation
3. Hardware assisted virtualisation

With full virtualisation everything is performed in software, with the VMM intercepting the instructions and either forwarding them unmodified, or altering them if access to hardware system resources is required.

Paravirtualisation requires the modification of the guest operating system. This modification allows the guest operating system to know that it is running on a

virtual machine and therefore allows it to communicate and co-ordinate with the hypervisor to achieve the best performance possible. XEN [28] supports paravirtualisation and as might be expected, paravirtualisation will not work on commercial operating systems such as windows XP because they are not capable of being modified by users.

Hardware assisted virtualisation is a recent development by chip manufactures AMD and Intel that allows the virtualisation to be done on the processor chip and thus hoping to achieve performance gains over software [27]. The initial results show that there is a moderate gain in some cases, but in other cases there is no gain at all over full virtualisation [27]. Hardware assisted virtualisation is still in its first generation though and as a result it is expected that in later generations there will be a greater performance gain as the technology matures.

2.3.3 Costs

Using virtualisation does not come without its costs [25]. A workstation may be able to run one VMM but any more could potentially end up making the workstation unusable due to insufficient resources. This is why a powerful server with plenty of ram and processing power is required to be able to run multiple VMMs successfully [29].

The overall speed of the guest operating system on a VMM will not be the same when compared against the operating system being run on a physical computer [25]. The reason for this is because running a virtual machine comes with several performance overheads ranging from I/O virtualisation, processing and memory [30]. On a system which is heavily utilised with VMMs then the overheads mentioned are aptly demonstrated, with each VMM competing with each other for the resources. This means that performance optimisation has to be performed to get the best performance from each system [31].

2.3.4 Optimisation

One of the questions that has to be asked is how many VMMs should be placed on a server. If too many VMMs are placed on a server then this will result in all of the VMMs suffering from resource starvation, because they are fighting for access to the resources [32]. Too few VMMs result in the server being under utilised, which means that the cost savings are not fully realised. To achieve optimal performance it is recommended that 50% process utilisation should be the desired goal [33]. This should allow all the VMMs to operate correctly while leaving the rest of the server resources free in case one of the VMMs has a sudden increase in demand.

2.3.5 Uses

The main area that has benefitted from virtualisation is data centres that host applications for customers [34]. Usually an application is run just on one computer and with no other applications running. This is so that if the computer or application goes down then no other mission critical services will be lost [25]. With only one application running this can result in the computer being underutilised for a vast majority of the time.

An alternative to using the one machine, one application approach is to use virtualisation to consolidate all the machines onto one powerful server. The isolation capabilities of virtualisation will insure that if either a VMM or the application running on the VMM crashes then it will not have any affect on the other VMMs running on the server [25].

Another area that has benefitted from the use of virtualisation is that of software development and testing [35]. A software developer usually has to develop for various operating systems (e.g. Linux, Windows, etc). Normally this would mean that they would have to have multiple development systems, but with virtualisation they are able to have one system that is capable of running different operating systems. This results in saved time since the developer would not need to switch between development machines all the time. There would also be a cost saving for the company since they would not have to purchase multiple development machines.

With testing one of the most time consuming activities is that of setting up the machine and installing all of the required software. For testing this has to be repeated multiple times and this can be quite time consuming. One of the features of virtual machine that can help reduce the time taken to do the installations is called check pointing [30].

Check pointing allows a snap shot to be taken of the virtual machine and to be stored as an image [30]. The image can then be reloaded any time in a matter of minutes (this is similar to using ghost to restore a machine). The tester would configure the machine with all the default software that is required and then take a checkpoint. If during testing a bug is found then check pointing can be used to preserve the state of the machine and then allow testing to proceed. This is a handy feature since the developer may not always be around to immediately debug the problem on the machine. It should be noted that when a checkpoint is done that some applications that are running at the time could not react well to being restored [33].

2.3.6 Benefits

Virtualisation can be seen in use in data centres, whereby the consolidation of multiple servers onto one using virtualisation results in the following [25, 27, 29, 32]

1. Reduction of the total power being used
2. Lower cooling required in the machine rooms
3. Reduction in the space required to store the machine, which results in more efficient use of space.

With using virtualisation all the VMMs can be run from one dedicated server using blade servers [29]. Blade servers are designed to be compact and to take up as little space as possible using a rack, and the rack also allows additional servers to be added as needed, thus providing easy expansion.

2.3.7 Education

The final area that will be examined where virtualisation is being used is that of education. In education virtualisation offers many of the same benefits that simulation provides but also provides several additional benefits that would not be possible when using simulation.

Like simulation, virtualisation allows the running of specialised courses without the need for an expensive dedicated lab [36]. This approach can be taken with courses such as security, networks and operating system learning.

Virtualisation has several benefits over simulation. One of the big problems with simulation is that the topology is generally fixed and does not allow for experimentation by the students [37]. Also sometimes the simulation does not provide a fully implemented interface, thus they do not experience the full range of commands. With using Virtualisation it allows the students to configure a topology and then experiment with “what if” questions while providing the full range of commands that they would find on a real device.

One of the biggest problems faced by lecturer’s running a lab is making sure that the students cannot interfere with the University’s equipment during labs, be it internet or computer. Protecting the Internet during lab time is easily solved by the ability of virtualisation to provide Internet isolation within the virtual machines [36, 38].

One of the nicest features of virtualisation is its ability to quickly recover from mistakes and problems [36]. When initially loading the machines a default image can be used and if the student makes a mistake, or makes the VMM unusable, then the image can be quickly restored and the student can continue the lab with little loss of time. If real computers were being used and this situation was encountered then the student would have to wait for the machine to be restored by a lab technician, who might not be available, and this may take a considerable amount of time.

With the students using virtualisation for labs it promotes hands on practice that helps the students to clarify theoretical concepts encountered in the lectures [37]. It also promotes active learning and experimentation that has been proven to have a more positive impact for learning and remembering.

2.3.8 Setup

One important issue that has to be taken into consideration when setting up a virtual lab is that of the cost of licences for the relevant software [33]. For every virtual machine set up a licence has to be acquired for the relevant software. One licence is not valid for all virtual machines.

2.3.9 Future Trends

Virtualisation is a growing area and is not just restricted to the areas mention. One new area that has been proposed is a new type of IDS based upon hypervisors [39].

Hardware virtualisation is another area that will grow as the chip manufacturers refine their techniques. Currently the first generation hardware virtualisation

does not offer the performance gain that was expected over software, but it is anticipated that this will change over time [27].

The current implementations of hypervisors offer a high degree of isolation that results in inefficient use of the available resources, an example of this is when the hypervisors each have to load a separate instance of the guest operating system. An alternative approach called containers has been suggested, which trades off isolation for a return in greater efficiency by allowing an operating system image to be shared between all the hypervisors [35]. The trouble with this approach is that it may open the possibility that if a hypervisor crashes then it may cause the other hypervisors on the machine to crash as well.

2.4 Education

The final section will examine the how learning takes place on a course. The section will begin by examining the cognitive architecture and then followed on by the cognitive load theory. The final part of this section will examine the most popular learning taxonomies and then followed by examining learning styles.

2.4.1 Cognitive Architecture

The cognitive architecture refers to how people store, process and organise memory [40-42]. No two people will operate in the same way and this should be taken into consideration when designing and presenting lectures [40].

There are two types of memory, working memory and long-term memory [41]. Working memory is used to build new information into constructs called schemas, which are then stored in long-term memory. The traditional view of working memory is that is capable of storing 7 ± 2 slots and this illustrates the limited capacity that is available to process new schemas.

Unlike working memory, which has a limited capacity, long-term memory has unlimited capacity to store schemas but it has to be remembered that long-term memory does not take part in the construction of schemas [41].

A schema is a construct that consists of memory elements. The schema itself can range from being quite simple and contain only a few memory elements, or it can be quite complex and contain numerous memory elements.

When new information is required to be stored a pre-existing schema is copied from long-term memory into working memory. When the schema is copied it will occupy only one of the slots in working-memory, no matter how complex the schema is. The new information is then amalgamated into the schema and once the process has finished it will be copied back into long-term memory. This process is known as schema acquisition.

Cognitive Load Theory

As has been previously noted all learning takes place in working memory, but this area of memory has limited slots to store information. People require mental energy to store and process new information. Over a long period of time a person's ability to process new information deteriorates and as a result the

learning process suffers. This is the theory behind the Cognitive Load, which tries to achieve the optimal load of working memory for efficient learning [5, 41].

There are three factors that make up the cognitive load [5]: -

- Extraneous Cognitive Load
- Germane cognitive Load
- Intrinsic cognitive load

Extraneous Cognitive load deals with external stimuli that are not related to the lecturer and thus is using up valuable working memory that could be used for learning.

Germane cognitive Load is related to how much working memory the student needs to process other data that is required for the new information being processed.

Intrinsic cognitive load relates to how complex the new information being presented is. For example a student with prior knowledge of a subject being presented may have low intrinsic load, however another student who is starting from first principals on the subject may have a high intrinsic load.

From the three factors mentioned every effort should be made to reduce the Extraneous load, to maximise the free working memory, while maximising the Germane load [5, 41]. One such technique that can be used to reduce the cognitive load is called fading worked examples [5].

Normally when a student has been presented with a complex example, working memory is required to be used to store all the details. This means that the student is storing information that does not help realise the learning outcomes and as a result increase the cognitive load that is placed upon them.

The idea behind fading worked examples is to present to the student a worked example but with certain parts missing. It would be up to the student to solve the missing parts without having to start from scratch.

The lecturer could present the student with a simple problem and then gradually increase the complexity of the problems as the student goes on. For the student only the information that is related to solving the problem need be stored in working memory and as a result the cognitive load will be reduced. This process also allows the student to start of with a simple schema in working memory. As the problems get more complex, then so does the schema in working memory since the student is getting a better understanding of the knowledge and how to use it.

2.4.2 Learning Taxonomy

There have been several attempts to model the cognitive learning process into learning Taxonomies, so that it can be used to gauge how far a student has mastered a particular subject. There have been various taxonomies proposed but only the most common ones that are used in education will be examined. The taxonomies that will be examined are Bloom's, revised Bloom and SOLO.

Bloom

Bloom splits the cognitive domain into six levels and each level builds upon the previous one [3]. The levels are shown in table 2-1.

Level	Description
1	Knowledge/Recall
2	Comprehension
3	Application
4	Analysis
5	Synthesis
6	Evaluation

Table 2-1 Bloom Levels

In Bloom's taxonomy Knowledge is the lowest level which represents the ability to remember information but not necessarily understand what is for, it could be compared to learning knowledge via parrot fashion. For example, what is a routing protocol?

The second level is Comprehension and it measures the understanding that the student has achieved of the knowledge learnt, and being able to describe it in their own words. For example, describe the OSPF routing protocol?

The third level is Application, and at this level the student is expected to be able to apply the knowledge to a particular scenario. For example implement OSPF in the network.

The fourth level is Analysis and at this level the student is expected to use the knowledge to analysis a problem. For example determining why a router interface is not routing packets.

The fifth level is Synthesis and this level the student is expected to combine various pieces of knowledge to build a solution to a problem. For example design a network from the beginning that uses both OSPF and EIGRP routing protocols.

The sixth and final level is evaluation where the student is expected to be able to argue the various merits of taking one approach over another. For example what are the benefits of using EIGRP over RIP in a particular situation?

The Bloom taxonomy is used in education to write course descriptors [3, 43]. The reason that this is used is so that all course descriptors are written using a universal standard and language, and it also allows lecturers to be made aware at which levels the students have attained knowledge at in previous modules.

An exam writer can also use Bloom to write questions that will test the student's knowledge at various levels [43]. For example if students are expected to be at the Application level, defined from the course descriptor, then questions can be written that will test the student at this level.

Bloom is used through the students University life. The aim is to progress the student through the taxonomy during their course [4]. So when the student

starts in the first year they will be working at the bottom three levels, known as low order thinking, but by the time they graduate the aim would be to have the students working at the top three levels, known as high order thinking.

One of the big problems with Bloom is that it is very subjective [3]. One lecturer may think that they are testing at the application level, but when another lecturer reviews the test they may think that the level is at the comprehension. Bloom is imperfect and there is not a uniform way of applying it, but it is the most applicable and popular model.

Revised Bloom

Recently the bloom model has been revised [4]. The changes include using verbs instead of nouns to describe the levels, and instead of the levels being flat another dimension has been added to represent the types of knowledge being learnt. The changes are represented in table 2-2

Level	Factual	Conceptual	Procedural	Metacognitive
1.Remember				
2.Understand				
3.Apply				
4.Analyze				
5.Evaluate				
6.Create				

Table 2-2 Revised Bloom

The revised bloom taxonomy is a recent change and as a result there isn't much evidence of the revised model being used. From the papers viewed Bloom's original model is still the most quoted.

SOLO

While Bloom can be used to both specify the learning outcomes for the module/course and to create exam questions, it is not used to gauge how well the student has answered the question. A taxonomy called SOLO can be used to gauge how well somebody has answered the question, e.g. is there any structure to how the question was answered or is it just a list of facts with no structure behind the answer [3, 4].

There are five SOLO levels and they are described in table 2-3

Level	Name	Description
1	Prestructural	Does not answer the question posed in any way or form.
2	Unistructural	Answer focuses on a simple meaning and does not go into any depth
3	Multistructural	A disorganised list with no order
4	Relational	Shows that the student understands how to apply the theory to a problem
5	Extended abstract	Shows that the student has read around the area and answered at a level that illustrates understanding at the next level.

Table 2-3 SOLO Levels

The student's answer to the question can be marked based upon what level the answer is judged to have reached in the SOLO level. This is a good way to insure that the student doesn't just remember the facts to answer a question, the student gains mark for being to answer the question in a certain structure.

Course Design

The Myers-Briggs indicator shows that there are 4 classifications of personality and they are introvert-extrovert, sensing-intuitive, thinking-feeling and perceiving judging [44]. As can be seen there are various types of personality and because of this each person will learn in different ways. The lecturer must be aware of this and realise that one teaching style will not suit all students. As a result the lecturer should design the lecture to spread a wide range of learning styles to appeal to as many students as possible and to retain their interest during the class.

Collaboration Learning

One of the biggest problems that employers find with university graduates is their lack of soft skills like group working, social skills, self-motivation, cooperation, communication and work ethic [45-47]. Most University courses are focused on learning the technical skills to the detriment of everything else, and thus not producing fully rounded personal for the working environment [46].

A new way of teaching called collaborative learning can be used to resolve the problems that are encountered with traditional teaching methods. In Collaboration learning students are split into small groups and asked to work on problems together [46, 47]

With the students working in groups it promotes an exchange of ideas, discussion and conflict in trying to solve the problem within the group dynamics [46]. Trying to solve the problem as a group helps to reinforce ideas presented in the lecture, since it is being discussed openly any misconceptions with the material presented will be highlighted. Instead of the learning being primarily

passive it has now gone to being active, which will help the students retain the information and have a better interest in the subject.

For the students group working can be quite challenging if they have been working on their own for the majority of their course. By using collaboration teaching it should help prepare the students better for the working environment, and also make them more employable by focusing on some of the skills that businesses require.

Kinesthetic

The traditional way of learning in computer science is to use the lecture based approach [44]. One of the main problems with this approach is that it is seen as being passive in that all the students are just listening to the lecturer and doing nothing else. For some students this would be fine, but for other students they may lose focus and interest with the subject being presented. The reason for this happening is because different students learn in different ways

Recently the number of computer graduates has fallen and this is a continuing trend [23]. One of the biggest problems is that learning in the computer area has been primarily done using passive teaching and computer science has a lot of abstract concepts that are not easy to grasp, and using a passive teaching in a class does not help all of the students to understand the concept fully [23].

A way to help the students understand the concepts and at the same time to make learning more fun is to use Kinesthetic learning [44]. The idea behind Kinesthetic is to promote learning through doing using physical props, games and competitions. Rather than the teaching being passive where the student is not involved, using Kinesthetic, the student is now active and participating in the lesson.

Kinesthetic makes learning more fun for the students and has been shown to help the students learn about abstract concepts encountered in computer sciences [44]. It is hoped that using new learning techniques like Kinesthetic will help to retain more students in the computing field.

3 Design

3.1 Introduction

This chapter details the design that will be undertaken for three outputs. The first output is a framework that will allow students to practice networking concepts on Cisco emulated routers using the actual IOS operating system. The second output will be the results of a questionnaire that will be answered by the students who used the previously mentioned program. The final output will be a set of experimental results that will compare the performance of using the framework on various operating systems.

3.2 Challenge Generator Program Design

This section details the design steps that will be undertaken to produce a framework that will be capable of giving the student a network challenge that they will be able to configure using emulated routers.

The block diagram (figure 3-1) shows the proposed framework design.

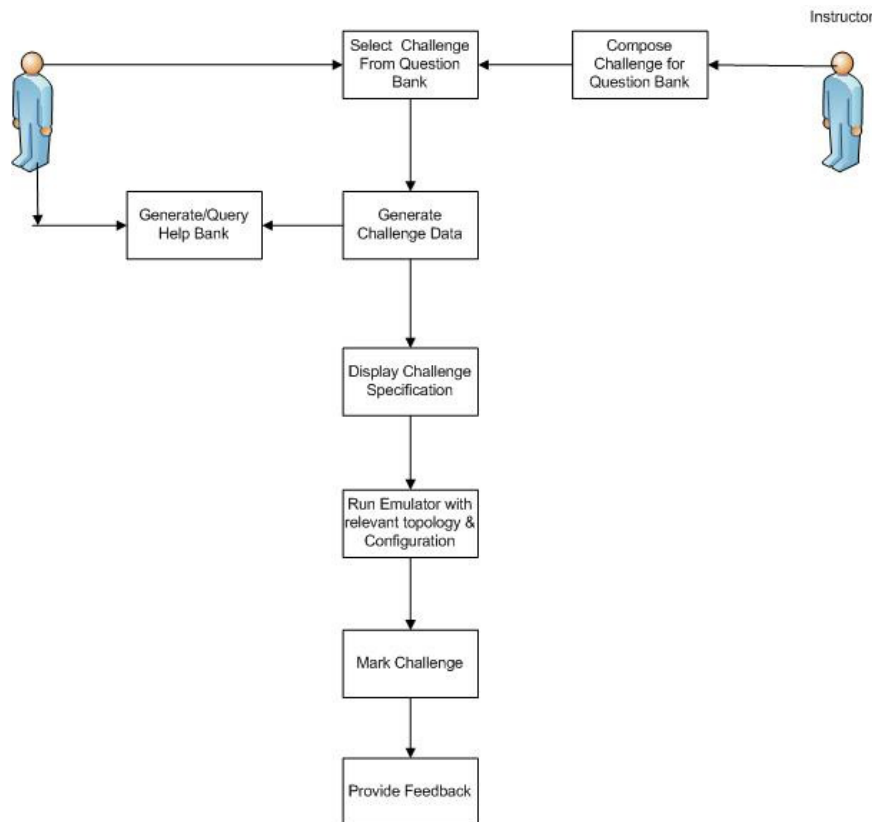


Figure 3-1 Program Design Block Diagram

The following sections will describe the functionality of each process, starting with the compose challenge process.

3.2.1 Compose Challenge

Before the student is able to select a challenge the instructor must create the challenge itself for inclusion in the question bank.

The first step is to define the network topology that will detail what routers are being used and how they are connected to each other. The next step is to create the startup-config files that are required for the challenge, followed by composing help data for the user in case they require assistance during the exercise. The next step is to write the functions that will generate the dynamic variables when the challenge is selected. The final step is to write the functions that will dictate how the answer will be marked.

Now that a challenge is available to be selected, the next process that will be examined is select challenge.

3.2.2 Select Challenge

It is envisaged that the user will have a selection of challenges that they will be able to perform and as a result the user needs to have a way of selecting the challenges. The user will be presented with a menu from which they will be able to select from either a specific challenge that they would like to repeat, perform a set of related challenges or to take a random challenge.

Once the challenge has been selected then the dynamic variables relevant to that challenge can be generated

3.2.3 Generate Challenge Data

Now that the challenge has been selected the dynamic variables such as IP addresses, host names, EIGRP autonomous systems numbers and so on can be generated. The idea behind this is to make each instance of the challenge unique and to keep the attention of the student during multiple attempts. This issue is discussed further in the challenger design section. Once the challenge data has been generated then the help data can be loaded.

3.2.4 Generate/Query Help Bank

The next step is to generate a help bank that is specific for the challenge. The idea behind the help bank is that if a student gets stuck on a challenge then they will be able to consult the help bank for clues as to how to solve the problem. Marks will be deducted from the students overall score each time that they consult the help bank.

3.2.5 Display Challenge Specification

Once the challenge data has been generated the user will be shown the specification of the exercise that they have to configure

3.2.6 Run Emulator

Once the challenge has been selected and the relevant challenge data generated then the selected emulator engine can be started with the location of both the topology configuration and startup-config files required for the challenge.

3.2.7 Mark Challenge

When the student has finished their configuration they will then be marked based upon how long they took to complete the challenge, how many times the student had to access the help feature, how many times the student had to use the “?” feature of the Cisco IOS firmware and the number of commands used.

The portion of the mark that is comprised of the time taken will be computed, by comparing the student’s time against a time that is seen as the best average for completion of the task. If the student is over the time then they will get less marks, however if the student is under the time then they will get more marks.

If the student required frequent access to the help feature for the challenge and the Cisco IOS firmware then the score will be reduced, as it will show that the student does not know the relevant commands for completing the task.

3.2.8 Provide Feedback

The final process that will be examined is that of providing feedback to the student. If the student has not managed to complete a task successfully then they will be informed, and also shown how to solve the problem correctly. If the student has managed to complete the task then they will be told so.

3.3 Data Flows

This section shows how the data flows from process to process. Figure 3-2 shows the data that is generated from the create challenge process.

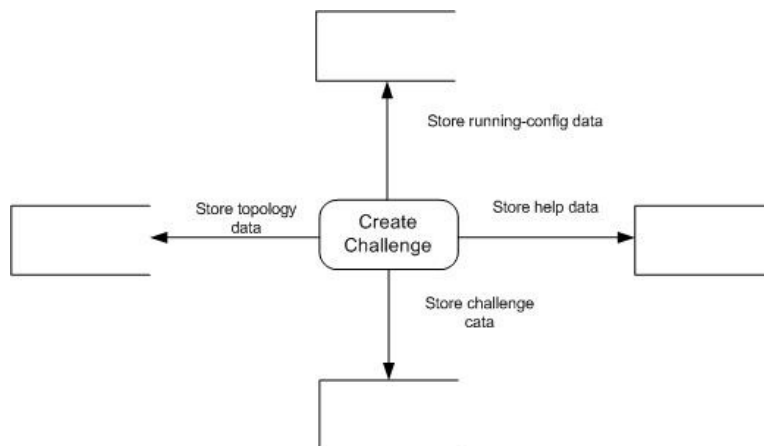


Figure 3-2 Create Challenge Data Flow

Figure 3-3 shows the data flow from when the student starts the challenge to when they finish the challenge

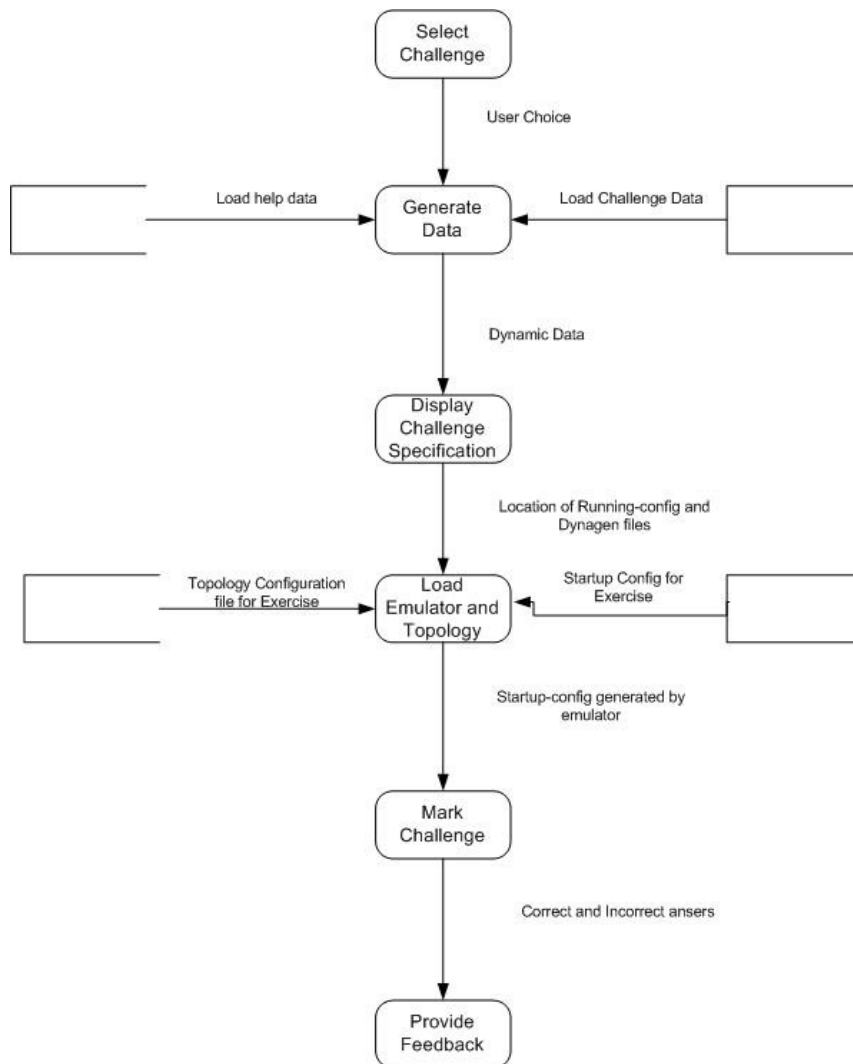


Figure 3-3 Data Flow Diagram

3.4 Challenges

This section will begin by discussing the problems with Cisco labs and offering some alternatives that would improve the learning process for the student. This will then be followed by a discussion of the challenges that were designed for use with the emulator.

3.4.1 Cisco Labs

This section will briefly examine how the Cisco labs are structured and what the problems are with following this structure, before finally offering some alternative suggestions.

Lab Specification

The general anatomy of a Cisco exercise is to present a network diagram with the entire configuration shown on the first page. Immediately on the next page the student is shown how to do the configuration. The student is not encouraged to

find out how to do the configuration by themselves, and thus becoming more actively engaged in the learning process. If the student is not properly motivated to learn in the right way there is the danger that the exercise would turn into nothing more but a typing exercise.

A better process might be to present the network diagram and then get the student to write down the commands as to how they would configure it. This would involve the student having to review the Cisco academy lessons for all the relevant commands. Another alternative approach could be to show the student how to do a generic example and then in the following exercises show them a network diagram and then ask them to do the configuration without providing step-by-step instructions.

The changes suggested could engage the student more and also help them to learn the material better. There is the down side though that the lab exercises would take longer to complete. Further study would be required to determine if this is a better way of learning the material.

Lab Setup

Another problem with the Cisco exercises is the repetition involved in setting up some of the labs. In the Cisco exercises there may be a whole series of labs to complete and in the majority of times the student will have to start the configuration and wiring of the network from the beginning.

Getting the student to repeat the setup and configuration of the labs is ok for when the student are just starting to learn, since it reinforces how to do things, but it could become tedious and boring for the more experienced students.

Repeating the setup and configuration also results in lost time in the learning process, since the student will have to do the initial configuration and they are not really learning anything during this time. A better way to speed up the initial configuration could be to provide partial configurations that would provide the initial configuration and thus speed up the learning process [5].

3.4.2 Challenge Generation

The following sections will examine the how the challenges for the framework were designed. For the initial proof of concept there will be three challenges and each of the challenges will build upon each other.

All of the challenges are based upon the concept of fading work examples [5] that present a partial configuration to the student, and then asking them to complete the problem. By providing a partial configuration it allows the student to focus on the topics that are being taught, while reducing the cognitive load by not having to spend time on remembering concepts that are out with the task.

As the student progresses through the challenges it can be seen that they are getting progressively more complex and require more thought and analysis. The final two exercises test the student's diagnostic and debugging skills. These types of skills, which Industry requires of graduates, are very difficult to teach in University's and they are generally learnt on the job [46, 47]. By using the emulator program it allows the developer to devise challenges that will allow the student to develop these skills.

Challenge 1

The initial challenge will involve two routers in the configuration shown in figure 3-4



Figure 3-4 Challenge 1 Network Topology

The student will be asked to perform the following tasks on R0: -

1. Configure a hostname on R0 using the stated parameters
2. Configure an IP address on FastEthernet0/0 using the stated parameters
3. Bring up the FastEthernet0/0 interface

The student will be asked to perform the following tasks on R1: -

1. Configure a hostname on R1 using the stated parameters
2. Configure an IP address on FastEthernet0/0
3. Bring up the FastEthernet0/0 interface

In the specification for both R0 and R1 the hostnames to be configured will be specified from a set list of names. The IP address and subnet mask specified will be randomly generated from either class A, B or C. The reason for the randomness of the hostname and IP addresses is to get the student to think about what they are typing. If the values were static instead of dynamic, then there is the danger that when the student repeats the exercise they would be doing it from memory instead of doing active learning. This again is a weakness of the Cisco labs in that the specification is not dynamic.

One of the main problems when the student is asked to configure a network is that the student implicitly trusts the simulator and does exactly what is asked of them and no more. In real life this is not always the case as a task may be given to the student that has implicit requirements.

The above specification details explicit requirements; however there are also implicit requirements that have to be performed for the student to successfully complete the task. In this exercise the implicit requirement is

1. Copy the running-config to the startup-config

When the student has finished the exercise the program will evaluate the running-config file and report if the student managed to fulfil the requirements. If any of the requirements have been failed then the student will be informed and the correct command for satisfying the requirement will be displayed.

Each of the challenges will be ranked based upon the Bloom model [3, 43]. The reason that the Bloom model was chosen instead of the revised Bloom [4] is because this is the most prevalent model being used. Also as it has been noted previously Bloom is the most quoted taxonomy used in research papers and the

papers reviewed for this thesis shows that the revised Bloom is not being widely used yet.

The more Bloom levels used, the more difficult the challenge will be. In this challenge the bloom levels tested are shown in table 3-1.

Description	Used
Knowledge/Recall	X
Comprehension	X
Application	X
Analysis	
Synthesis	
Evaluation	

Table 3-1 Bloom rating for Challenge 1

In this challenge the student is expected to be able to comprehend the specification and know what steps they have to take to be able to complete the task. Once the student has understood the specification then they will have to solve the problem to the best of their ability.

Challenge 2

The second challenge builds upon the first challenge and will again involve two routers. The network topology for challenge is shown in figure 3-5



Figure 3-5 Challenge 2 Network Topology

In this challenge instead of the student having to repeat the configuration of the previous challenge again, the routers will load with a startup-config that will already have these requirements pre-configured, see Appendix A for the startup-config files used. By having the routers pre-configured, it allows the student to focus on the objectives of the challenge and thus both reducing the cognitive load [5, 41] and time spent on doing things that they already know how to do.

The student will be asked to perform the following tasks on R0: -

1. Configure 4 loopback addresses using the stated parameters
2. Configure EIGRP using the stated parameters
3. Advertise the loopback addresses into EIGRP

The student will be asked to perform the following tasks on R1: -

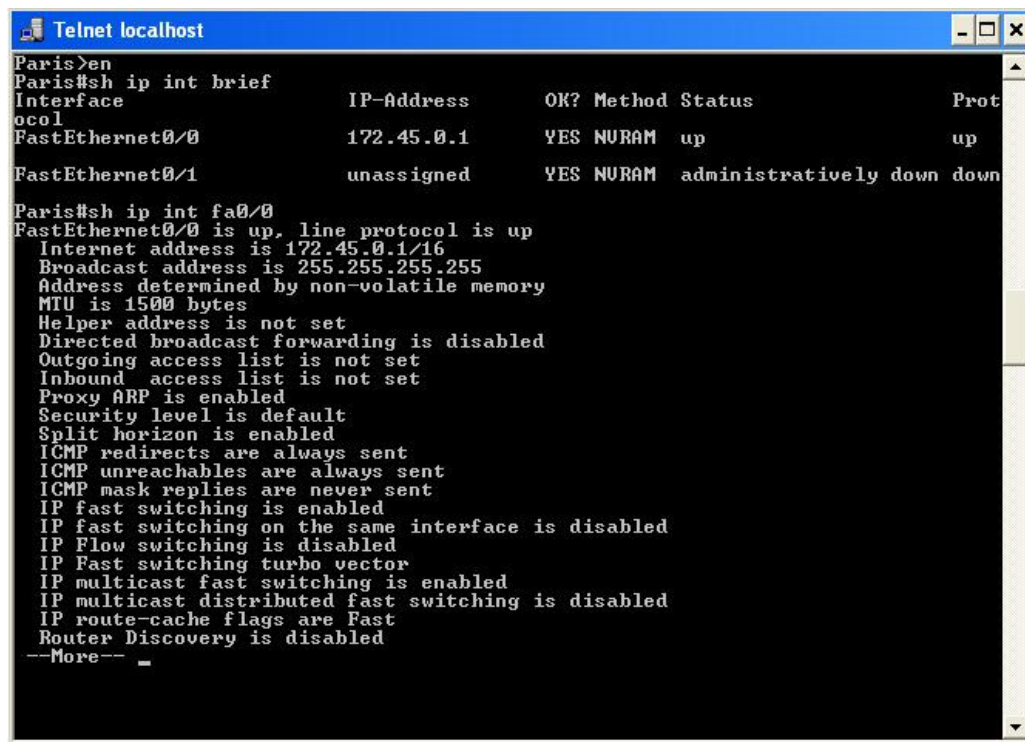
1. Configure 4 loopback addresses using the stated parameters
2. Configure EIGRP using the stated parameters
3. Advertise the loopback addresses into EIGRP

As with the previous challenge there are some dynamic elements that force the student to think about what they are doing, instead of blindly typing the same information time and time again. In this exercise the EIGRP numbers are randomly generated between 1 and 65535, while the loopback addresses are a randomly generated class C IP address.

Again there are implicit tasks that have to be configured properly before the student has successfully completed the challenge. These are: -

1. Copy the running-config to the startup-config
2. Advertise the directly connected links between R0 and R1 into EIGRP

In this challenge all the pertinent information is not given to the student, unlike in the previous exercise. Before the student can advertise the directly connected links between R0 and R1, they will have to find out the IP subnet and subnet mask used using IOS diagnostic commands. See figure 3-6 for an example diagnostic screenshot.



```
Telnet localhost
Paris>en
Paris#sh ip int brief
Interface                               IP-Address      OK? Method Status  Prot
FastEthernet0/0                         172.45.0.1      YES NURAM  up      up
FastEthernet0/1                         unassigned      YES NURAM  administratively down down
Paris#sh ip int fa0/0
FastEthernet0/0 is up, line protocol is up
Internet address is 172.45.0.1/16
Broadcast address is 255.255.255.255
Address determined by non-volatile memory
MTU is 1500 bytes
Helper address is not set
Directed broadcast forwarding is disabled
Outgoing access list is not set
Inbound access list is not set
Proxy ARP is enabled
Security level is default
Split horizon is enabled
ICMP redirects are always sent
ICMP unreachable are always sent
ICMP mask replies are never sent
IP fast switching is enabled
IP fast switching on the same interface is disabled
IP Flow switching is disabled
IP Fast switching turbo vector
IP multicast fast switching is enabled
IP multicast distributed fast switching is disabled
IP route-cache flags are Fast
Router Discovery is disabled
--More--
```

Figure 3-6 Challenge 2 Diagnostic Screenshot

By including this type of challenge it allows the student to practice their diagnostic skills. With the Cisco IOS there are various ways of obtaining the required information and since the actual IOS is being used it allows the student free reign to gather the information in any way. If a simulator were being used

then the student would be restricted to the available commands implemented by the simulator [1, 2].

Like in the first challenge detailed feedback will be provided to the student once the challenge is completed.

For Challenge 2, table 3-2 shows the Bloom levels that will be tested.

Description	Used
Knowledge/Recall	X
Comprehension	X
Application	X
Analysis	X
Synthesis	
Evaluation	

Table 3-2 Challenge 2 Bloom Rating

In this challenge the student is expected to be able to comprehend the specification and know what steps they have to take to be able to complete the task. Once the student has understood the specification then they will have to solve the problem to the best of their ability. Finally the student will have to analysis the router configuration to determine the IP addresses of the link connecting R0 and R1 for inclusion in the EIGRP routing.

Exercise 3

The third and final challenge again builds upon the work done in the previous two challenges. The network configuration for the third challenge is shown in figure 3-7.

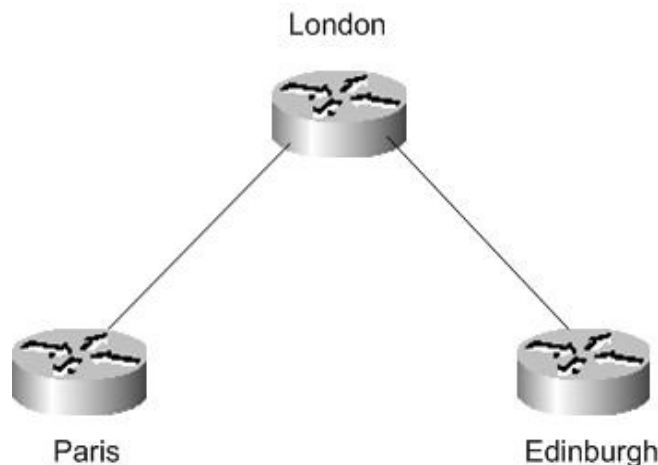


Figure 3-7 Challenge 3 Network Topology

Like in the previous challenge the student will not have to configure everything and in this challenge, the links between Paris and London and London and Edinburgh have already been configured along with 4 loopback addresses on

both London and Paris. EIGRP has also been configured between London and Paris. See Appendix B for the startup-config files used.

It is assumed that the student has already mastered the skills in the first two challenges and as a result there is no need to further test them on this. This has the desired effect of lowering the cognitive load on the student and also allows them to focus on the skills that are trying to be taught.

The student will be asked to perform the following tasks on London.

1. Configure OSPF on the router
2. Redistribute the EIGRP routes into OSPF
3. Redistribute the OSPF routes into EIGRP using the stated parameters

The student will be asked to perform the following tasks on Edinburgh: -

1. Configure 4 loopback addresses using the stated parameters
2. Configure OSPF on the router

Again there are implicit tasks that have to be configured properly before the student has successfully completed the challenge. These are: -

1. Copy the running-config to the startup-config
2. Advertise the directly connected links between London and Edinburgh into EIGRP
3. Enable the link between London and Edinburgh

Like in the first challenge detailed feedback will be provided to the student once they have completed the exercise.

In this challenge the bloom levels that will be tested are shown in table 3-3.

Description	Used
Knowledge/Recall	X
Comprehension	X
Application	X
Analysis	X
Synthesis	
Evaluation	

Table 3-3 Challenge 3 Bloom Rating

For Challenge 3, the student is expected to be able to comprehend the specification and know what steps they have to take to complete the task. Once the student has understood the specification then they will have to solve the problem to the best of their ability. Finally the student will have to analysis the router configuration to determine the IP addresses of the link connecting London and Edinburgh for inclusion in the OSPF routing, and also the state of

the interfaces to realise that they have to enable the links between London and Edinburgh.

3.5 Questionnaire

Once the framework has been created there needs to be a way to evaluate it to determine if it meets certain criteria. The evaluation method chosen is to use volunteers to go through the three exercises and then ask them to fill in a questionnaire about their experiences using the program.

The questionnaire will be designed to have five options from which the user can select an answer. The options available will be Strongly Agree (SA), Agree (A), Neutral (N), Disagree (D) and Strongly Disagree (SD).

The volunteers will be asked the following questions based on their experience of using the program: -

- The level of feedback was adequate for the student to learn from the mistakes made?
- The program was easy to use and set up.
- The specification of the challenges was stated in a clear manner.
- Not having to start from scratch in each challenge saved time and allows the challenge to be completed quickly.
- Using a fully functional IOS is better than the restricted IOS used in simulators (e.g. full use of the back and forward arrow keys for the command history)?
- The challenges maintain my interest throughout.

For the purpose of this evaluation ten volunteers will be asked to complete the challenges and then fill in the questionnaire. The results will then be compiled in a table and evaluated. The evaluation of the results of the questionnaire will allow the developers to gauge where future improvements could be made.

3.6 Footprint Evaluation

There are several elements that make up the overall framework. The components are comprised of GNS for the graphical interface, Dynamips that is responsible for the emulation and finally the challenge generator program that generates the exercises and marks the end result.

With all these elements working together the footprint that they impose on the operating system must be determined, this is to verify that everything can run together while proving to the student a viable working environment.

To determine the footprint of the overall framework, performance monitor will be used to monitor the CPU performance, memory and hard disk activity. Performance monitor will be configured to take a reading every 5 seconds.

This experiment will be repeated ten times using the following operating systems: -

- Windows XP
- Windows XP running in Emulation mode using VMWARE Fusion on a Macintosh computer.
- Windows Vista Business

The reason that Windows XP and Vista were chosen is because they are seen as being the most predominant operating systems in the market place today, and as result they are most likely the operating systems that will be used to run the applications. The reason that the XP emulation on a Macintosh option was selected is because today people are choosing to run applications on virtual machines.

In each instance the following performance counter objects to be monitored are listed in table 3-4.

Component	Performance Counter	Used For
Disk	Physical Disk (instance)\Disk Transfers/sec counter	Disk Transfers/sec is the rate of read and write operations on the disk.
Memory	Memory\Pages/sec	Indicates the number of paging operations to disk during the measuring interval
Processor	\Processor(_Total)\% Processor Time	The percentage of time all threads spend using the processors

Table 3-4 Windows Performance Counters

During the execution of the tests no other programs will be running, only those required to provide the proposed framework. The Monitoring will not start until after all the elements are in the running state. This is because when a program is starting there is an increased demand on the resources for a short period of time, and this may cause an anomaly within the results.

The experiments will be run with a topology of 3 routers. Each of the routers will be running the BGP routing protocol because of the demands it places on the processor and memory components.

A program will be created that will allow the automatic gathering of performance counters and logs. For every minute there are 12 readings, the program will take the 12 readings and produce an average reading for the minute. Since the performance counter will be running for an hour, this will give a total of 60 readings per experiment.

3.7 Conclusions

In this chapter a framework has been proposed using a block and data flow diagrams. Each process in the framework was described. The next phase of the framework that was described was the generation and solution of the challenges. The challenges were based upon the fading worked examples and the Bloom taxonomy was used to assign a difficulty level.

The evaluation of the proposed solution is going to be done in two phases. In the first phase the performance of the solution is going to be monitored using a BGP network topology and performance monitor. The second phase of the evaluation will use a questionnaire and volunteers with network experience to test the proposed solution.

4 Implementation

4.1 Introduction

This chapter will be split into three sections. The first section will detail the design steps that were taken for the framework. The second section will discuss how the footprint evaluation was performed and the third section will discuss how the questionnaire was carried out.

4.2 Framework Implementation

This section will detail how the framework design that was discussed earlier was implemented. For the implementation a rapid prototyping development approach was used since only a proof of concept was required and this would also allow a quick development time. It should be noted that the final output should not be taken to mean that all functionality is present and correct.

Each section will show how a process was implemented, starting off with the selection of the router emulator. This will then be followed by composing a challenge for the question bank, generating challenge data, display challenge specification, running the emulator, marking the challenge and finally providing feedback.

4.2.1 Router Emulation

As has been noted in previous sections one of the biggest problems with using a Cisco simulator is the fact that the full IOS operating system is not available to the user, which usually means that only a subset of the commands are available[2]. This means that a user could configure a link between two routers but they would have no way of being able to verify that the link is operational before finishing the exercise.

To overcome this problem it was decided that an emulation package called Dynamips [10] would be used. Dynamips works by using a virtual machine monitor (VMM) to provide the router architecture, each class of router (2600, 3600, 3700, 7200) requires a different architecture. Once the VMM has been selected the router's firmware (IOS image) can now be loaded onto the VMM. Since the VMM is an actual copy of the actual router hardware it means that no modifications have to be made to the router firmware. As a result the user will have at their disposal a fully operational command line operating system.

In a physical lab the student would have to connect the network devices together using Ethernet and serial cables. For Dynamips this process is replicated using an application called Dynagen [10]. The configuration for Dynagen is stored in a file with a similar structure to that of an INI file. See Appendix C for a sample of a Dynagen configuration file.

One of the main problems that users need to be aware of when running Dynamips, is that the processor utilisation jumps to 100% regardless if the router is being used or in the idle state. The reason for the processor utilisation being stuck at 100% is because Dynamips cannot determine when the router is doing useful work or when it is being idle.

To resolve the processor utilisation problem an attribute called “idlepc” can be calculated. The idlepc value represents code points in the IOS where the virtual router is performing an idle loop. When dynamips determines that the router is executing one of the idle codes points it can put the router to sleep for a while, and thus reduce the amount of CPU process required.

GNS3

One of the main problems with using Dynamips is that it is a text-based application. This means that if a user were asked to configure a network they would have no visual representation of how the network looked. There is an additional application called GNS3 [48] that is available that can be used to provide a GUI interface for the user. See figure 4-1 below for a screenshot of the GNS3 interface.

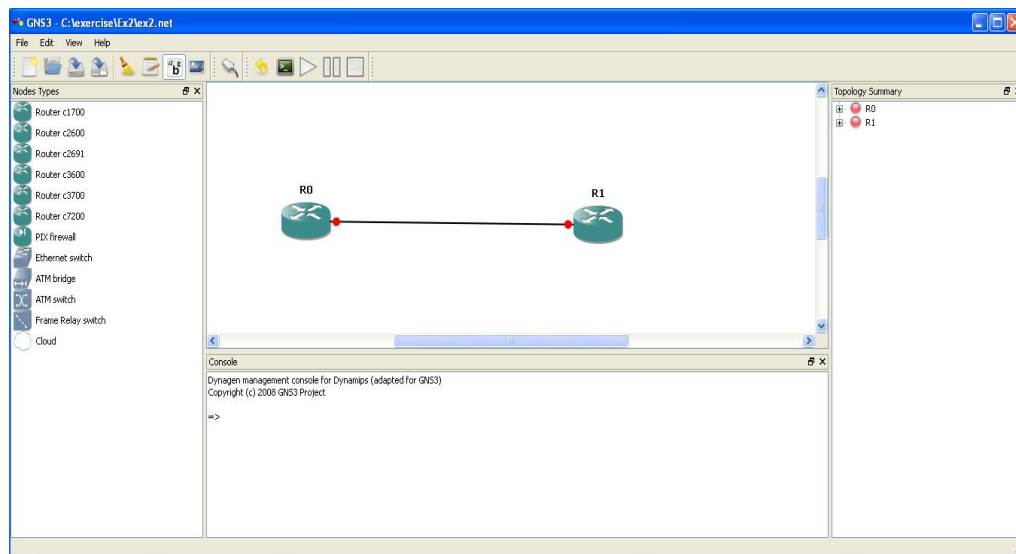


Figure 4-1 GNS3 Screenshot

The window pane on the left shows the network devices that can be used with the emulator. The top middle pane is used to present a graphical representation of the network topology and how all the devices are connected together. The bottom middle pane is called the hypervisor console window and is used to interact with all hypervisor instances that are in the topology. The hypervisor console can be used to perform actions such as extracting the startup-config from the routers or performing additional physical configuration of the routers. The window pane on the right is used to show the active routers and the network connections between them.

As can be seen from the screenshot GNS3 provides a good graphical representation of the network topology to the user and it also provides a more friendly user interface. To speed up the implementation it was decided to use GNS3 to provide the graphical interface to the user instead of writing one from the beginning.

Virtualisation

There is very little documentation available as to how the virtualisation is carried out in Dynamips. One of the major requirements for virtualisation is that of isolation between hypervisors, so that if one hypervisor crashes then there will be no affect on the other hypervisors that are running on the same machine, also known as the safety goal.

The image in figure 4-2 shows the windows task manger with three instances of dynamips-wxp.exe running.

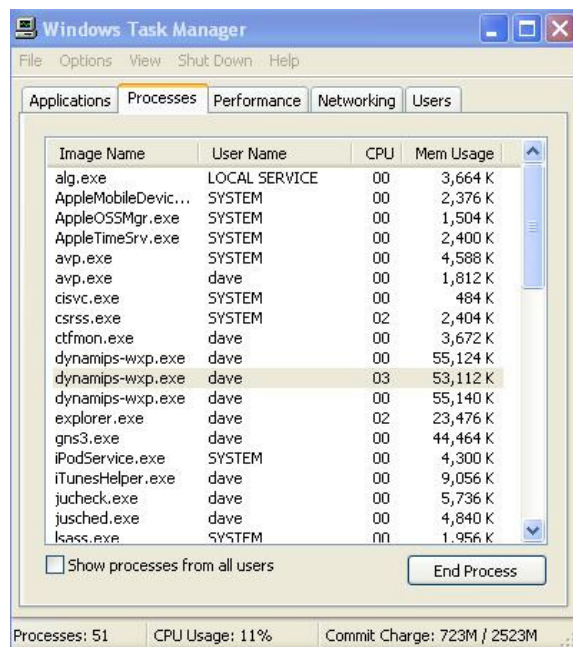


Figure 4-2 Dynamips in Task Manger

Each instance refers to a router hypervisor that matches the topology shown in figure 4-3.

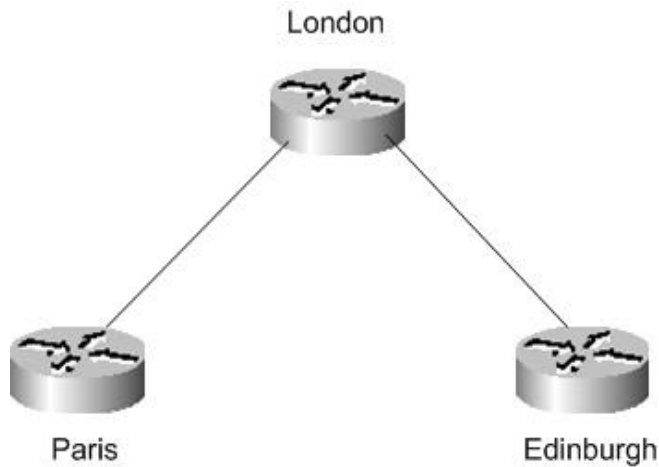


Figure 4-3 GNS3 Network Topology

To verify that the safety goal has been met it was decided to end one of the dynamips-wxp.exe hypervisors and the end result of this operation is shown in figure 4-4.

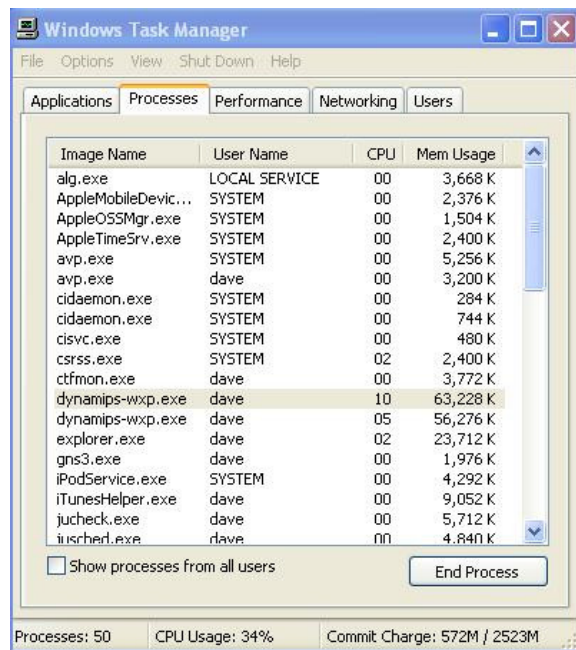


Figure 4-4 Dynamips Hypervisor Killed

As can be seen from figure 4-4 there is now only two hypervisors that are running, and to meet the safety goal the other two hypervisors should be running with no ill effects from the termination of the other hypervisor, as might be expected there will be a disruption in the network operation. The termination of the dynamips-wxp.exe process stopped the Edinburgh router and this is shown in figure 4-5 by the telnet session being disconnected.

```

Telnet localhost
O E2 209.165.199.0/24 [110/20] via 172.16.0.1, 00:00:30, FastEthernet0/0
O E2 220.88.24.0/24 [110/20] via 172.16.0.1, 00:00:30, FastEthernet0/0
O E2 209.165.189.0/24 [110/20] via 172.16.0.1, 00:00:30, FastEthernet0/0
O E2 209.165.169.0/24 [110/20] via 172.16.0.1, 00:00:30, FastEthernet0/0
O E2 220.88.14.0/24 [110/20] via 172.16.0.1, 00:00:30, FastEthernet0/0
Edinburgh#
Edinburgh#
Edinburgh#
Edinburgh#sh ip int brief
Interface      IP-Address      OK? Method Status      Prot
ocol
FastEthernet0/0 172.16.0.2      YES NVRAM  up          up
FastEthernet0/1 unassigned      YES NVRAM  administratively down down
Edinburgh#ping 192.168.10.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.10.2, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 116/138/168 ms
Edinburgh#
Connection to host lost.

```

Figure 4-5 Edinburgh Router Losing Connection

Any further attempts to connect to the Edinburgh router results in failure. Figure 4-6 shows a screenshot that verifies the Paris router is still active and shows three ping attempts being made.

```

Telnet localhost
Loopback2      209.165.189.1   YES NVRAM  up          up
Loopback3      209.165.199.1   YES NVRAM  up          up
Loopback4      209.165.169.1   YES NVRAM  up          up
Paris#ping 172.16.0.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.0.2, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 72/104/168 ms
Paris#ping 172.16.0.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.0.2, timeout is 2 seconds:
*****
Success rate is 0 percent (0/5)
Paris#ping 172.16.0.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.0.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 36/65/76 ms
Paris#

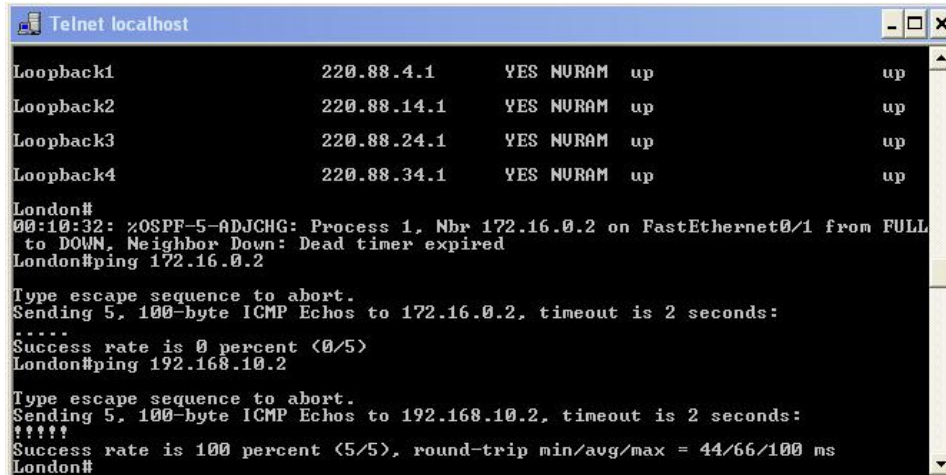
```

Figure 4-6 Paris Router Ping Response

The first ping was done before the Edinburgh hypervisor process was killed and shows that there is full connectivity between the routers. The second ping was done after the Edinburgh hypervisor was killed, and shows that the pings are no longer getting through to Edinburgh and verifies that the router is no longer

operational. The third ping is used to verify that there is still connectivity between the Paris and London router.

Figure 4-7 shows the London router.



```
Telnet localhost
Loopback1          220.88.4.1      YES NVRAM  up          up
Loopback2          220.88.14.1     YES NVRAM  up          up
Loopback3          220.88.24.1     YES NVRAM  up          up
Loopback4          220.88.34.1     YES NVRAM  up          up
London#
00:10:32: %OSPF-5-ADJCHG: Process 1, Nbr 172.16.0.2 on FastEthernet0/1 from FULL
to DOWN. Neighbor Down: Dead timer expired
London#ping 172.16.0.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.0.2, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
London#ping 192.168.10.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.10.2, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 44/66/100 ms
London#
```

Figure 4-7 London Router Ping Response

As can be seen from figure 4-7, the OSPF neighbour connection to Edinburgh has been torn down. The first ping shows that there is no connectivity towards Edinburgh but the second ping shows that there is still a connection between London and Paris.

As has been demonstrated the safety goal of virtualisation has been met. The second goal that also has to be satisfied is that of fidelity and this is achieved by the usage of the actual Cisco IOS. Due to time restrictions the final objective of performance was unable to be verified and this will have to be verified by future experiments.

4.2.2 Compose Challenge

This section will describe the process undertaken for designing the challenges. The challenge that will be described is Challenge 2, but the steps described are the same for all the other challenges and thus there is no need to go into further discussion for these challenges.

For the design Challenge 2, the following three steps were followed

1. Generate network topology
2. Generate running-config files
3. Write the code for generating the dynamic variables

Generate Network Topology

The generation of the network topology was achieved by using the graphical properties of the GNS3 program. Figure 4-8 shows the end result of creating the topology for Challenge 2.

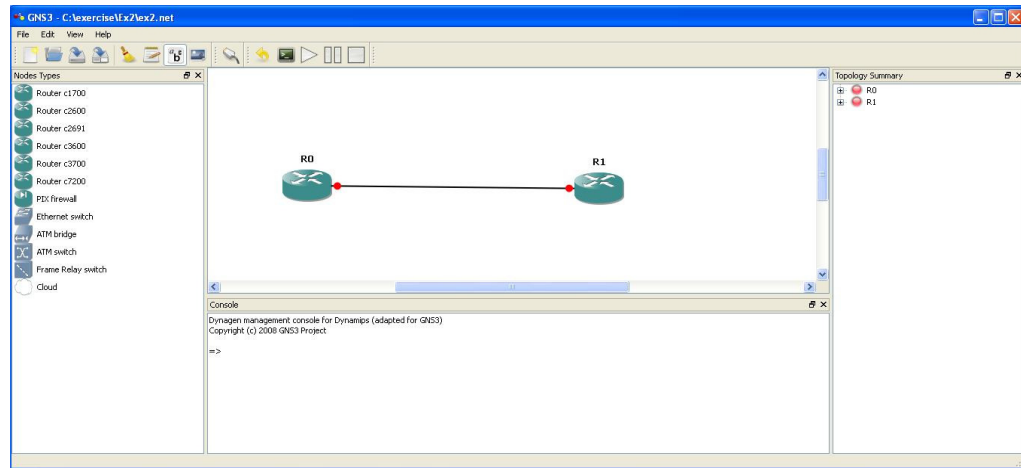


Figure 4-8 Challenge 2 Network Topology

Two 2600 routers were dragged from the left of the screen to the top middle screen and positioned as shown. A fast Ethernet connection between Fa0/0 on R0 and FA0/0 on R1 was then configured.

Generate running-config Files

The next step was to create the running-config files for R0 and R1, and this was accomplished by manually configuring each router. For this challenge it was required for the following elements to be configured.

- R0 hostname set to Paris
- R1 hostname set to LA
- Configure R0 Fastethernet0/0 with IP address 172.45.0.1 and subnet mask 255.255.0.0
- Configure R1 Fastethernet0/0 with IP address 172.45.0.2 and subnet mask 255.255.0.0
- Enabling the link between R0 and R1.

On each router after the elements have been configured the running-config will be copied to the startup-config, and then exported from the routers by the execution of the *export/ all* command in the hypervisor console window. This action is shown in figure 4-9.

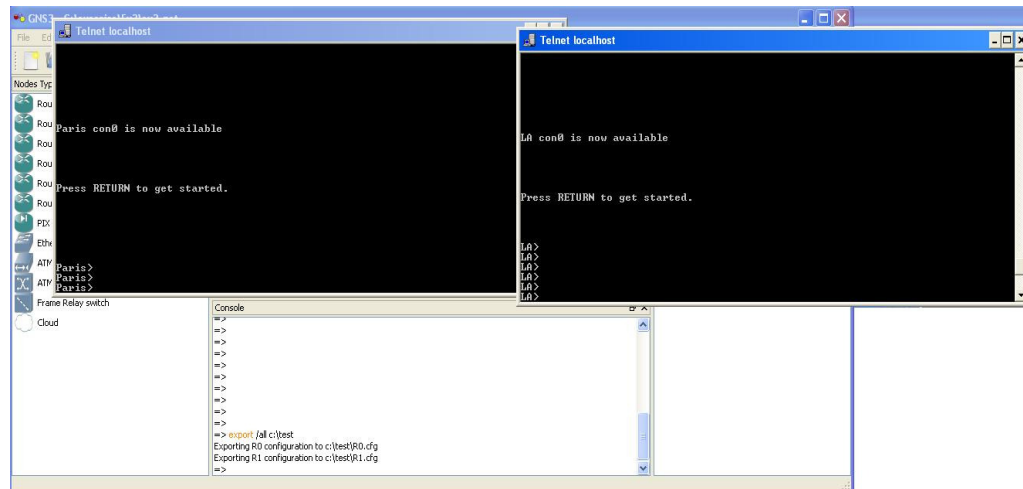


Figure 4-9 Exporting Challenge2 Configs

Once the startup-config was exported the next step is to configure the router with the location of the file, so that it could be loaded each time the router starts and this is shown in Figure 4-10.

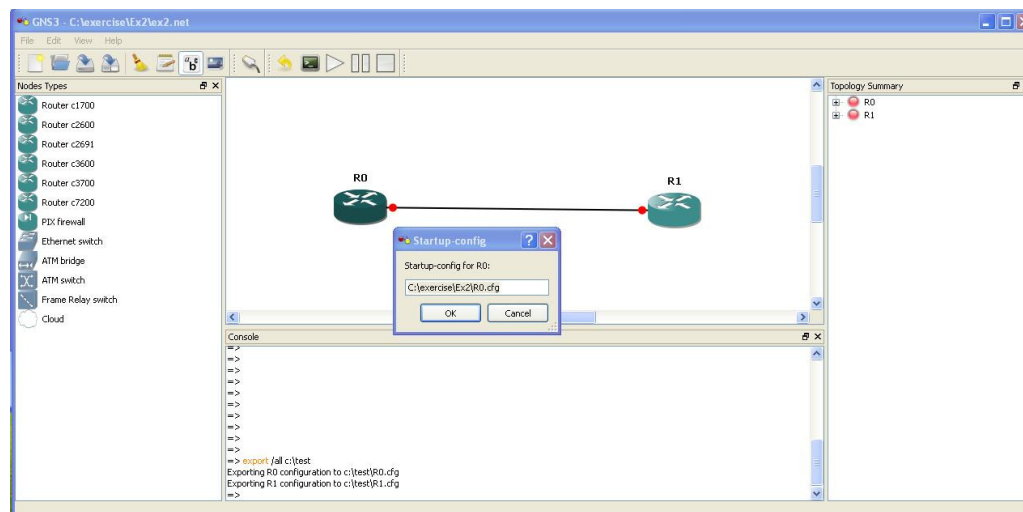


Figure 4-10 Challenge 2 startup-config files

Once all this information has been generated the final step is to save the topology file created using GNS3. See Appendix A,B and C for the generated topology and startup-config files.

Challenge Generator

Now that the topology and startup-config files have been generated the next step is to write the code that will generate the dynamic variables for the challenge. The code will be created using Microsoft Visual Studio 2005 and the C# programming language .As mention before, generating dynamic variables for the student to configure helps them to maintain their interest during the challenge, and also insures that they are thinking about the commands that they are entering instead of just remembering the commands from the previous time they attempted the challenge.

Each challenge will be contained in its own class, because this allows a challenge to have its own unique methods and not be reliant on a standard way of doing things.

To start with, there is a class called the router that is contained in the challenge class and this is used to contain all the relevant dynamic data for the router.

```
1. class router
2. {
3.     private string[] hostArray = new string[6] { "Perth", "Dundee",
        "Edinburgh", "London", "Paris", "Amsterdam" };
4.     private string[] subnetMasks = new string[7] { "0", "128", "192",
        "224", "240", "248", "252" };
5.     public loopback myLoop = new loopback();
6.     string hostName;
7.     string ipAddress;
8.     string ipNextAddress;
9.     string ipSubnetMask;
10.    string routerName;
11.    int eigrpAS;
12.    int bandwidth;
13.    int delay;
14.    private Random fixRand = new Random();
```

Code Snippet 4-1 Router Class Definition

Code snippet 4-1 shows the initial setup of the router class. Line 3 shows the possible hostnames that the router can be configured with, and Line 4 shows the subnet masks that can be configured on the links connecting the routers. As can be seen by Line 5 another class is used to store information about possible loopback adapters on the router. The loopback class will be discussed later. Finally Line 14 is used to seed the random number generator so that each time the challenge is run different data will be chosen.

There are two public constructor members for the router class. The first public constructor is used to generate the data for the first router in the network topology. Code snippet 4-2 shows the first few lines of the router constructor.

```
15. public router(string myName)
16. {
17.     routerName = myName;
18.     // choose hostname
19.     hostName = hostArray[fixRand.Next(0,2)];
20.     eigrpAS = fixRand.Next(1, 65535);
21.     //choose ip address and subnet
22.     int ipClass = fixRand.Next(1, 3);
23.     string mySubnetMask = subnetMasks[fixRand.Next(0, 6)];
24.     string tempIpAddress;
25.     delay = fixRand.Next(1, 1000);
26.     bandwidth = fixRand.Next(1, 50000);
```

Code Snippet 4-2 First Router Class Constructor

Line 15 shows the first public constructor with a variable passed in. This variable is used on Line 17 to name the router such that it matches up with the names displayed on the GNS3 graphical interface (E.G R0 and R1).

Line 19 is used to give a router a hostname. As can be seen only the first three names will be used from Line 3. The reason that this is done is to insure that two routers will not have the same hostname. The second constructor that will be discussed later will be using the next three hostnames.

The EIGRP autonomous system number is chosen on Line 20 and on Line 22 the class of IP address is chosen where class A is number 1, class B is number 2 and Class C is number 3. The subnet mask is chosen on Line 23 from the available subnets that were defined on Line 4 earlier. Finally Line 25 and 26 is used to generate the delay and bandwidth values for use with EIGRP redistribution.

The final step in the router construction is to generate the class A, B or C IP address that was specified by Line 22. The code for generating a class A IP address is shown in code snippet 4-3.

```
27. if (ipClass == 1) //class A
28. {
29. tempIpAddress = fixRand.Next(1, 126).ToString();
30. if (mySubnetMask == "0")
31. {
32. ipAddress = tempIpAddress + ".0.0.1";
33. ipNextAddress = tempIpAddress + ".0.0.2";
34. }
35. else
36. {
37. ipAddress = tempIpAddress + "." + (256 - Convert.ToInt32(mySubnetMask)) +
    ".0.1";
38. ipNextAddress = tempIpAddress + "." + (256 - Convert.ToInt32(mySubnetMask)) +
    ".0.2";
39. }
40. ipSubnetMask = "255." + mySubnetMask + ".0.0";
41. }
```

Code Snippet 4-3 Class A IP Address

The class A address that is going to be used is generated on Line 29 where a random number between 1 and 126 is used to generate a valid address. On Line 30 if the subnet mask selected by Line 23 is 255.0.0.0, then the first valid address is defined by Line 32 and the next valid IP address is defined on Line 33. The next valid IP address is required for when two routers are connected to each other.

The first valid IP address for a given subnet that is not 255.0.0.0 is calculated on Lines 37 and on Line 38 the next valid IP address is calculated so that it can be used on a connecting router. Finally the subnet mask is calculated on Line 40.

The logic for calculating Class B and C addresses are the same and so to keep the discussion of the code to a minimum they are not included in this discussion. See Appendix D for the full code listing.

The second router constructor is not as complex as the first constructor as it is used to configure routers attached to the first router. As a result the IP addresses used on the link have already been configured, as has also the EIGRP autonomous number, EIGRP delay, EIGRP bandwidth and IP subnet mask. The code snippet below shows the second constructor code

```
42. public router(router connectRouter, string myName)
43. {
44. eigrpAS = connectRouter.getEigrp();
45. routerName = myName;
46. // choose hostname
47. hostName = hostArray[fixRand.Next(3, 5)];
48. ipAddress = connectRouter.getNextIpAddress();
49. ipSubnetMask = connectRouter.getIpSubnetMask();
50. bandwidth = connectRouter.getBand();
51. delay = connectRouter.getDelay();
52. }
```

Code Snippet 4-4 Router Class Second Constructor

As can be seen from Line 42 the connecting router object and the name of the router to be created are passed into the constructor. By passing in the connecting router instance it will allow it to be queried for important information, as seen from Lines 44, 48 – 51. Line 45 gives the router a name that matches with that used in the GNS3 graphical representation. The only new piece of data that has to be generated is that of the hostname of the router and this is carried out on Line 47. Note that the hostname selected comes from the second half of the array defined on Line 3 and thus there is no chance of a duplicate hostname being used between the connecting routers.

As has been seen in the definition of the second router constructor there are various member functions that can be used to query the router for information. For completeness they are shown in code snippet 4-5.

```
53. public string getHostname() { return hostName; }
54. public string getIp() { return ipAddress; }
55. public string getIpSubnetMask() { return ipSubnetMask; }
56. public string getNextIpAddress() { return ipNextAddress; }
57. public string getRouterId() { return routerName; }
58. public int getEigrp() { return eigrpAS; }
59. public int getBand() { return bandwidth; }
60. public int getDelay() { return delay; }
```

Code Snippet 4-5 Router Class Query Members

The final piece of code that has to be examined is that for the loopback class. Instead of including everything in the router class it was decided to do the loopback adapters as a separate class, but contained in the router class.

The initial definition of the loopback class is shown in code snippet 4-6.

```
1. class loopback
2. {
3.     private ArrayList myLoopIp = new ArrayList();
4.     private ArrayList myLoopSubnet = new ArrayList();
5.     string subnet = "255.255.255.0";
```

Code Snippet 4-6 Loopback Class Definition

Two arraylists on Lines 3 and 4 are going to be used to store the IP address and subnet mask respectively. It was decided to make all of the loopback addresses generated to be a class C address and the subnet mask for this is defined on Line 5.

```

1. public loopback()
2. {
3.     string ipLoopback;
4.     string subnetAddress;
5.     Thread.Sleep(1);
6.     Random fixRand = new Random();
7.     subnetAddress = fixRand.Next(192, 223).ToString() + "." +
        fixRand.Next(0, 254).ToString() + ".";
8.     int tempOctet = fixRand.Next(0, 214);
9.     for (int i=0; i < 4; i++)
10.    {
11.        ipLoopback = subnetAddress + tempOctet.ToString() + ".1";
12.        myLoopIp.Add(ipLoopback);
13.        myLoopSubnet.Add((subnetAddress + tempOctet.ToString() + ".0"));
14.        tempOctet = tempOctet + 10;
15.    }
16. }

```

Code Snippet 4-7 Loopback Constructor Class

The constructor for the loopback class is shown in code snippet 4-7. One of the problems when creating the loopback class was that when other router instances were creating their own loopback classes, the IP numbers being generated were the same. To overcome this problem a short delay of 1ms was introduced on Line 4. On Line 7 the first two octets of a class C address are being generated while on Line 8 the third octet is generated. The third octet is restricted to a maximum value of 214 because it was decided to generate four loopback addresses with a 10-subnet gap between them. This will give a maximum IP address of 254. If the third octet was not capped in such a manner then there is the danger that invalid subnet numbers would be generated.

The final piece of code for the loopback constructor is used to generate the four loopback addresses. This can be seen on Lines 9 through 14. The first valid IP address is calculated on Line 10 and added to the relevant arraylist on Line 11. The valid subnet for the loopback is calculated on Line 12 and stored in the relevant arraylist.

There are other member functions for the loopback class that deal with retrieving information from the class, but these functions are pretty much self-explanatory and are not included in this discussion. See Appendix D for the full listing of the loopback class.

This concludes the discussion of the generation of the data for Challenge 2 and the next area that will be examined is that of generating the challenge data.

4.2.3 Generate Challenge Data

The next class that will be examined in detail is that of the Challenge 2 class, since this is used to generate the data for the challenge. Only the code that is of interest will be discussed, there are some functions that are missed out only because they perform a standard function that is self-explanatory. Code snippet 4-8 shows the initial declaration of the variables in the Challenge2 class.

```

1. class challenge2
2. {
3.     private ArrayList myStartup = new ArrayList();
4.     private static router myRouter0 = new router("R0");
5.     private static router myRouter1 = new router(myRouter0, "R1");

```

Code Snippet 4-8 Challenge Initial Definition

The first step is to define an arraylist to store the startup-config that is going to be generated by the user, and this is accomplished on Line 3. The reason that an arraylist is being used is because the arraylist size is not fixed, unlike that of a traditional array, and since there is no way to determine the size of the startup-config that is going to be generated by the student, an Arraylist is the best option to use to store it.

Line 4 sets up the first router class with a name of R0 and on Line 5 the second router class is set up using the first router as the connecting router, and also giving the second router a name of R1. These are the steps that have to be followed to generate the data for all challenges.

4.2.4 Display Challenge Specification

The next process is to display the challenge specification to the user and this is accomplished by calling a function called `displaySpec`, which itself is called from the function `process`. The function `process` is responsible for controlling the running order of the challenge and is shown in code snippet 4-9.

```
1. public void process()  
2. {  
3.     displaySpec(myRouter0);  
4.     displaySpec(myRouter1);  
5.     Console.ReadLine();
```

Code Snippet 4-9 Process Member Function

For each router certain tasks have to be performed and that is why on Lines 3 and 4 the `displaySpec` function is called with different instances of the router class. Line 5 is used to pause the screen so that the student can read the specification and also perform the challenge on the emulator before the marking process starts. The actual `displaySpec` function is shown in code-snippet 4-10.

```
1. public void displaySpec(router myRouter)  
2. {  
3.     Console.WriteLine("Configure " + myRouter.getRouterId());  
4.     Console.WriteLine("The link between R0 and R1 has already been  
       preconfigured");  
5.     Console.WriteLine("1. Set up 4 loopback addresses as follows");  
6.     myRouter.myLoop.displayLoop();  
7.     Console.WriteLine("2. Set up EIGRP routing with AS " +  
       myRouter.getEigrp());  
8.     Console.WriteLine("3. Advertise the loopback addresses using EIGRP");  
9. }
```

Code Snippet 4-10 displaySpec Member Function

As can be seen from the code only the data that is contained within the router class is displayed.

4.2.5 Mark Challenge / Provide Feedback

The final two processes in the design that are going to be looked at are Mark Challenge and Provide Feedback. In the original design both of these processes are separate, but it was decided to incorporate them as one process to speed up the development process.

The first step in the process is to obtain the work that the student has done on the emulated routers. In the time available no way was discovered of how to hook into the routers to determine what the student was typing in real-time. One proposed way of monitoring the routers was to use SNMP but this would mean that the routers would have to be preconfigured and this thus means that the student would not be working with virgin routers.

The way that this problem was resolved was the discovery that dynamips was storing the NVRAM as a file. This meant that the NVRAM file could be opened to extract the startup-config. Unfortunately pre-processing had to be performed to extract the relevant information, as the content of the file was not in a text readable format. The entirety of the process function is shown in code snippet

```
1. public void process()
2. {
3.     displaySpec(myRouter0);
4.     displaySpec(myRouter1);
5.     Console.ReadLine();
6.     processConfigs("C:\\Documents and Settings\\dave\\Local
   Settings\\Temp\\c2600_R0_nvram", "C:\\exercise\\Ex2\\R0.txt");
7.     displayRun("C:\\exercise\\Ex2\\R0.txt");
8.     int index = myStartup.IndexOf("version 12.1");
9.     if (index < 0)
10.    {
11.        Console.WriteLine("Oops, user has forgotten to copy running config to
   startup config for " + myRouter0.GetHostname());
12.    }
13.    else
14.    {
15.        checkConfigs(myRouter0);
16.    }
17.    myStartup.Clear();
```

Code Snippet 4-11 Full Process Member Function

Line 6 does the processing of the nvram to obtain the startup-config. There are two arguments that are passed into the processConfigs, the first argument is the location of the nvram file and the second argument is the location of where the processed startup-config should be stored. On Line 7 the displayRun function is used to open the processed startup-config and to store it in the myStartup arraylist. Lines 8 through 12 are a check to verify that the nvram file that was processed contains a valid startup-config file. Once the startup-config has been verified to be valid the actually marking and feedback is started by Line 15.

After the startup-config file has been marked the process is repeated for the second router. On Line 17 the startup-config file that was used for the first router has to be deleted so that it doesn't contaminate the second router's file. The process for checking the second router is exactly the same as the first and as result it will not be repeated.

The specification for the first router on Challenge 2 was as follows: -

1. Configure 4 loopback addresses using the stated parameters
2. Configure EIGRP using the stated parameters
3. Advertise the directly connected links between R0 and R1 into EIGRP
4. Advertise the loopback addresses into EIGRP

5. Copy the running-config to the startup-config

The verification of specification 5 has already been verified in the discussion of the previous code snippet. The first task is that of the verifying that the loopback addresses have been configured properly and this is shown in code snippet 4-12.

```

1. public void checkConfigs(router myRouter)
2. {
3.     Console.WriteLine("Checking      Configuration      for      "      +
myRouter.getRouterId());
4.     for (int i =1; i < 5; i++)
5.     {
6.         Int index = myStartup.IndexOf("interface Loopback" + i.ToString());
7.         if (index < 0)
8.         {
9.             Console.WriteLine("");
10.            Console.WriteLine("Loopback has not been set as specified");
11.            Console.WriteLine("The correct ISO command is :- interface loopback" +
i.ToString());
12.        }
13.    else
14.        Console.WriteLine("Loopback" + i.ToString() + " has been specified
correctly");

```

Code Snippet 4-12 checkConfigs Member Function

There are four loopback interfaces to check and a loop on Line 4 is going to be used to process all the interfaces have been defined. The first thing to check is that the loopback interfaces have been implemented and this is done on Line 6, which searches the myStartup arraylist for the definition of the loopback interfaces. If the interfaces have not been defined then the student is informed of this and the correct command for defining the interfaces are displayed. This functionality is implemented on Lines 7 through to 11. If the loopback interface has been defined properly the Line number of where it was found is stored in the *index* variable, and a message is displayed saying that the interface has been specified correctly.

Since it has been determined that the loopback interface have been defined properly the next step is to determine if an IP address has been specified on the interface and this is shown in code snippet 4-13.

```

15. if ( myStartup[index+1].ToString() == "no ip address")
16. {
17.     Console.WriteLine("Loopback" + i.ToString() + "no ip address has been
specified for Loopback" + i.ToString());
18.     Console.WriteLine("The correct IOS command is :- ip address " +
myRouter.myLoop.getLoopIp(i-1) + " " + myRouter.myLoop.getSubnet());
19. }
20. else

```

Code Snippet 4-13 Verifying Loopback Interfaces

The verification that an IP address has been specified is done on Line 15 where the next line, as found by Line 6, in the myStartup arraylist is queried to determine if there is an IP address or not. If no IP address has been specified then a message is displayed to the student, Line 17, stating such and is then followed by the correct commands to use, Line 18.

Once it has been verified that the loopback has an IP address, the next step is to determine if the IP addresses that was specified to the student in the specification has been used. This is accomplished in code snippet 4-14.

```
21. if ( myStartup[index+1].ToString() == (" ip address " +  
    myRouter.myLoop.getLoopIp(i-1) + " " + myRouter.myLoop.getSubnet()))  
22. {  
23. Console.WriteLine("Ip address for Loopback" + i.ToString() + "is  
    correct");  
24. }  
25. else  
26. {  
27. Console.WriteLine("Ip address for Loopback" + i.ToString() + "is  
    incorrect");  
28. Console.WriteLine("The IOS command that was used :- " +  
    myStartup[index+1].ToString());  
29. Console.WriteLine("The correct IOS command is :- ip address " +  
    myRouter.myLoop.getLoopIp(i-1) + " " + myRouter.myLoop.getSubnet());  
30. }
```

Code Snippet 4-14 Verifying IP Address

The first step on Line 21 is to extract the relevant loopback IP address and subnet mask from the loopback class, and then build the command that was supposed to be used. The second step on Line 21 is to compare the built command from step 1 with the actual command that was used for specifying the IP address in the myStartup arraylist. If the correct command was used then the student is told so on Line 23, however if the incorrect command was used then Line 27 and Line 28 tell the student such, and Line 29 shows the student how the IP address for the loopback interface should have been configured.

The process described above is used to verify that all four loopback interfaces have been specified correctly. The next task that had to be done from the challenge specification was to configure EIGRP on the router. The code snippet below shows how this was done.

```
31. int    eigrpIndex    =    myStartup.IndexOf("router    eigrp    " +  
    myRouter.getEigrp());  
32. if (eigrpIndex < 0)  
33. {  
34. Console.WriteLine("Eigrp has not been configured properly");  
35. Console.WriteLine("The correct IOS command is :- router eigrp " +  
    myRouter.getEigrp());  
36. }
```

Code Snippet 4-15 EIGRP Verification

Line 31 is used to check if the arraylist myStartup contains the *router eigrp* command along with the correct autonomous system number. If the EIGRP has not been properly configured then the student will be told so, Line 31, and the correct command to perform the operation will be displayed, Line 35.

After it has been verified that EIGRP was properly configured, the next step from the challenge specification was to advertise the directly connected interfaces into EIGRP. The following code snippet shows how this was accomplished.

```
1. int index2 = myStartup.IndexOf(" network 172.45.0.0");
2. if (index2 < 0)
3. {
4. Console.WriteLine("Directly connected links between R0 and R1 have not
   been advertised via EIGRP");
5. Console.WriteLine("The correct IOS command is:- network 172.45.0.0");
6. }
7. else
8. {
9. Console.WriteLine("The directly connected links between R0 and R1 have
   been correctly advertised via EIGRP");
10. eigrpIndex++;
11. }
```

Code Snippet 4-16 Verifying Connected Interfaces

Since the directly connected links are in the 172.45.0.0/16 class B address, Line 1 determines if this address has been advertised into EIGRP. If this address has not been advertised then a message is displayed to the student, Line 4, and then the correct command line that they should have used is displayed, Line 5.

The final part of the specification that had to be accomplished was that of advertising the loopback interfaces into EIGRP. This following code snippet shows how this was accomplished.

```
1. for (int i = 1; i < 5; i++)
2. {
3. index2          =          myStartup.IndexOf("          network          " +
   myRouter.myLoop.getSubnetIp(i-1));
4. if (index2 < 0)
5. {
6. Console.WriteLine("Loopback" + i.ToString() + "has not been advertised
   via EIGRP properly");
7. Console.WriteLine("The          current          IOS          command          is          :-          " +
   myStartup[eigrpIndex + i]);
8. Console.WriteLine("The correct IOS command to use is network " +
   myRouter.myLoop.getSubnetIp(i-1));
9. }
10. else
11. {
12. Console.WriteLine("Loopback" + i.ToString() + " has been configured
   correctly via EIGRP");
13. }
14. }
```

Code Snippet 4-17 Advertising Loopback Interfaces into EIGRP

Since there are four loopback interfaces a loop was used to process them, Line 1. Line 3 is used to determine if the relevant loopback interface has been advertised using the correct subnet. If the subnet has not been advertised correctly then the student is informed, Line 6, and the correct commands that they should have used are displayed, Lines 7 and 8.

This concludes the section on checking the Challenge 2 router configuration. The next section will examine the steps taken to implement the footprint evaluation.

4.3 Footprint Evaluation Implementation

This section will describe the implementation of the experimentation that was carried out to verify the performance of the Dynamips emulator on the following three operating system platforms: -

- Windows XP
- Windows XP running in Emulation mode using VMWARE Fusion on a Macintosh computer.
- Windows Vista Business

Testing was performed on a Mac book Pro running bootcamp, this is a program that allows Windows XP or Vista to be run on Apple Mac hardware. The hardware specification of the Mac book for both Windows XP and Vista is shown in table 4-1:-

Component	Capacity
Display	15.4-inch (diagonal) TFT LED backlit display
Processor	2.4GHz Intel Core 2 Duo processor
Memory	2GB (two SO-DIMMs) of PC2-5300 (667MHz) DDR2
Network Card	Built-in 10/100/1000BASE-T Gigabit Ethernet (RJ-45 connector)
Hard Drive	120GB 5400-rpm Serial ATA
Graphics Card	NVIDIA GeForce 8600M GT with 256MB of GDDR3 SDRAM

Table 4-1 Native Machine Specification

When XP is run in emulation on a MAC then there are some changes to the hardware specification, most notably that the emulation is only using one core of the processor and there is a reduction in the amount of memory and disk space used. When using emulation the hardware specification of the guest operating system is shown in Table 4-2. The software listed in Table 4-3 was used to build and test the solution.

Component	Capacity
Display	15.4-inch (diagonal) TFT LED backlit display
Processor	2.4GHz Intel Core 2 Duo processor (Only one core being used)
Memory	512MB
Network Card	Built-in 10/100/1000BASE-T Gigabit Ethernet (RJ-45 connector)
Hard Drive	21GB partition 5400-rpm Serial ATA
Graphics Card	VMware SVGA II

Table 4-2 Emulation Machine Specification

Software	Version
Windows XP Professional	SP3
Windows Vista Business	SP1
Visual Studio NET 2003	SP1
Boot Camp	2
VMWARE Fusion	1.1.1

Table 4-3 Software Versions

To monitor the performance of the operating system when the proposed framework is running, a window utility called performance monitor will be used. This utility is capable of automatically monitoring and capturing performance counters within windows to a log file. The log file generated can be specified to be in various formats and for this experiment a CSV file format will be used since it makes processing this information by other programs simpler. It was decided to monitor the performance counters listed in Table 4-4.

Performance Counter
Physical Disk (instance)\Disk Transfers/sec counter
Memory\Pages/sec
\Processor(_Total)\% Processor Time

Table 4-4 Performance Counters

For each operating system the experimentation was run for an hour and repeated 10 times. A program was written and used to start and stop the logging, and also to process the results from the performance log file into a CSV file that could be imported into Excel for further processing. See Appendix E for the program listing of the program.

The screenshot in Figure 4-11 shows an example of the experiment running along with the program used to control the experiment: -

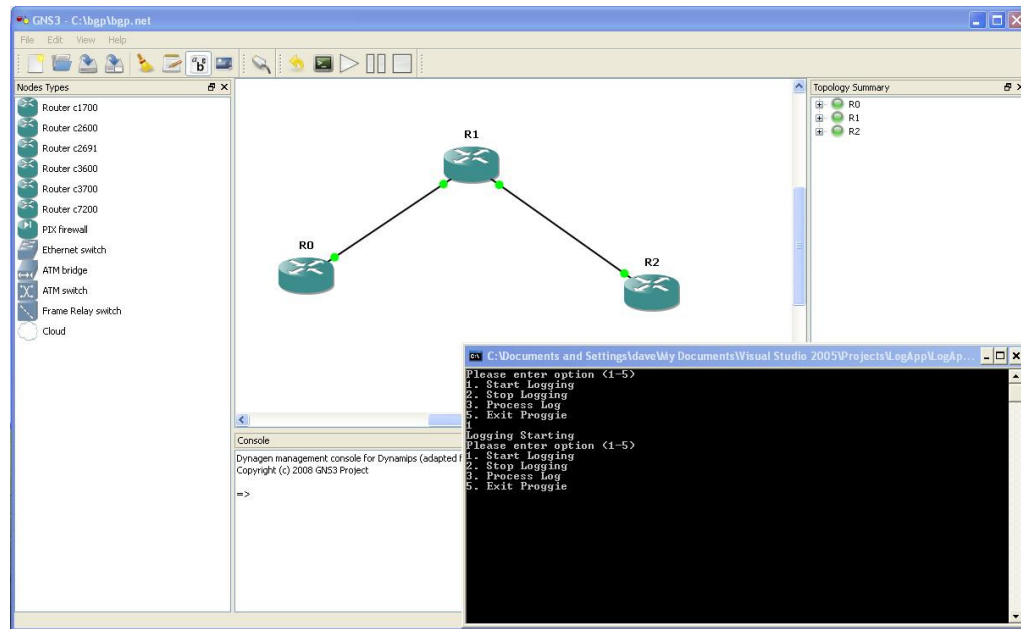


Figure 4-11 Screenshot of Experiment Running

4.4 Questionnaire Implementation

Students and staff who were involved with the MSc advance-networking course were asked to evaluate the overall solution, and provide feedback in the form of a questionnaire.

The volunteers were given an orientation on the overall usage of the solution and then asked to complete the three challenges. After the volunteers had finished they were then presented with the questionnaire to complete.

4.5 Conclusions

This chapter has shown the implementation of the framework, footprint and questionnaire. The Framework implementation has described how the generation of the challenges were done using the Dynamips and GNS3 emulation technology. This was followed by a discussion of how the framework was implemented using the Microsoft Visual Studio C# programming language, and even though only certain processes were implemented it was still possible to obtain a fully functional concept for testing.

The footprint evaluation detailed the machine specifications, software and performance counters that were going to be used to monitor the overall performance of the proposed framework. The questionnaire implementation described the process that was going to be followed in the selection of the volunteers and how they were going to be trained up in its use.

5 Evaluation

5.1 Introduction

This chapter will review the results obtained by the performance evaluation performed on the overall solution and highlight any areas of concern. This will then be followed by an explanation of further testing that was carried out to verify the performance issues of running XP in emulation mode, and the final part of this chapter will review the results of the questionnaire.

5.2 Performance Evaluation

From the ten experiments that were run on the specified operating systems, the average range of results for each counter are presented in tables 5-1, 5-2 and 5-3.

Operating System	Average Disk Transfers (/second)
Windows XP	2.1 – 2.5
Windows Vista Business	1.5 – 1.7
Windows XP Emulation	2.7 – 3.1

Table 5-1 Average Disk Transfers/second

Operating System	Average Memory\Pages (/second)
Windows XP	0 – 0.08
Windows Vista Business	0 – 0.01
Windows XP Emulation	0 – 0.02

Table 5-2 Average Memory\Pages /second

Operating System	Average Processor Utilisation Range (%)
Windows XP	1.6 – 1.7
Windows Vista Business	0.9 – 1.2
Windows XP Emulation	49.8 – 50.9

Table 5-3 Average Processor Utilisation

To interpret the result table 5-4 shows the recommended values that indicate acceptable performance for the performance counters used [49-51]: -

Performance Counter	Normal Operation
Physical Disk (instance)\Disk Transfers/sec counter	Anything below 25 disk I/Os indicates good performance
Memory\Pages/sec	Anything below 50 pages/sec is considered good performance
\Processor(_Total)\% Processor Time	A value below 70% indicates good solid performance

Table 5-4 Optimal Values for Performance Counters

The interpreted results based upon the table above show that all of the operating systems are working within acceptable parameters. One of the results of interest is that the Windows Vista operating system came out with the lowest readings in all three counters. The reason for these low readings may point to the fact that Vista was designed for use with multi-core processors, and also have more efficient algorithms for scheduling Memory, Disk and Processor over that of the XP operating system.

One area of concern is the high processor utilisation (50%) when XP is running in emulation. Even though the processor utilisation is at 50% and within the acceptable performance guidelines, it poses the question of how many routers would it take to take the processor utilisation to over 70%. To answer this question further experimentation was performed, by increasing the number of routers running BGP, starting with two and then three, four, five and six. The average processor utilisation results from this experimentation are shown in table 5-5 and a graph showing the plotted results are shown in Figure 5-1.

Number of Routers	Average Processor Utilisation Range (%)
2	36.1
3	43.04
4	49.61
5	52.44
6	55.7

Table 5-5 Average Processor Utilisation

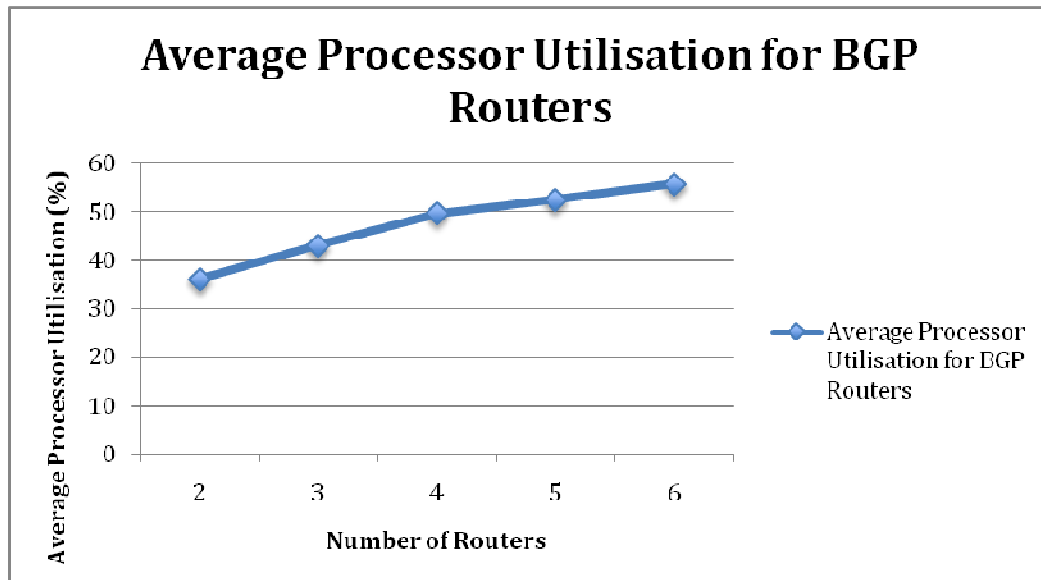


Figure 5-1 Chart of Average Processor Utilisation

From the chart in Figure 5-1 it can be seen that the increase in processor utilisation is almost linear when going from 2 to 3 to 4 routers, however the increase in processor utilisation is no longer linear when 5 and 6 routers are used.

The change suggests that the Dynamips emulator has some optimisation functions that can allow more routers running but without over burdening the processor. This would suggest that if more routers were added to the challenges then the emulated XP would be able to handle the increase.

The main problem with the way that the experiments were implemented was that the readings taken corresponded to the whole of the operating system, so this means that if another service or process is using the resources then this is reflected in the relevant counter. This may explain why when the experiment was being run for the second time, that average processor utilisation for the 3 routers was 43.04 (table 5-5) and this is significantly below the average range in table 5-3 (49.8 – 50.9) when the experiment was first being run.

A better design for the experiment would have been to only monitor the performance counters for the relevant processes so that no outside interference affects the results.

5.3 Questionnaire Evaluation

The number of responses from the volunteers for each question is shown in table 5-5.

Num.	Question	SA	A	N	D	SD
1	The level of feedback was adequate for the student to learn from the mistakes made?	2	6	2		
2	The program was easy to use and set up?		2	2	6	
3	The specification of the challenges was stated in a clear manner?	5	3	2		
4	Not having to start from scratch in each challenge saved time and allows the challenge to be completed quickly?	8	2			
5	Using a fully functional IOS is better than the restricted IOS used in simulators (e.g. full use of the back and forward arrow keys for the command history)?	8	2			
6	The challenges maintain my interest throughout?		6	4		

Table 5-6 Questionnaire Responses

One of the most positive reactions obtained from the volunteers was that a full command line IOS interface was present and useable and this is reflected by the answers to Question 5. One thing to consider is that the volunteers were experienced in networking and would fully appreciate a complete interface, but if a novice was being used for the evaluation, then they may not have the experience to fully realise the importance of this fact.

Examining the results shows questions 4 and 6 have a very positive response and suggests that using fading work examples for the questions to reduce the cognitive load were well received [5, 41].

Question 1 asking about the feedback received positive feedback, but it was not a overall positive response and this may be attributed to the fact that a text based prompt was being used, and also that the student was being shown a one line command and not a step by step process for solving the problem.

Another area that got a strong response was that the specification of the challenges was stated in a clear manner, question 3. This was seen to be a bit of a surprise result, since it was expected that the volunteers would disagree or strongly disagree with this question, mainly because a text-based and not a graphical user interface was being used to present the specification.

The only question that received a negative response was question 2, and this was expected since the steps that have to be taken to set up the challenge for the user

is quite cumbersome. This shows that an automated process using one key press is required, so that everything is ready for the student to attempt the question.

5.4 Conclusions

From the performance evaluation it has been shown that the performance of the proposed framework works on a variety of platforms, and within acceptable performance parameters. The only real concern was the high processor utilisation when XP emulation was being used, and the possibility of its impact when using more complex network topologies. To answer this concern, further experimentations were carried out with ever increasing number of routers, and it was proven that the processor utilisation was nearly linear going from 2 to 3 to 4 routers, but adding 5 and 6 routers showed that the increase was no longer linear as might have been expected.

The questionnaire carried out proved that using the fading worked examples and a fully working command line IOS was very well received by the students. The only real concern was the number of steps that had to be taken in the setup of the challenges and in a future release it should be investigated how to make this process less complex.

6 Conclusions

6.1 Introduction

This chapter is split into three sections; the first section will deal with the main conclusions of the thesis, the second section will examine the aim and objectives and show how they have been met, along with any problems encountered and suggestions for improvements. The third section will discuss future improvements.

6.2 Conclusions

6.2.1 Performance Issues

One of the main findings was the burden that was placed on the processor when windows XP was running in emulation. Windows XP and VISTA, when running natively, do not show a huge processing demand when using the proposed framework. However the main concern is the processing demand when XP is running in emulation, since it is always near 50% processor utilisation, and this may cause performance issues if XP is required to perform other tasks, or more complex network topologies are being used in challenges.

The reasons as to why the processor utilisation is so high when XP is running in emulation could be because of a combination of the restricted amount of RAM that is available to the operating system, and also because XP is running on only a single core of a dual core system. When XP and VISTA are ran natively then they both have access to the full specification of the computer, while with the emulation running the processor has also got to cope with the demands of the host operating system.

It also poses the question of what would happen if the experiment were repeated on a single core system, would both the host and guest operating system suffer severe performance loss.

6.2.2 Bloom Model

There is some discussion in the academic community about the original Bloom taxonomy and if it is applicable to be used for computer science [3]. The debate seems to be mainly focused on teaching programming, with a proposed new bloom level called application [3], and very little discussion on networking.

With programming there are many abstract concepts that can be used in many ways, to solve many different problems. However the concepts with configuring networking are generally used to solve one problem, and as a result there is a fixed way of solving certain problems. During the ranking of the challenges it was found that the Bloom model was suited for this task, and it may suggest that Bloom's taxonomy is ideal for use in teaching networking.

One of the problems with using Bloom is that it is very subjective [3, 43], and it has been known for two different lecturer's to come to two different bloom ratings for the same material. To verify the Bloom ratings for the challenges it would be recommended to review them with a member of academic staff, who has got experience of using Bloom.

6.3 Aim and Objectives

The overall aim of this project is to develop and evaluate an educational framework that could be used to enhance education learning in universities for network courses. To achieve the overall aim for this project the following objectives had to be met:

1. Investigate the current learning taxonomies that are used in an educational environment and investigate how these can be applied in an emulated network environment to provide an enhanced learning tool for students.
2. Integrate a network emulation module that will automatically generate a dynamic network scenario for the user to configure and then automatically grade the user based upon how they solved the problem.
3. Monitor the performance of the overall solution to verify that the solution developed can be run on the user's machines without any adverse affects that would spoil the user experience.

6.3.1 Objective 1

The first objective was met by reviewing the current literature in the areas of learning taxonomies. From the review it was discovered that there were a wide range of taxonomies in use for different purposes, and only the most common ones were reviewed. From the review it was deemed that Bloom [3, 4, 43] and revised Bloom [3] are used to help specify module descriptors and exam questions, and the SOLO taxonomy [3, 4] was used to gauge how well the student had answered the question, with the student gaining more marks for being judged to have answered the question higher up the scale and less marks for being lower on the scale.

For the generation of questions it was determined that it would be best to use the Bloom model, and for marking the students answers the SOLO taxonomy would be used. It was decided to use the Bloom taxonomy instead of revised Bloom to generate the questions since it is the most widely used taxonomy from the papers reviewed. Revised Bloom has only been recently introduced and is still not in wide use by Universities.

Unfortunately In the final implementation, the design goal of using SOLO to mark the student's answers was not possible due to limitations of the framework that was chosen, and this will be discussed in the later sections.

The next stage of meeting the first objective was to investigate why emulation should be used instead of simulation, and what advantages can be gained over that of using traditional labs for networking.

From the literature review it was discovered that one of the major advantages for using emulation instead of simulation was that the student had a fully implemented operating system at their disposal [2], instead of being restricted to the commands that have been implemented on the simulation [1]. During the evaluation of the proposed solution it was noted that the majority of students agreed that having a fully implemented operating system was a desirable trait.

One of the major advantages of setting up an emulated lab over a classroom-based lab is that of cost and space. The initial setup cost of building a specialist lab for a course is quite high and sometimes outside financing has to be sought [1]. One of the other major disadvantages of specialist labs is trying to find the space to house them since space in universities can be at a premium.

The advantage of using an emulation lab is that it can be set up anywhere and use up a lot less space than that of a dedicated lab. Another advantage of emulation is that the student could load the emulator on virtually any machine, depending on the machine specification, and thus work from anywhere and not be restricted to one lab in the university [18].

The final stage of meeting the first objective was to investigate how students process information, and determine how best to design challenges so that they are able to learn to the best of their abilities without getting tired and bored.

The literature review revealed there are two types of memory used to process and store information and they are short-term and long-term memory [41]. Long-term memory has near unlimited capacity to store vast amounts of information in the form of schemas while short-term memory, where the learning is done in the form of constructing schemas, has limited capacity, and as a result the processing load in this area of memory needs to be carefully managed to achieve efficient learning and to avoid overloading it.

Reducing the load on short-term memory leads to the cognitive load theory that tries to maximise the students learning, by designing the lectures to lower the cognitive load on the student [5, 41]. While the cognitive load theory addresses how to present the material, it does not address the best way to design exercises for the student that would lower the cognitive load. This issue is addressed by using fading worked examples, and works by presenting the student with a partial solution to a problem that they then have to complete [5]. By presenting the student with a partial solution, they then only have to concentrate on the material required for completing the missing parts to solve the challenge, thus reducing the cognitive load.

6.3.2 Objective 2

The second objective was met by creating a framework for generating and marking network challenges. The framework was comprised of the following elements: -

1. Dynamips Emulator

2. GNS3 Graphical User Interface
3. Challenge Generator

Dynamips Emulator

The first element that was used in the framework was a program called Dynamips, and this is an emulator based upon virtualisation, like running XP on VMWare [9]. Since virtualisation is being used, then this means that the actual IOS firmware can then be loaded onto the hypervisor to give the user a fully functional command line operating system, and thus eliminating one of the biggest complaints of using simulators [2].

Since virtualisation is being used it has to be shown that the three goals of virtualisation (fidelity, performance and safety) have been adhered to [26, 27].

The first goal has been met since the actual Cisco IOS firmware is being used, and this results in a fully operational router that is the same as the actual hardware. The third goal of performance was verified by ending the process of one hypervisor, and by ending it in this manner it proved that there were no ill effects with the other hypervisors. The only goal that was not verified due to time limitations was that of performance.

The virtualisation goal of performance was not able to be proved, but this could be carried out in future work by using a traffic generator to bombard the virtual router with packets, and then calculating how many packets the device was able to process. The same experiment could then be repeated on a real router and the results compared to determine how the performance of the two routers differed.

GNS3 Graphical User Interface

One of the main problems with using the Dynamips emulator is that it is based upon a text interface and the use of it is not intuitive. To overcome this problem it was decided to use a graphical user interface, and to save time and effort a package called GNS3 was used. GNS3 is a GUI bolt on to Dynamips that presents to the user a friendly graphical user interface.

Even though using the GNS3 interface saved time and effort it did introduce a few problems into the design. One of the intentions of the design was to monitor the student in real-time to determine how they were solving certain problems, and then to mark them based upon the SOLO taxonomy [3, 4]. With using GNS3 this design goal was unable to be met, and this illustrates one of the problems of using packages off the shelf that were designed by other people, in that development time is saved but this is offset by loss of functionality that has to be worked around. In a future development it is suggested that a custom GUI be written that would resolve the problems encountered.

Challenge Generator

The third element, called the challenge generator, was created using Microsoft Visual Studio 2005 and the C# programming language. This element created dynamic challenges that changed each time the student attempted them. Each of

the challenges were assigned a difficulty rating according to the Bloom level, with the lowest difficulty question being attempted first and then followed by the next hardest and so on [3].

Each of the questions were designed with the concept of fading worked examples, so that the cognitive load placed on the student would be at a minimal and thus help them to maximise the learning outcomes [5].

The initial design was to use the SOLO taxonomy to mark the answers to the questions, so that if the student answered the question at what is deemed a higher level (e.g. Relational) then they would receive higher marks than if they are deemed to have answered the question at a lower level (e.g. Multistructural) [3, 4].

For example when configuring a Fastethernet interface the normal steps taken are as follows: -

```
Interface fastethernet0/0  
Ip address 192.168.10.1 255.255.255.0  
No shutdown
```

If the student follows all the steps then they would get full marks as the answer is relational according to SOLO, and shows that the student fully understands how to apply the theory to the problem. If the student only configures the first two lines and then goes and does another task before coming back to perform the last line, then the answer according to SOLO is multistructural, a disorganised list with no order. This would show that the student has not grasped the full concept and so should not get full marks for completing the task.

Unfortunately this design goal was unable to be met as it would require real time monitoring of the student, and this was determined not to be possible within the framework chosen and the time available. To resolve this problem the GNS3 Graphical User Interface would have to be replaced with a custom generated GUI that would allow the real-time monitoring of the commands used by the students.

One of the problems with the challenge generator is that the student using it is presented with a text-based interface that is not user friendly, and it also makes it hard to decipher the feedback being presented. To resolve this problem it is suggested that a GUI interface be created so that it would make it easier for the user to use the program. For the feedback a word document should be generated so it is more presentable and can be saved for later study.

To evaluate the overall effectiveness of the solution a questionnaire was designed, and ten volunteers who had been on the Advanced Networking course at Napier University were asked to participate in the study. The overall findings of the questionnaire showed that using the fading worked examples and a fully working command line IOS was very well received by the students. The only real concern shown was the number of steps that had to be taken in the setup of the challenges and in a future release it should be investigated how to make this process less complex.

One potential problem with the questionnaire study was that only people who had experience in the networking field were asked to participate, and because of this experience they may have a different opinion on certain evaluation questions than that of a novice user. For example the experience users indicated that the feedback was adequate, but a novice user may require more detailed feedback to help them understand as to where they went wrong. In future studies it is recommended that a mixture of experienced and novice users be used to achieve a more balanced view.

6.3.3 Objective 3

The third objective was met by using a windows utility called performance monitor to monitor the overall performance of the proposed framework. The testing was carried out on the following three platforms: -

- Windows XP
- Windows XP running in Emulation mode using VMWARE Fusion on a Macintosh computer.
- Windows Vista Business

Using the performance monitor utility the counters monitored are shown in Table 4-4.

Each experiment was performed 10 times on each of the operating systems using a network topology of three routers running BGP. The reason that BGP was used is because it is seen as being the most memory and process intensive routing protocol, but the results for XP and Vista prove that there is very little burden on the disk, processor and memory.

6.4 Future Work

This section will look examine some of the problems with the implementation and go on to detail the future work and improvements that can be made to the project. The following areas will be reviewed, network evaluation, challenge creation, Bloom levels, marking techniques and implementation problems.

6.4.1 Evaluation

The initial outline of the project was to perform a comparison of the network performance of an emulated network with that of a real network, but due to time limitations this could not be performed. For future research it is proposed to test how the emulated routers cope with a large quantity of packets being sent, and to determine if it can actually cope with the volume or will the end result be a big percentage of the packets being dropped.

6.4.2 Challenge Creation

Challenges 2 and 3 use the idea of fading worked examples [5] to lowering the cognitive load of the student, by providing pre-configuration of various areas of the router through the use of startup-config scripts. One of the problems with the implementation of this approach is that while the specification of the challenge changes every time it is run, the startup-config scripts are always the same.

Therefore if the student has already carried out the challenge once, then the next time they perform the challenge they may remember the information required to complete the challenge, instead of having to perform the debug and analysis operations to gather the information. This has the end result of lowering the Bloom rating of the challenges since the student is not using the Analysis level of the cognitive domain. To resolve this problem it is proposed that in a future development the startup-config files are dynamically generated thereby increasing the uniqueness of each challenge even more.

Each of the challenges uses a fixed network topology that was pre-created using the GNS3 program and stored in the Dynagen INI file format. A suggested future improvement is to create the network topology dynamically, so that the student is presented with a different topology each time they attempt the challenge, and this has the end result of making the challenge creation quicker for the instructor.

One of the major problems with the development of the challenges is the number of steps that the instructor has to follow to generate all the relevant data. For a future development a better way of specifying the challenge would be to have a template that could be used to specify various concepts that the challenge should have. Once the template has been specified a generator could be run that would automatically create all the relevant data for the challenge.

Each of the challenges has been designed with the intent of a single user performing the entire configuration. A skill that employers are wanting in graduates is that of team working [46, 47]. A proposed improvement is to investigate if it is feasible to design challenges that would require collaborative working to solve the problem.

6.4.3 Bloom Challenge Level

The maximum Bloom level [3] that is tested on the challenges is level 4. A proposed future development is to develop challenges that can test at levels 5 (Synthesis) and level 6 (Evaluation).

For challenges at level 5 the student could be asked to design a network to a given design, and it would be up to the student to decided how they would end up implemented everything. This is similar to the challenges that Cisco gives its academy students before they can graduate from the module.

For challenges at level 6 the student could be presented with a network scenario and asked to improve it. In the scenario there may be some things that have not been implemented efficiently, like using RIP instead of EIGRP, and it would be up to the student to recognise this fact and implement the correct solution.

6.4.4 Challenge Grading

A problem with the feedback being presented to the student is that it only displays the correct command, and not the entire series of commands required to get to the correct command prompt. For example if the IP address was incorrectly configured the correct command is displayed, and not the steps for how to get to the correct interface. A future improvement would be to detail all the required steps and not just the one command.

When the challenge is being marked the marking criteria is if the student has got the answer correct or incorrect. There is no mark presented that would indicate to the student how they performed. In future developments the student mark could be based on how long they took to perform the task, how many commands they used, how many times they had to access the challenge help bank and how many times they had to use the help feature of the IOS.

Each of the challenges has got specific tasks that the student has to perform and there may be times when the student is strong on certain tasks but not on others. A future release should monitor the student when they are performing each task and take note if they are struggling with the task. When the student is presented with the feedback it should include suggestions for where they should concentrate in improving their skills.

6.4.5 Challenge Implementation

With the current design the student is required to perform a few steps before they can begin the challenge. The student has to run the emulator, load the network configuration, start all the routers and then run the challenge. The design needs to be modified so that the user should be able to select the challenge, and then everything should load automatically and be in a state where the student can start instantly.

From the section on the implementation of the challenge generator it can be seen that the generation of the subnets and loopback addresses are limited. With the generation of Class A and B subnets the full range of subnets are not being used, and with the loopback addresses they are restricted to Class C only. These problems should be resolved in a future release so that the challenges provide a greater scope of difficulty.

One of the major problems with trying to integrate all the parts together to provide the overall framework is getting each part talking to each other. This was a problem when trying to resolve how to obtain the startup-config from GNS3. In the end the solution found of extracting the startup-config from the NVRAM files is far from ideal, and one of the limiting factors was the lack of programming hooks available to monitor the student. One of the proposed changes for future development is that instead of using GNS3 as the GUI a new one should be written from scratch that would allow more control and interaction with the routers.

If the overall solution was being considered for commercial release then there are two major obstacles that would have to be overcome. The first major obstacle is that to use Dynamips a copy of Cisco's IOS firmware for the router is required. The only legal way to obtain the firmware is to download it from Cisco's customer web site but this requires a person to own a Cisco router first.

The second major obstacle that would have to be resolved is that the Dynamips program is released under the GNU General Public Licence (GPL). If the public were charged for the solution then it would have to be verified that the GPL was not being breached, and that the original developers did not require to be reimbursed for their work.

The Dynamips program can emulate a wide range of routers as well as PIX firewall devices. However one device that has not been emulated is a switch. The main reason for this is that a switch is based primarily on hardware and not software, and one of the major problems with functions being performed in hardware is determining how everything works and also the speed of execution. The performance of a software-emulated switch will not be able to match that of a hardware switch, but there is a software switch in Dynamips, but it is not fully functional and this should be taken into consideration when designing exercises with switches.

The overall solution is comprised of many components that have to be installed individually. Getting all the components installed to the right location can take time and effort, and to speed this process up it is proposed that an installation package (MSI) should be created that would install all the components automatically, and thus reduce the scope for installation errors.

Another problem with the prototype is that standard functions have not been written for marking the tasks. For example in each challenge the marking function has implemented its own code for checking that an interface has been configured properly. In a future release common code should be used for marking generic tasks.

One of the main benefits of using virtualisation is the ability to set up remote labs for students to use at any time during the day. It should be investigated if this possible to do with Dynamips.

In the development of the challenges only router devices were used despite other devices being available. Another device that should be taken into future consideration is that of the PIX firewall. In Dynamips the PIX can is fully emulated and this would help with the design of security courses.

Appendix A - Challenge 2 Startup-Configs

Appendix A.1 - Router R0

```
!  
version 12.1  
no service single-slot-reload-enable  
service timestamps debug uptime  
service timestamps log uptime  
no service password-encryption  
!  
hostname Paris  
!  
!  
!  
!  
!  
!  
memory-size iomem 15  
ip subnet-zero  
!  
!  
!  
!  
interface FastEthernet0/0  
ip address 172.45.0.1 255.255.0.0  
duplex auto  
speed auto  
!  
interface FastEthernet0/1  
no ip address  
shutdown  
duplex auto  
speed auto
```

```
!  
ip classless  
ip http server  
!  
!  
line con 0  
line aux 0  
line vty 0 4  
!  
end
```

Appendix A.2 Router R1

```
!  
version 12.1  
no service single-slot-reload-enable  
service timestamps debug uptime  
service timestamps log uptime  
no service password-encryption  
!  
hostname LA  
!  
!  
!  
!  
!  
!  
memory-size iomem 15  
ip subnet-zero  
!  
!  
!  
!  
interface FastEthernet0/0  
ip address 172.45.0.2 255.255.0.0  
duplex auto  
speed auto  
!  
interface FastEthernet0/1  
no ip address  
shutdown  
duplex auto  
speed auto  
!  
ip classless
```

```
ip http server
```

```
!
```

```
!
```

```
line con 0
```

```
line aux 0
```

```
line vty 0 4
```

```
!
```

```
end
```

Appendix B - Challenge 3 Startup-Configs

Appendix B.1 – Paris Router

```
!  
version 12.1  
no service single-slot-reload-enable  
service timestamps debug uptime  
service timestamps log uptime  
no service password-encryption  
!  
hostname Paris  
!  
!  
!  
!  
!  
!  
!  
memory-size iomem 15  
ip subnet-zero  
!  
!  
!  
!  
interface Loopback1  
ip address 209.165.179.1 255.255.255.0  
!  
interface Loopback2  
ip address 209.165.189.1 255.255.255.0  
!  
interface Loopback3  
ip address 209.165.199.1 255.255.255.0  
!  
interface Loopback4
```



```
ip address 209.165.169.1 255.255.255.0
```

```
!
```

```
interface FastEthernet0/0
```

```
ip address 192.168.10.2 255.255.255.0
```

```
duplex auto
```

```
speed auto
```

```
!
```

```
interface FastEthernet0/1
```

```
no ip address
```

```
shutdown
```

```
duplex auto
```

```
speed auto
```

```
!
```

```
router eigrp 55017
```

```
network 192.168.10.0
```

```
network 209.165.169.0
```

```
network 209.165.179.0
```

```
network 209.165.189.0
```

```
network 209.165.199.0
```

```
auto-summary
```

```
no eigrp log-neighbor-changes
```

```
!
```

```
ip classless
```

```
ip http server
```

```
!
```

```
!
```

```
line con 0
```

```
line aux 0
```

```
line vty 0 4
```

```
login
```

```
!
```

```
end
```

Appendix B.2 – London Router

```
version 12.1
no service single-slot-reload-enable
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname London
!
!
!
!
!
!
!
memory-size iomem 15
ip subnet-zero
!
!
!
!
interface Loopback1
ip address 220.88.4.1 255.255.255.0
!
interface Loopback2
ip address 220.88.14.1 255.255.255.0
!
interface Loopback3
ip address 220.88.24.1 255.255.255.0
!
interface Loopback4
ip address 220.88.34.1 255.255.255.0
!
interface FastEthernet0/0
ip address 192.168.10.1 255.255.255.0
duplex auto
```

```
speed auto
!
interface FastEthernet0/1
ip address 172.16.0.1 255.255.0.0
shutdown
duplex auto
speed auto
!
router eigrp 55017
network 192.168.10.0
network 220.88.4.0
network 220.88.14.0
network 220.88.24.0
network 220.88.34.0
auto-summary
no eigrp log-neighbor-changes
!
ip classless
ip http server
!
!
line con 0
line aux 0
line vty 0 4
login
!
end
```

Appendix B.3 – Edinburgh Router

```
!  
version 12.1  
no service single-slot-reload-enable  
service timestamps debug uptime  
service timestamps log uptime  
no service password-encryption  
!  
hostname Edinburgh  
!  
!  
!  
!  
!  
!  
memory-size iomem 15  
ip subnet-zero  
!  
!  
!  
!  
interface FastEthernet0/0  
ip address 172.16.0.2 255.255.0.0  
shutdown  
duplex auto  
speed auto  
!  
interface FastEthernet0/1  
no ip address  
shutdown  
duplex auto  
speed auto  
!  
ip classless  
ip http server
```

```
!  
!  
line con 0  
line aux 0  
line vty 0 4  
!  
end
```

Appendix C - Sample Dynagen File

```
autostart = False
[localhost:7202]
    workingdir = C:\DOCUME~1\dave\LOCALS~1\Temp
    udp = 10200
    [[2621]]
        image = C:\Documents and Settings\dave\Desktop\IOS file\c2600-i-mz.121-27b.bin
        idlepc = 0x8027070c
        ghostios = True
        chassis = 2621
    [[ROUTER Edinburgh]]
        model = 2621
        console = 2004
        cnfg = C:\exercise\Ex3\Edinburgh.cfg
        f0/0 = London f0/1
        x = 35.0
        y = 37.0
[localhost:7200]
    workingdir = C:\DOCUME~1\dave\LOCALS~1\Temp
    [[2621]]
        image = C:\Documents and Settings\dave\Desktop\IOS file\c2600-i-mz.121-27b.bin
        idlepc = 0x8027070c
        ghostios = True
        chassis = 2621
    [[ROUTER London]]
        model = 2621
        console = 2006
        cnfg = C:\exercise\Ex3\London.cfg
        f0/0 = Paris f0/0
        f0/1 = Edinburgh f0/0
        x = -112.0
        y = -97.0
[localhost:7201]
```

workingdir = C:\DOCUME~1\dave\LOCALS~1\Temp

udp = 10100

[[2621]]

image = C:\Documents and Settings\dave\Desktop\IOS file\c2600-i-mz.121-27b.bin

idlepc = 0x8027070c

ghostios = True

chassis = 2621

[[ROUTER Paris]]

model = 2621

console = 2005

cnfg = C:\exercise\Ex3\Paris.cfg

f0/0 = London f0/0

x = -241.0

y = 37.0

Appendix D – Challenge Generator Code Listing

```
using System;
using System.Collections;
using System.Text;
using System.IO; // without this name space it would be impossible to read in a file
using System.Threading;

namespace Ex01_SimpleRead
{
    /// <summary>
    /// This application reads in the content of the the file "demofile.txt".
    /// </summary>
    /// <remarks>
    /// The source text file is expected to be located within the same folder
    /// as the source code of this application.
    /// </remarks>
    class Demo
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]

        static void Main(string[] args)
        {
            challenge1 myChallenge1 = new challenge1();
            challenge2 myChallenge2 = new challenge2();
            challenge3 myChallenge3 = new challenge3();
            myChallenge1.process();
            Console.ReadLine();
            myChallenge2.process();
        }
    }
}
```



```
        Console.ReadLine();
        myChallenge3.process();
            Console.ReadLine();
        }//main

    }//class

class loopback
{
    private ArrayList myLoopIp = new ArrayList();
    private ArrayList myLoopSubnet = new ArrayList();
    string subnet = "255.255.255.0";

    public loopback()
    {
        string ipLoopback;
        string subnetAddress;
        Thread.Sleep(1);
        Random fixRand = new Random();

        subnetAddress = fixRand.Next(192, 223).ToString() + "." + fixRand.Next(0, 254).ToString()
+ ".";
        int tempOctet = fixRand.Next(0, 214);

        for (int i=0; i < 4; i++)
        {
            ipLoopback = subnetAddress + tempOctet.ToString() + ".1";
            myLoopIp.Add(ipLoopback);
            myLoopSubnet.Add((subnetAddress + tempOctet.ToString() + ".0"));
            tempOctet = tempOctet + 10;
        }
    }
    public void displayLoop()
    {
```

```
IEnumerator myEnum = myLoopIp.GetEnumerator();
int count = 1;

while (myEnum.MoveNext())
{
    Console.WriteLine("Loopback" + count + " address " + myEnum.Current);
    count++;
}

public string getLoopIp(int loopNum)
{
    return myLoopIp[loopNum].ToString();
}

public string getSubnetIp(int loopNum)
{
    return myLoopSubnet[loopNum].ToString();
}

public string getSubnet()
{
    return subnet;
}

class router
{
    private string[] hostArray = new string[6] { "Perth", "Dundee", "Edinburgh", "London",
"Paris", "Amsterdam" };
    private string[] subnetMasks = new string[7] { "0", "128", "192", "224", "240", "248", "252" };
    public loopback myLoop = new loopback();
    string hostName;
    string ipAddress;
    string ipNextAddress;
```

```
string ipSubnetMask;
string routerName;
int eigrpAS;
int bandwidth;
int delay;

private Random fixRand = new Random();

public router(string myName)
{
    routerName = myName;
    // choose hostname
    hostName = hostArray[fixRand.Next(0,2)];
    int numLoop = fixRand.Next(0, 6);
    eigrpAS = fixRand.Next(1, 65535);
    //choose ip address and subnet
    int ipClass = fixRand.Next(1, 3);
    string mySubnetMask = subnetMasks[fixRand.Next(0, 6)];
    string tempIpAddress;
    delay = fixRand.Next(1, 1000);
    bandwidth = fixRand.Next(1, 50000);

    if (ipClass == 1) //class A
    {
        tempIpAddress = fixRand.Next(1, 126).ToString();
        if (mySubnetMask == "0")
        {
            ipAddress = tempIpAddress + ".0.0.1";
            ipNextAddress = tempIpAddress + ".0.0.2";
        }
        else
        {
            ipAddress = tempIpAddress + "." + (256 - Convert.ToInt32(mySubnetMask)) + ".0.1";
            ipNextAddress = tempIpAddress + "." + (256 - Convert.ToInt32(mySubnetMask)) +
".0.2";
```

```
    }
    ipSubnetMask = "255." + mySubnetMask + ".0.0";
}

if (ipClass == 2) //class B
{
    if (mySubnetMask == "0")
    {

    }
    else
    {
        tempIpAddress = fixRand.Next(128, 191).ToString() + "." + fixRand.Next(0,
254).ToString();
        ipAddress = tempIpAddress + "." + (256 - Convert.ToInt32(mySubnetMask)) + ".1";
        ipNextAddress = tempIpAddress + "." + (256 - Convert.ToInt32(mySubnetMask)) +
".2";
    }
    ipSubnetMask = "255.255." + mySubnetMask + ".0";
}

if (ipClass == 3) //class C
{
    tempIpAddress = fixRand.Next(192, 223).ToString() + "." + fixRand.Next(0,
254).ToString() + "." + fixRand.Next(0, 254).ToString();
    if (mySubnetMask == "0")
    {
        ipAddress = tempIpAddress + ".255.1";
        ipNextAddress = tempIpAddress + ".255.2";
    }
    else
    {
        ipAddress = tempIpAddress + "." + ((256 - Convert.ToInt32(mySubnetMask)) + 1);
        ipNextAddress = tempIpAddress + "." + ((256 - Convert.ToInt32(mySubnetMask)) + 2);
    }

    ipSubnetMask = "255.255.255." + mySubnetMask;
```

```
    }
}

public router(router connectRouter, string myName)
{
    eigrpAS = connectRouter.getEigrp();
    routerName = myName;
    // choose hostname
    hostName = hostArray[fixRand.Next(3, 5)];
    ipAddress = connectRouter.getNextIpAddress();
    ipSubnetMask = connectRouter.getIpSubnetMask();
    bandwidth = connectRouter.getBand();
    delay = connectRouter.getDelay();
}

public string getHostname() { return hostName; }
public string getIp() { return ipAddress; }
public string getIpSubnetMask() { return ipSubnetMask; }
public string getNextIpAddress() { return ipNextAddress; }
public string getRouterId() { return routerName; }
public int getEigrp() { return eigrpAS; }
public int getBand() { return bandwidth; }
public int getDelay() { return delay; }

}

class challenge1
{
    private ArrayList myStartup = new ArrayList();
    private static router myRouter0 = new router("R0");
    private static router myRouter1 = new router(myRouter0, "R1");
    private int myIterator(string toFind)
    {
        IEnumerator myEnum = myStartup.GetEnumerator();
        string tempString;
        int indexCount = -1;
    }
}
```

```
bool found = false;

while (myEnum.MoveNext() && !found)
{
    indexCount++;
    tempString = myEnum.Current.ToString();
    if (tempString.StartsWith(toFind))
    {
        found = true;
    }
}

if (found)
    return indexCount;
else
    return 0;
}

public void process()
{
    displaySpec(myRouter0);
    displaySpec(myRouter1);
    Console.ReadLine();

    processConfigs("C:\\Documents and Settings\\dave\\Local
Settings\\Temp\\c2600_R0_nvram", "C:\\exercise\\Ex1\\R0.txt");
    displayRun("C:\\exercise\\Ex1\\R0.txt");
    int index = myStartup.IndexOf("version 12.1");
    if (index < 0)
    {
        Console.WriteLine("Oops, user has forgotten to copy running config to startup config for
" + myRouter0.getHostname());
    }
    else
    {
        checkConfigs(myRouter0);
    }
}
```

```
    }

    myStartup.Clear();

    processConfigs("C:\\Documents and Settings\\dave\\Local
Settings\\Temp\\c2600_R1_nvram", "C:\\exercise\\Ex1\\R1.txt");
    displayRun("C:\\exercise\\Ex1\\R1.txt");

    index = myStartup.IndexOf("version 12.1");
    if (index < 0)
    {
        Console.WriteLine("Ooops, user has forgotten to copy running config to startup config for
" + myRouter1.getHostname());
    }
    else
    {
        checkConfigs(myRouter1);
    }
}

private void processConfigs(string nvram, string output)
{
    StreamReader sr = new StreamReader(@nvram);
    StreamWriter sw = new StreamWriter(@output);
    bool start = false;
    bool finished = false;
    string theLine = "";
    string[] stringSeparators = new string[] { "\\0\\0\\0" };

    while ((theLine = sr.ReadLine()) != null)
    {
        string[] elements = theLine.Split(stringSeparators,
StringSplitOptions.RemoveEmptyEntries);
        if (elements.Length != 0)
        {
            if (elements[0] == "!" && finished == false)
            {
                start = true;
            }
        }
    }
}
```

```
    }
}

if (start == true)
{
    foreach (string substring in elements)
    {
        sw.Write(substring);
    }

    if (elements.GetLength(0) != 0)
        sw.WriteLine();

    if (elements.GetLength(0) != 0)
    {
        if (elements[0] == "e" && elements[1] == "n" && elements[2] == "d")
        {
            start = false;
            finished = true;
        }
    }
}

//Console.WriteLine("There is nothing left in the file");
sr.Close();
sw.Close();
}

public void displayRun(string myFile)
{
    StreamReader sr2 = new StreamReader(@myFile);
    string theLine = "";

    while ((theLine = sr2.ReadLine()) != null)
    {
```



```
        myStartup.Add(theLine);
    }
    sr2.Close();
}

public void checkConfigs(router myRouter)
{
    Console.WriteLine("Checking Configuration for " + myRouter.getRouterId());
    int index = myStartup.IndexOf("hostname " + myRouter.getHostname());
    int tempIndex;

    if (index < 0)
    {
        Console.WriteLine("");
        Console.WriteLine("Hostname has not been set as specified");
        tempIndex = myIterator("hostname");

        if (tempIndex > 0)
        {
            Console.WriteLine("Hostname line is set to:- " + myStartup[tempIndex]);
            Console.WriteLine("The correct ISO command is :- hostname " +
myRouter.getHostname());
        }
        else
        {
            Console.WriteLine("Hostname has not been configured");
        }
    }
    else
    {
        Console.WriteLine("Hostname has been configured as specified");
    }

    int firstIndex = myStartup.IndexOf("interface FastEthernet0/0");
    int lastIndex = myStartup.IndexOf("interface FastEthernet0/1");
    if (firstIndex < 0)
    {
        Console.WriteLine("Oops we have an error");
    }
}
```

```
else
{
    if (myStartup.IndexOf(" no ip address", firstIndex, (lastIndex - firstIndex)) != -1)
    {
        Console.WriteLine("IP address is not set on FastEthernet0/0");
        Console.WriteLine("Current line is:- no ip address");
        Console.WriteLine("Correct line is:- ip address " + myRouter.getIp() + " " +
myRouter.getIpSubnetMask());
    }
    else
    {
        index = myStartup.IndexOf(" ip address " + myRouter.getIp() + " " +
myRouter.getIpSubnetMask(), firstIndex, (lastIndex - firstIndex));
        if (index < 0)
        {
            Console.WriteLine("IP address has not been set as specified");
            tempIndex = myIterator(" ip address");

            if (tempIndex > 0)
            {
                Console.WriteLine("Hostname line is set to:- " + myStartup[tempIndex]);
                Console.WriteLine("The correct ISO command is :- ip address " + myRouter.getIp()
+ " " + myRouter.getIpSubnetMask());
            }
            else
            {
                Console.WriteLine("Hostname has not been configured");
            }
        }
        else
        {
            Console.WriteLine("IP address has been configured correctly");
        }
    }
}

if (myStartup.IndexOf(" shutdown", firstIndex, (lastIndex - firstIndex)) != -1)
{
    Console.WriteLine("FastEthernet0/0 is in the shutdown state");
    Console.WriteLine("The correct command to use is :- no shutdown");
}
```

```
    }
    else
    {
        Console.WriteLine("FastEthernet0/0 is in the up state");
    }
}

public void displaySpec(router myRouter)
{
    Console.WriteLine("Configure " + myRouter.getRouterId());
    Console.WriteLine("1. Set R0 hostname to " + myRouter.getHostname());
    Console.WriteLine("2. Set IP address " + myRouter.getIp() + " with subnet " +
myRouter.getIpSubnetMask() + " on Fastethernet0/0");
    Console.WriteLine("3. Bring up Fastethernet0/0");
}
}
```

class challenge2

```
{
    private ArrayList myStartup = new ArrayList();
    private static router myRouter0 = new router("R0");
    private static router myRouter1 = new router(myRouter0, "R1");
    private int myIterator(string toFind)
    {
        IEnumerator myEnum = myStartup.GetEnumerator();
        string tempString;
        int indexCount = -1;
        bool found = false;

        while (myEnum.MoveNext() && !found)
        {
            indexCount++;
            tempString = myEnum.Current.ToString();
            if (tempString.StartsWith(toFind))
            {
                found = true;
            }
        }
    }
}
```

```
    }

}

if (found)
    return indexCount;
else
    return 0;
}

public void process()
{
    displaySpec(myRouter0);
    displaySpec(myRouter1);
    Console.ReadLine();
    processConfigs("C:\\Documents and Settings\\dave\\Local
Settings\\Temp\\c2600_R0_nvram", "C:\\exercise\\Ex2\\R0.txt");
    displayRun("C:\\exercise\\Ex2\\R0.txt");

    int index = myStartup.IndexOf("version 12.1");
    if (index < 0)
    {
        Console.WriteLine("Oops, user has forgotten to copy running config to startup config for
" + myRouter0.getHostname());
    }
    else
    {
        checkConfigs(myRouter0);
    }

    myStartup.Clear();
    processConfigs("C:\\Documents and Settings\\dave\\Local
Settings\\Temp\\c2600_R1_nvram", "C:\\exercise\\Ex2\\R1.txt");
    displayRun("C:\\exercise\\Ex2\\R1.txt");
    index = myStartup.IndexOf("version 12.1");
    if (index < 0)
```

```
    {
        Console.WriteLine("Ooops, user has forgotten to copy running config to startup config for
" + myRouter1.getHostname());
    }
    else
    {
        checkConfigs(myRouter1);
    }
}
```

```
private void processConfigs(string nvram, string output)
{
    StreamReader sr = new StreamReader(@nvram);
    StreamWriter sw = new StreamWriter(@output);
    bool start = false;
    bool finished = false;
    string theLine = "";
    string[] stringSeparators = new string[] { "\\0\\0\\0" };

    while ((theLine = sr.ReadLine()) != null)
    {
        string[] elements = theLine.Split(stringSeparators,
StringSplitOptions.RemoveEmptyEntries);
        if (elements.Length != 0)
        {
            if (elements[0] == "!" && finished == false)
            {
                start = true;
            }
        }
    }

    if (start == true)
    {
        foreach (string substring in elements)
        {
            sw.Write(substring);
        }
    }
}
```

```
    }

    if (elements.GetLength(0) != 0)
        sw.WriteLine();

    if (elements.GetLength(0) != 0)
    {

        if (elements[0] == "e" && elements[1] == "n" && elements[2] == "d")
        {
            start = false;
            finished = true;
        }
    }
}

//Console.WriteLine("There is nothing left in the file");
sr.Close();
sw.Close();
}

public void displayRun(string myFile)
{
    StreamReader sr2 = new StreamReader(@myFile);
    string theLine = "";

    while ((theLine = sr2.ReadLine()) != null)
    {
        myStartup.Add(theLine);
    }
    sr2.Close();
}

public void checkConfigs(router myRouter)
{
    Console.WriteLine("Checking Configuration for " + myRouter.getRouterId());
}
```

```
for (int i =1; i < 5; i++)
{
    int index = myStartup.IndexOf("interface Loopback" + i.ToString());
    //int tempIndex;

    if (index < 0)
    {
        Console.WriteLine("");
        Console.WriteLine("Loopback has not been set as specified");
        Console.WriteLine("The correct ISO command is :- interface loopback" + i.ToString());
    }
    else
    {
        Console.WriteLine("Loopback" + i.ToString() + " has been specified correctly");

        if ( myStartup[index+1].ToString() == "no ip address")
        {
            Console.WriteLine("Loopback" + i.ToString() + "no ip address has been specified for
Loopback" + i.ToString());

            Console.WriteLine("The correct IOS command is :- ip address " +
myRouter.myLoop.getLoopIp(i-1) + " " + myRouter.myLoop.getSubnet());
        }
        else
        {

            if ( myStartup[index+1].ToString() == (" ip address " +
myRouter.myLoop.getLoopIp(i-1) + " " + myRouter.myLoop.getSubnet()))
            {
                Console.WriteLine("Ip address for Loopback" + i.ToString() + "is correct");
            }
            else
            {
                Console.WriteLine("Ip address for Loopback" + i.ToString() + "is incorrect");

                Console.WriteLine("The IOS command that was used :- " +
myStartup[index+1].ToString());

                Console.WriteLine("The correct IOS command is :- ip address " +
myRouter.myLoop.getLoopIp(i-1) + " " + myRouter.myLoop.getSubnet());
            }
        }
    }
}
```

```
        }
    }
}

int eigrpIndex = myStartup.IndexOf("router eigrp " + myRouter.getEigrp());
if (eigrpIndex < 0)
{
    Console.WriteLine("Eigrp has not been configured properly");
    Console.WriteLine("The correct IOS command is :- router eigrp " +
myRouter.getEigrp());
}
else
{
    Console.WriteLine("Eigrp has been configured correctly");
    int index2 = myStartup.IndexOf(" network 172.45.0.0");
    if (index2 < 0)
    {
        Console.WriteLine("Directly connected links between R0 and R1 have not been
advertised via EIGRP");
        Console.WriteLine("The correct IOS command is:- network 172.45.0.0");
    }
    else
    {
        Console.WriteLine("The directly connected links between R0 and R1 have been
correctly advertised via EIGRP");
        eigrpIndex++;
    }
}

for (int i = 1; i < 5; i++)
{

    index2 = myStartup.IndexOf(" network " + myRouter.myLoop.getSubnetIp(i-1));
    if (index2 < 0)
    {
```



```
        Console.WriteLine("Loopback" + i.ToString() + "has not been advertised via EIGRP properly");
        Console.WriteLine("The current IOS command is :- " + myStartup[eigrpIndex + i]);
        Console.WriteLine("The correct IOS command to use is network " + myRouter.myLoop.getSubnetIp(i-1));
    }
    else
    {
        Console.WriteLine("Loopback" + i.ToString() + " has been configured correctly via EIGRP");
    }
}

}

public void displaySpec(router myRouter)
{
    Console.WriteLine("Configure " + myRouter.getRouterId());
    Console.WriteLine("The link between R0 and R1 has already been preconfigured");
    Console.WriteLine("1. Set up 4 loopback addresses as follows");
    myRouter.myLoop.displayLoop();
    Console.WriteLine("2. Set up EIGRP routing with AS " + myRouter.getEigrp());
    Console.WriteLine("3. Advertise the loopback addresses using EIGRP");
}
}

class challenge3
{
    private ArrayList myStartup = new ArrayList();
    private static router myRouter0 = new router("Paris");
    private static router myRouter1 = new router(myRouter0, "London");
    private static router myRouter2 = new router(myRouter1, "Edinburgh");

    private int myIterator(string toFind)
    {
        IEnumerator myEnum = myStartup.GetEnumerator();
        string tempString;
```

```
int indexCount = -1;
bool found = false;

while (myEnum.MoveNext() && !found)
{
    indexCount++;
    tempString = myEnum.Current.ToString();
    if (tempString.StartsWith(toFind))
    {
        found = true;
    }
}

if (found)
    return indexCount;
else
    return 0;
}

public void process()
{
    displaySpec(myRouter2);
    Console.ReadLine();
    processConfigs("C:\\Documents and Settings\\dave\\Local
Settings\\Temp\\c2600_London_nvram", "C:\\exercise\\Ex3\\London.txt");
    displayRun("C:\\exercise\\Ex3\\London.txt");

    int index = myStartup.IndexOf("version 12.1");
    if (index < 0)
    {
        Console.WriteLine("Ooops, user has forgotten to copy running config to startup config for
London");
    }
    else
```

```
{
    checkConfigsRedis(myRouter1);
}

myStartup.Clear();

processConfigs("C:\\Documents and Settings\\dave\\Local
Settings\\Temp\\c2600_Edinburgh_nvram", "C:\\exercise\\Ex3\\Edinburgh.txt");
displayRun("C:\\exercise\\Ex3\\Edinburgh.txt");
index = myStartup.IndexOf("version 12.1");
if (index < 0)
{
    Console.WriteLine("Ooops, user has forgotten to copy running config to startup config for
Edinburgh");
}
else
{
    checkConfigs(myRouter2);
}
}

private void processConfigs(string nvram, string output)
{
    StreamReader sr = new StreamReader(@nvram);
    StreamWriter sw = new StreamWriter(@output);
    bool start = false;
    bool finished = false;
    string theLine = "";
    string[] stringSeparators = new string[] { "\\0\\0\\0" };

    while ((theLine = sr.ReadLine()) != null)
    {
        string[] elements = theLine.Split(stringSeparators,
StringSplitOptions.RemoveEmptyEntries);
        if (elements.Length != 0)
        {
            if (elements[0] == "!" && finished == false)
            {
```

```
        start = true;
    }
}

if (start == true)
{
    foreach (string substring in elements)
    {
        sw.Write(substring);
    }

    if (elements.GetLength(0) != 0)
        sw.WriteLine();

    if (elements.GetLength(0) != 0)
    {
        if (elements[0] == "e" && elements[1] == "n" && elements[2] == "d")
        {
            start = false;
            finished = true;
        }
    }
}

//Console.WriteLine("There is nothing left in the file");
sr.Close();
sw.Close();
}

public void displayRun(string myFile)
{
    StreamReader sr2 = new StreamReader(@myFile);
    string theLine = "";

    while ((theLine = sr2.ReadLine()) != null)
```

```
{
    myStartup.Add(theLine);
}

sr2.Close();
}

public void checkConfigsRedis(router myRouter)
{
    Console.WriteLine("Checking Configuration for " + myRouter.getRouterId());
    int index = myStartup.IndexOf("interface FastEthernet0/1");

    if ((myStartup[index + 2]).ToString() == " shutdown")
    {
        Console.WriteLine("Interface FastEthernet0/1 is still in the shutdown state");
        Console.WriteLine("The correct IOS command line to use is:- no shutdown");
    }
    else
    {
        Console.WriteLine("Interface FastEthernet 0/1 has been enable correctly");
    }

    int ospfIndex = myStartup.IndexOf("router ospf 1");
    if (ospfIndex < 0)
    {
        Console.WriteLine("OSPF has not been configured properly");
        Console.WriteLine("The correct IOS command is :- router ospf 1");
    }
    else
    {
        Console.WriteLine("OSPF has been configured correctly");
        int index2 = myStartup.IndexOf(" network 172.16.0.0 0.0.255.255 area 0");
        if (index2 < 0)
        {
            Console.WriteLine("Directly connected links between London and Edinburgh have not
been advertised via OSPF");
            Console.WriteLine("The correct IOS command is:- network 172.16.0.0 0.0.255.255
area 0");
        }
    }
}
```

```
    }
    else
    {
        Console.WriteLine("The directly connected links between London and Edinburgh have
        been correctly advertised via OSPF");
    }
}

int myNum = Convert.ToInt32(myRouter.getDelay()) * 10;

index = myStartup.IndexOf(" redistribute ospf 1 metric " + myRouter.getBand() + " " +
myNum.ToString() + " 255 1 100");

if (index < 0)
{
    Console.WriteLine("OSPF has not been redistributed into EIGRP correctly");

    Console.WriteLine("The correct IOS command to use is :- redistribute ospf 1 metric " +
myNum.ToString() + " " + Convert.ToInt32(myRouter.getDelay()) * 10 + " 255 1 100");
}
else
{
    Console.WriteLine("OSPF has been redistributed into EIGRP correctly");
}

index = myStartup.IndexOf(" redistribute eigrp 55017");

if (index < 0)
{
    Console.WriteLine("EIGRP has not been redistributed into OSPF as specified");

    Console.WriteLine("The correct ios command to use is :- redistribute eigrp " +
myRouter.getEigrp());
}
else
{
    Console.WriteLine(" EIGRP has been redistributed into OSPF as specified");
}
}

public void checkConfigs(router myRouter)
{

```

```
Console.WriteLine("Checking Configuration for " + myRouter.getRouterId());

int index = myStartup.IndexOf("interface FastEthernet0/0");

if ((myStartup[index + 2]).ToString() == " shutdown")
{
    Console.WriteLine("Interface FastEthernet0/1 is still in the shutdown state");
    Console.WriteLine("The correct IOS command line to use is no shutdown");
}
else
{
    Console.WriteLine("Interface FastEthernet 0/1 has been enable correctly");
}

for (int i = 1; i < 5; i++)
{
    index = myStartup.IndexOf("interface Loopback" + i.ToString());
    //int tempIndex;

    if (index < 0)
    {
        Console.WriteLine("");
        Console.WriteLine("Loopback has not been set as specified");
        Console.WriteLine("The correct ISO command is :- interface loopback" + i.ToString());
    }
    else
    {
        Console.WriteLine("Loopback" + i.ToString() + " has been specified correctly");

        if (myStartup[index + 1].ToString() == "no ip address")
        {
            Console.WriteLine("Loopback" + i.ToString() + "no ip address has been specified for
Loopback" + i.ToString());

            Console.WriteLine("The correct IOS command is :- ip address " +
myRouter.myLoop.getLoopIp(i - 1) + " " + myRouter.myLoop.getSubnet());
        }
    }
}
```

```
        else
        {

            if (myStartup[index + 1].ToString() == (" ip address " +
myRouter.myLoop.getLoopIp(i - 1) + " " + myRouter.myLoop.getSubnet()))
            {
                Console.WriteLine("Ip address for Loopback" + i.ToString() + "is correct");
            }
            else
            {
                Console.WriteLine("Ip address for Loopback" + i.ToString() + "is incorrect");
                Console.WriteLine("The IOS command that was used :- " + myStartup[index +
1].ToString());
                Console.WriteLine("The correct IOS command is :- ip address " +
myRouter.myLoop.getLoopIp(i - 1) + " " + myRouter.myLoop.getSubnet());
            }
        }
    }
}

int ospfIndex = myStartup.IndexOf("router ospf 1");
if (ospfIndex < 0)
{
    Console.WriteLine("OSPF has not been configured properly");
    Console.WriteLine("The correct IOS command is :- router ospf 1");
}
else
{
    Console.WriteLine("OSPF has been configured correctly");
    int index2 = myStartup.IndexOf(" network 172.16.0.0 0.0.255.255 area 0");
    if (index2 < 0)
    {
        Console.WriteLine("Directly connected links between London and Edinburgh have not
been advertised via OSPF");
        Console.WriteLine("The correct IOS command is:- network 172.16.0.0 0.0.255.255
area 0");
    }
}
```



```
    }
    else
    {
        Console.WriteLine("The directly connected links between London and Edinburgh have
been correctly advertised via OSPF");
        ospfIndex+=2;
    }

    for (int i = 1; i < 5; i++)
    {

        index2 = myStartup.IndexOf(" network " + myRouter.myLoop.getSubnetIp(i - 1) + "
0.0.0.255 area 0");
        if (index2 < 0)
        {
            Console.WriteLine("Loopback" + i.ToString() + "has not been advertised via OSPF
properly");

            Console.WriteLine("The current IOS command is :- " + myStartup[ospfIndex + i]);

            Console.WriteLine("The correct IOS command to use is network " +
myRouter.myLoop.getSubnetIp(i - 1) + " 0.0.0.255 area 0");
        }
        else
        {
            Console.WriteLine("Loopback" + i.ToString() + " has been configured correctly via
OSPF");
        }
    }
}

public void displaySpec(router myRouter)
{
    Console.WriteLine("The link between Paris, London and Edinburgh has already been
preconfigured");

    Console.WriteLine("Eigrp has already been configured between Paris and London");

    Console.WriteLine("1. Set up 4 loopback addresses as follows on " +
myRouter.getRouterId() + " and advertise via OSPF with process number 1");

    myRouter.myLoop.displayLoop();
}
```

```
        Console.WriteLine("Configure OSPF between London and Edinburgh with process number  
1");  
        Console.WriteLine("Redistribute the OSFP routes into EIGRP using the following metrics of  
delay " + myRouter.getDelay() + " and bandwidth " + myRouter.getBand());  
        Console.WriteLine("Redistribute the EIGRP routes into OSPF using default options.");  
    }  
}  
  
} //namespace
```

Appendix E – Performance Monitor Control Program

```
using System;
using System.Diagnostics;
using System.IO;

namespace LogApp
{
    class captureData
    {
        private string strTaskName          =
@"C:\WINDOWS\System32\logman.exe";

        // String for starting the logman application
        private string strStartArgs          = @"start myLog" ;

        // String for stopping the logman application
        private string strStopArgs           = @"stop myLog" ;

        // String for creating counters
        private string strCreateCountrArgs   = @"create counter myLog -
config ../../logmanCfg.txt" ;

        // String for deleting counters
        private string strDeleteCountrArgs   = @"delete counter myLog" ;

        // The constructor for captureData
        public captureData()
        {
            Process proc = new Process();
            proc.StartInfo.FileName = strTaskName ; // process we want to start

            proc.StartInfo.Arguments = strCreateCountrArgs; // argument
associated to the process

            proc.StartInfo.UseShellExecute = false;
```

```
        proc.StartInfo.RedirectStandardOutput = true; // we do NOT open
another console window
        string o = "" ; // variable where the output from the process is stored.
        try
        {
            proc.Start(); // start the process
            o = proc.StandardOutput.ReadToEnd(); // we capture the
output from the process
            proc.WaitForExit(); // we wait for the process to end
        }
        catch (Exception eee)
        {
            Console.WriteLine("Error while performing process: {0} ",
eee.Message);
        }
    }
}
```

```
    //The deconstructor for the class captureData
    // Tidy up what we created. IE the counter
    ~captureData()
    {
        Process proc = new Process();
        proc.StartInfo.FileName = strTaskName ; // process we want to start

        proc.StartInfo.Arguments = strDeleteCountrArgs; // argument
associated to the process
        proc.StartInfo.UseShellExecute = false;
        proc.StartInfo.RedirectStandardOutput = true; // we do NOT open
another console window
        string o = "" ; // variable where the output from the process is stored.
        try
        {
            proc.Start(); // start the process
            o = proc.StandardOutput.ReadToEnd(); // we capture the
output from the process
            proc.WaitForExit(); // we wait for the process to end
        }
    }
}
```

```
        catch (Exception eee)
        {
            Console.WriteLine("Error while performing process: {0} ",
eee.Message);
        }
    }

    // Process to start capturing the performance counter data
    // Written by David Mcluskie with additonal code taken from Lionel Saliou
    public void startCapture()
    {
        Process proc = new Process();
        proc.StartInfo.FileName = strTaskName ; // process we want to start

        proc.StartInfo.Arguments = strStartArgs; // argument associated to the
process
        proc.StartInfo.UseShellExecute = false;
        proc.StartInfo.RedirectStandardOutput = true; // we do NOT open
another console window
        string o = "" ; // variable where the output from the process is stored.
        try
        {
            proc.Start(); // start the process
            o = proc.StandardOutput.ReadToEnd(); // we capture the
output from the process

            Console.WriteLine("Logging Starting");
            //proc.WaitForExit(); // we wait for the process to end
        }
        catch (Exception eee)
        {
            Console.WriteLine("Error while performing process: {0} ",
eee.Message);
        }
    }

    // Process to stop capturing the performance counter data
    // Written by David Mcluskie with additonal code taken from Lionel Saliou
```

```
public void stopCapture()
{
    Process proc = new Process();
    proc.StartInfo.FileName = strTaskName ; // process we want to start
    proc.StartInfo.Arguments = strStopArgs; // argument associated to the
process
    proc.StartInfo.UseShellExecute = false;
    proc.StartInfo.RedirectStandardOutput = true; // we do NOT open
another console window
    string o = "" ; // variable where the output from the process is stored.
    try
    {
        proc.Start(); // start the process
        o = proc.StandardOutput.ReadToEnd(); // we capture the
output from the process
        Console.WriteLine("Logging Stopping");
        //proc.WaitForExit(); // we wait for the process to end
    }
    catch (Exception eee)
    {
        Console.WriteLine("Error while performing process: {0} ",
eee.Message);
    }
}
// Process to process the performance counter data
// Written by David Mcluskie with additional code taken from Lionel Saliou
public void processCapture()
{
    StreamReader reader = new StreamReader(@"C:\PerfLogs\windows-
monitor.csv");
    StreamWriter sw = new StreamWriter(@"./out.csv"); //Name of file
that the processed data will be saved to
    string text = ""; // Used to store result of the string split operation
    int numIncrements = 0 ; //Used to keep track of the number of 5 second
increments
    int numMins = 1; // Keeps track of the number of minutes processed

    // Get the column headings from the first line of the csv file
```

```
text = reader.ReadLine();
string [] strHeaders = text.Split("\");

// Writes the headings to the csv file
sw.WriteLine("Minute" + "," +
    (strHeaders[3]) + "," +
    (strHeaders[5]) + "," +
    (strHeaders[7]) + ",");

double[] myAvg = new double[10]; // Used to store the average
calculations

//
of each performance counter

while ((text = reader.ReadLine() ) != null)
{
    Console.WriteLine(text);
    // we split the string with comas '\'
    // and we place ALL the elements in an array of strings
    string [] elements = text.Split("\");
    Console.WriteLine("Value      read      in      is      "      +
Convert.ToDouble(elements[3]));

    // Process the elements into the element array. First meaningful
    // value is found at array position 3, 5, 7 and so on.
    // Best to place this in a separate function
    int position = 0;
    for (int i=3; i < elements.GetLength(0); i=i+2)
    {
        // Sometimes we get a null string in the reading. This
        // an error from being generated when the conversion
        // value of 0 is inserted in place of the null value
        if (elements[i] == " ")
            elements[i] = "0";
```

```

        myAvg[position] = myAvg[position] +
Convert.ToDouble(elements[i]); // Keeps a running total of the average
        position++;
    }

    numIncrements++;

    // Will only write average whole minutes to file
    if ( numIncrements == 12)
    {
        sw.WriteLine(numMins + "," + (myAvg[0] / numIncrements) + "," +
            (myAvg[1] / numIncrements) + "," +
            (myAvg[2] / numIncrements));
        // Clears the array and resets variables for the next
minute
        Array.Clear(myAvg, myAvg.GetLowerBound(0),
myAvg.GetLength(0));

        numMins++;
        numIncrements = 0;
    }
}
reader.Close();
sw.Close();
}
}

class Demo
{
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main(string[] args)
    {
        captureData cTheLogger = new captureData();
        string strUserChoice;
```



```
do
{
    Console.WriteLine("Please enter option (1-5)");
    Console.WriteLine("1. Start Logging");
    Console.WriteLine("2. Stop Logging");
    Console.WriteLine("3. Process Log");
    Console.WriteLine("5. Exit Proggie");
    strUserChoice = Console.ReadLine();

    switch(strUserChoice)
    {
        case "1" :
            cTheLogger.startCapture();
            break;
        case "2" :
            cTheLogger.stopCapture();
            break;
        case "3" :
            cTheLogger.processCapture();
            break;
        case "5" :
            Console.WriteLine("Goodbye cruel world");
            cTheLogger.stopCapture(); // Ensures that
logging is switched off. Don;t want to leave this running.
            break;
        default :
            Console.WriteLine("Invalid input. Please try
again");
            Console.ReadLine();
            break;
    }
}
while (strUserChoice != "5");
```

```
}  
}  
}
```

Appendix F – Project Plan

[illegible]

Sample Project Plan

Appendix G – Example Project Diary

NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: David McLuskie

Supervisor: Professor Bill Buchanan

Date: 05/08/08

Last diary date:

Objectives:

Have written literature review ready

Progress:

Initial literature review ready for feedback

Supervisor's Comments:

Apply Bloom to Cisco/network courses
Review Cisco architecture
Investigate memory buffering for stressing
Footprint evaluation (memory, processing)

1. John, G. and T. Scott, *A conceptual overview of the virtual networking laboratory*, in *Proceedings of the 8th ACM SIGITE conference on Information technology education*. 2007, ACM: Destin, Florida, USA.
2. Chengcheng, L., *Blur the boundary between the virtual and the real*. J. Comput. Small Coll., 2009. **24**(3): p. 39-45.
3. Colin, G.J. and F. Ursula, *Is Bloom's taxonomy appropriate for computer science?*, in *Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006*. 2006, ACM: Uppsala, Sweden.
4. Ursula, F., et al., *Developing a computer science-specific learning taxonomy*. SIGCSE Bull., 2007. **39**(4): p. 152-170.
5. Simon, G., et al., *Suggestions for graduated exposure to programming concepts using fading worked examples*, in *Proceedings of the third international workshop on Computing education research*. 2007, ACM: Atlanta, Georgia, USA.
6. Anon. *Program Overview*. Program Overview 2008 [cited 2008 24/09/08]; Available from: <http://www.cisco.com/web/learning/netacad/index.html>.
7. Boson. *Boson Software: Exam Preparation*. 2008 [cited 2008 29/07/08]; Available from: www.boson.com.
8. Buchanan, B. *Networksims.com Home Page: The Best Cisco Simulator (Emulator) in the World*. 2008 [cited 2008 29/07/08]; Available from: www.networksims.com.
9. vmware. *VMware: Virtualization via Hypervisor, Virtual Machine & Server Consolidation*. 2008 [cited 2008 01/07/08]; Available from: www.vmware.com.
10. ANON. *Dynagen*. 2008 [cited 2008 30/07/08]; Dynagen Website]. Available from: <http://dynagen.org/>.
11. Rahul, A., et al., *Designing an adaptive learning module to teach software testing*. SIGCSE Bull., 2006. **38**(1): p. 259-263.
12. Anon. *Microsoft Certifications Overview*. 2008 [cited 2008 25/09/08]; Available from: <http://www.microsoft.com/learning/mcp/default.mspx>.
13. Anon. *IT Certification and Career Paths*. 2008 [cited 2008 25/09/08]; Available from: http://www.cisco.com/web/learning/le3/learning_career_certifications_and_learning_paths_home.html.
14. Anon. *Oracle Certification Program*. 2008 [cited 2008 25/09/08]; Available from: http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getpage?page_id=39&p_org_id=28&lang=US.
15. Cort, S. and M. John, *The computer curriculum and certification: a proposal*. J. Comput. Small Coll., 2005. **20**(4): p. 84-91.
16. Chengcheng, L., et al., *A practical study on networking equipment emulation*. J. Comput. Small Coll., 2008. **24**(2): p. 137-143.
17. Laurent, D., et al., *IREEL: remote experimentation with real protocols and applications over an emulated network*. SIGCSE Bull., 2007. **39**(2): p. 92-96.

18. Laverell, W.D., F. Zongming, and N.G. James, *Isn't it time you had an emulab?*, in *Proceedings of the 39th SIGCSE technical symposium on Computer science education*. 2008, ACM: Portland, OR, USA.
19. Kayo, F. and C. Henri, *Speed and accuracy of network simulation in the SimGrid framework*, in *Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*. 2007, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering): Nantes, France.
20. Toshiyuki, M., C. Ken-ichi, and S. Yoichi, *StarBED and SpringOS: large-scale general purpose network testbed and supporting software*, in *Proceedings of the 1st international conference on Performance evaluation methodologies and tools*. 2006, ACM: Pisa, Italy.
21. Marinho, P.B., F. Giovani, and H.M. Hisham, *Bridging the Gap between Simulation and Experimental Evaluation in Computer Networks*, in *Proceedings of the 39th annual Symposium on Simulation*. 2006, IEEE Computer Society.
22. Andy, B., et al., *In VINI veritas: realistic and controlled network experimentation*, in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*. 2006, ACM: Pisa, Italy.
23. Karen, A., *Making CS0 fun: an active learning approach using toys, games and Alice*. J. Comput. Small Coll., 2008. **23**(3): p. 98-105.
24. Simon, C. and B. David, *The virtualization reality*. Queue, 2007. **4**(10): p. 34-41.
25. George, D.H., *An overview of Virtual Machine (VM) technology and its implementation in I.T. student labs at Utah Valley State College*. J. Comput. Small Coll., 2008. **23**(6): p. 203-212.
26. Ulrich, D., *The cost of virtualization*. Queue, 2008. **6**(1): p. 28-35.
27. Keith, A. and A. Ole, *A comparison of software and hardware techniques for x86 virtualization*. SIGOPS Oper. Syst. Rev., 2006. **40**(5): p. 2-13.
28. Anon. *Welcom to xen.org*. 2008 [cited 2008 25/09/08]; Available from: <http://www.xen.org/>.
29. Alessio, G., et al., *March of the (virtual) machines: past, present, and future milestones in the adoption of virtualization in computing education*. J. Comput. Small Coll., 2008. **23**(5): p. 123-132.
30. Wei, H., et al., *A case for high performance computing with virtual machines*, in *Proceedings of the 20th annual international conference on Supercomputing*. 2006, ACM: Cairns, Queensland, Australia.
31. Sriram, G., et al., *Xen and co.: communication-aware CPU scheduling for consolidated xen-based hosting platforms*, in *Proceedings of the 3rd international conference on Virtual execution environments*. 2007, ACM: San Diego, California, USA.
32. Padma, A., et al., *Characterization \& analysis of a server consolidation benchmark*, in *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*. 2008, ACM: Seattle, WA, USA.
33. Tom, K., *Meet the virts*. Queue, 2008. **6**(1): p. 14-18.
34. Liana, F. and S. Malgorzata, *Duality of virtualization: simplification and complexity*. SIGOPS Oper. Syst. Rev., 2008. **42**(1): p. 96-97.

35. Stephen, S., et al., *Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors*, in *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*. 2007, ACM: Lisbon, Portugal.
36. William I. Bullers, Jr., B. Stephen, and F.S. Alessandro, *Virtual machines - an idea whose time has returned: application to network, security, and database courses*. SIGCSE Bull., 2006. **38**(1): p. 102-106.
37. Alessio, G., et al., *Enabling new pedagogies in operating systems and networking courses with state of the art open source kernel and virtualization technologies*. J. Comput. Small Coll., 2008. **23**(5): p. 189-198.
38. Alessio, G., et al., *The role of virtualization in computing education*, in *Proceedings of the 39th SIGCSE technical symposium on Computer science education*. 2008, ACM: Portland, OR, USA.
39. Jisoo, Y. and G.S. Kang, *Using hypervisor to provide data secrecy for user applications on a per-page basis*, in *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*. 2008, ACM: Seattle, WA, USA.
40. Sherry, Y.C. and L. Xiaohui, *An Integrated Approach for Modeling Learning Patterns of Students in Web-Based Instruction: A Cognitive Style Perspective*. ACM Trans. Comput.-Hum. Interact., 2008. **15**(1): p. 1-28.
41. Michael, E.C. and B. Jens, *Instructional design of a programming course: a learning theoretic approach*, in *Proceedings of the third international workshop on Computing education research*. 2007, ACM: Atlanta, Georgia, USA.
42. Linxiao, M., et al., *Using cognitive conflict and visualisation to improve mental models held by novice programmers*, in *Proceedings of the 39th SIGCSE technical symposium on Computer science education*. 2008, ACM: Portland, OR, USA.
43. Christopher, W.S., M. Bill, and H.S. RoxAnn, *Bloom's taxonomy revisited: specifying assessable learning objectives in computer science*. SIGCSE Bull., 2008. **40**(1): p. 261-265.
44. Paolo, A.G.S. and M.P. Scott, *A collection of kinesthetic learning activities for a course on distributed computing: ACM SIGACT news distributed computing column 26*. SIGACT News, 2007. **38**(2): p. 56-74.
45. Mario, et al., *Enhancing software engineering education: a creative approach*, in *Proceedings of the 2008 international workshop on Software Engineering in east and south europe*. 2008, ACM: Leipzig, Germany.
46. Dawn, M. and F.D. Leo, *Developing collaborative skills early in the CS curriculum in a laboratory environment*. SIGCSE Bull., 2006. **38**(1): p. 138-142.
47. Donna, T. and R. Paul, *Collaborative learning: towards a solution for novice programmers*, in *Proceedings of the tenth conference on Australasian computing education - Volume 78*. 2008, Australian Computer Society, Inc.: Wollongong, NSW, Australia.
48. Anon. *GNS3 Home*. 2008 [cited 2008 10/06/08]; Available from: <http://www.gns3.net/>.
49. Marquardt, T. *ASP.NET Performance Monitoring, and When to Alert Administrators*. 2003 [cited 2008 05/01/08]; Available from: <http://msdn2.microsoft.com/en-us/library/ms972959.aspx>.

50. Tulloch, M. *Key Performance Monitor Counters*. 2005 [cited 2008 05/01/08]; Available from: http://www.windowsnetworking.com/articles_tutorials/Key-Performance-Monitor-Counters.html.
51. Anon. *Important Counters for Web Testing*. 2008 [cited 2008 05/01/08]; Available from: [http://msdn2.microsoft.com/en-us/library/aa287688\(vs.71\).aspx](http://msdn2.microsoft.com/en-us/library/aa287688(vs.71).aspx).