Sébastien KOTZINE

Submitted in partial fulfilment of the Requirements of Napier University For the degree of

Bachelor of Science with Honours in Network Computing

School of Computing

May 2002
Authorship Declaration

I, Sébastien KOTZINE, 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 plagiarism.

Signed:

Date:

Matriculation number: 00198226
Abstract

The aim of the project was to develop a router emulator including configuration. A fully implemented package can be used to enhance the learning of router programming. The goals of the project were to deliver the documents and produce a fully working program that interprets and performs most of the IOS commands in order to be able to perform a full router configuration and improve router-programming skills. In addition, the software will include specifics function (Open configuration files, Save Configuration files, View the feedback of the actual configuration). A good understanding of how the IOS commands are structured and the behaviour of a physical router are really important in order to build a software as close as possible of the reality.

A router is a networking component, which fits at a layer 3 of the OSI model. Routers allow data to be passed from one network to another based on the destination network address of a data packet. In order to forward data traffic a router checks the data address, compares with its tables of IP addresses and forwards the packets to the intended networking device. Routers are complex devices, which require commands to program the operation of the router; the configuration also needs to be implemented within the emulator. An emulator is as accurately as possible, a representation of the reality. Unfortunately, expensive hardware is required in order to set-up router configuration. An emulator is then, a representation of the real router environment, which can interpret commands and interpret configurations. Routers include a main IOS (Internetwork Operating System) to interpret these commands. The software is intended for people who want to learn or improve their skills in router programming using the produced emulator.

The report shows how software can be developed which emulates IOS (Internetwork operating system) commands, and is able to access to other routers on the same network and affect the IP addresses and subnet masks. Another aspect of the emulator is that it includes a graphical representation of the commands typed, in order to provide a better feedback of the configuration performed and enhance router-programming skills. The developed software also allows router configuration information, and the command sequence used, to be saved, and recalled at a later time. To enhance system compatibility the router emulator uses Java, and Java Swing for the graphical user interface.

Most of the IOS commands are able to be interpreted and emulated in order to affect the IP addresses and subnet masks, affect the different passwords inherent to router programming, resolve hostnames, select and implement the routing protocol, and so on. Java has been selected from other object-oriented programming languages as it provides access to an excellent GUI (Graphical User Interfaces), using Java Swing components. Furthermore Java is powerful and is platform independent, which is a great advantage for educational purpose as the emulator is able to run over a wide range of platforms.

An important element in the development of the software is the structured design technique used. The report reviews several of the main types, including waterfall model, evolutionary development, and Boehm’s spiral model.

The technique used in the development is the Boehm’s spiral model for its good visibility of the process, the possibility of adapting the phases according to the requirements and its allows the possibility of creating the prototype and re-cycling in order to redefine the requirements, the development, the test, until the final software is developed.

The report also studies the behaviour of a real router (based on observation and analysis of the Cisco laboratory in Napier University using five Cisco Routers) and evaluators some existing emulators. Along with this user trails have been carried out on the operation of the
software, and these are used to appraise the merits of the package as opposed to other similar packages.

Finally the report concludes with a comparison with different emulators currently existing on the market and an evaluation by intended users, which have agreed that the dynamic graphical representation of the configuration and the extra facilities provided help router programming skills a great deal. The software reach the objectives stated in the introduction were to create an emulator of a router which implements major I.O.S. commands included into real router configuration. A future work might include a wider evaluation by intended users using H.C.I. tools. A full set of IOS commands could be implemented for this software in order to produce a powerful emulator fully compliant with the latest versions of IOS produced by the different router manufacturers. As the networking and more generally computing area is evolving so fast that the software needs to adapt to these constant-moving technologies by the different improvement and changes made in software and hardware architecture.
# Table of Contents

## Abstract

## 1 Introduction

1.1 Aims

1.2 Objectives

1.3 Deliverables

1.4 Report structure

1.5 Background

## 2 Theory

2.1 Introduction

2.2 TCP/IP

2.2.1 Historical factors relating to the popularity TCP / IP

2.2.2 TCP 14

2.2.3 TCP Header

2.2.4 TCP ports

2.2.5 TCP three-way handshake and window size.

2.3 UDP (User Datagram Protocol)

2.4 The OSI and TCP/IP models

2.5 TCP & OSI layers

2.6 IP Addresses

2.7 Subnet mask

2.7.1 IP version 6 (IPv6) / IPng

2.8 Routing protocols

2.8.1 RIP (Routing Information Protocol)

2.8.2 IGRP / EIGRP

2.8.3 OSPF

2.9 Router Fundamentals

2.10 Router memory

2.10.1 CLI

2.11 CLI prompts

2.12 IOS

2.13 Conclusion

## 3 Design/Modelling

3.1 Introduction

3.2 Requirements

3.3 Software engineering methods

3.3.1 Waterfall Model

3.3.2 Evolutionary development

3.3.3 Boehm’s spiral model

3.3.4 Risk analysis phase

3.4 High-level design

3.4.1 Modelling the software

3.4.2 Global Input / Output of the system

3.5 Components design analysis

3.5.1 Main class

3.5.2 Exit form

3.5.3 Initialisations (Variables; Graphic environment)

3.5.4 Java Swing Graphic environment

3.5.5 Key Event Listener

3.5.6 CLI Prompts

3.5.7 Interpretation of Commands typed

3.5.8 Feedback of the Command Typed and Variables Affected after the Interpretation of the commands
3.5.9 Java Swing Menu and Action Performed associated with the menu

3.6 Conclusion

4 Implementation

4.1 Introduction
4.2 Programming language chosen: Java
4.3 Implementation of the router commands
   4.3.1 Libraries declaration
   4.3.2 Main types of declaration
4.4 Exit method
4.5 Key event listener
   4.5.1 Displaying CLI prompts
   4.5.2 Interpretation of the command typed
   4.5.3 Example of an emulated IOS command
   4.5.4 Executive password verification
   4.5.5 Java Swing menu: (Open & save files configuration)
4.6 Conclusion

5 Tests/Evaluation

5.1 Introduction
5.2 Test based on the interpretation of real router configurations
5.3 Test performed using the Advanced Distributed Systems tutorial on router configuration
5.4 Java platform independent?
5.5 Users evaluation
5.6 Comparison with other router emulators.
5.7 Conclusion

6 Conclusion

6.1 Appraisal of Achievement
6.2 Suggestions for future work.
Acknowledgements

I would like to thank Dr. Bill Buchanan, my project supervisor at Napier University, for his constant help and advice during the development of my Project. I would also like to thank Pr. Peter Ross for his help.
1 Introduction

1.1 Aims

The aim of this project is to produce a package of an emulator which will enhance the learning of router programming; the intended emulator will be able to interpret major IOS (Internetwork Operation System) commands and provide feedback to the user in order to build a router configuration and therefore gain experience of router programming.

A router, as stated within the abstract part of this report, is a networking component, which fits at a layer three of the OSI model. Routers are considered as intelligent devices as they route information in a more efficient way using layer three of the OSI model. Hubs for example, operate at layer one of the OSI model; they only regenerate the signal and broadcast it to all of their ports. Switches & bridges (a bridge is a multi port bridge) using layer two of the OSI model, filter data traffic on a network according to the MAC addresses, they are consequently better than hubs as they make differences of the packets forwarded but routers work in a more efficient way, than the previous devices, as they make their decisions on a group of IP network addresses.

Routers are complex devices, which required commands to program the operation of the router and consequently build the configuration. As expensive hardware is required in order to set-up routers configurations the aim of the package produced will be an emulator which give a representation of the real router environment which can interpret commands and interpret these. Routers include a main IOS (Internetwork Operating System) to interpret these commands. The software will be able to interpret majors IOS commands and interpret them.

The system is intended to have an educational purpose, for a user that wants to gain knowledge of routers programming without the use of a consequent and expensive hardware.

1.2 Objectives

Create an emulator of a router which implements major IOS (Internetwork operating system) commands, such as:

- Resolve the hostname
- Configure the console, auxiliary telnet and executive passwords
- Affect the IP addresses and subnet masks on the different ports of the router
- Configure routing tables
- Configure hostname tables.

Furthermore the emulator produced should be able to save the current router configuration and load it, to perform changes or view the configuration previously programmed.

Another aspect of the emulator can include a graphical representation of the commands typed, in order to improve the router programming skills and provide a better feedback than the real router of the IOS commands entered in order to enhance IOS router configuration skills.

1.3 Deliverables

A software application programmed in Java language used to enhance the learning of router programming. Majors IOS commands will be able to be interpreted and emulated in order to build a router configuration.
1.4 Report structure

This part of the report will give details of the different parts of this dissertation. A major aspect of this project is based on theory in order to be able to implement a robust emulator which behaves as a real router, and is able to adapt to a new IOS or new router architectures, in other words be evolutionary to a new released of an IOS router operating system.

The theory will be focused on the networking theory focused around the TCP/IP (Transmission Control Protocol/Internet Protocol), which are the two major widely used protocols mainly due to the impressive Internet expansion. A description of these two protocols will be highlighted in order to outline a global understanding of the majors protocols used.

After this, a presentation of the OSI model will be presented, with the aim of giving details of the seven layers in order to introduce IP addresses and subnet mask according to RFC (Request for Comments) related.

A discussion will be extended on the new standard IPv6 for a possible extension of the router emulator. The discussion on the new standard of IP addresses (also called IPng where ng stands for next generation) will be based on different recent papers published.

In order to route data packets, router use routing protocols to determine the path that routed protocols follow in order to reach their destinations. The different routing protocols actually existing will be highlighted such as RIP, IGRP, EIGRP and OSPF, the research of these protocols will be based on RFCs.

The next part of the research will be based on the router fundamentals, which will highlight the router boot, memory, router connection, passwords, router prompts and a comment on IOS. A partial conclusion will be given in order to summarise the learning outcomes of the theory part.

The design / modelling part of this report will describe the specification and analysis of the router emulator. Different software engineering methods will be detailed as the router emulator is programmed using an Object-oriented programming and the justification for the method chosen will be describing the steps of the Object-oriented Programming used to develop this application.

Next, the high level design of the emulator will be given highlighting the global input/output of the software. Finally, a more specific analysis based on the component design will be given. A partial conclusion will summarise the ending outcomes realised by the design and modelling part.
The next part will highlight the implementation of the program developed using Java, this part will detailed the general conception of the application including the way of interpreting the IOS commands typed plus the way of implementing the different router prompts.

The implementation of the different IOS commands emulated by the application will be also given. The strength of the programming language used: Java will be detailed. The different functions implemented in this program, intended to enhance the router programming skills by providing a better feedback and the configuration files facilities will be given in this report such as the Java code associated. A partial conclusion will be given highlighting the different points viewed during the implementation part.

The tests part will highlight the different tests performed on the router emulator in order to demonstrate that it does fit with the requirements. An evaluation will be performed by typical intended users.

A comparison with other emulators available in the market will be made according to different factors (Usability, ease to use, robustness, facilities offered, rapidity, and price). A conclusion will then give a summary of the testing part of this report.

The last part of this report will be focused on a general conclusion in order to highlight all the point viewed during this report. An appraisal of achievement of the software package will be given and suggestions for future work will be presented. The following details the report:

I) Introduction

- Aims
- Objectives
- Report Structure
- Background

II) Theory

- Introduction
- Networking Theory
  - TCP/IP (Transmission Control Protocol/Internet Protocol)
  - OSI model with the seven layers associated
  - IP addresses
  - subnet mask
  - new standard IPv6 (or IPng)
  - Routing protocols
    - RIP
    - IGRP
    - E.IGRP
    - OSPF
  - Router fundamentals
    - router boot
    - memory
    - router connection
    - passwords
    - router prompts
    - IOS
- Partial conclusion

III) design / modelling

- Introduction
- Software engineering methods
Details of the different approaches

- Object-oriented Programming – The software engineering method chosen
- High level design (Input/Output of the program)
- Analysis on the component design
- Partial Conclusion

IV) Implementation of the program

- Introduction
- Strengths of the programming language used: Java
- Interpreting the IOS commands
- Different router prompts
- IOS commands emulated
- Further implementation to enhance the router programming skills
  - Open file configuration
  - Save file configuration
  - Graphical representation of the configuration
- Partial Conclusion

V) Tests

- Different tests performed on the router emulator
- Typical user evaluation
- A comparison with other emulators
- Partial conclusion

VI) Conclusions

- General conclusion
- An appraisal of achievement
- Suggestions for future work

Table 1: Summary of the Report Structure

1.5 Background

The idea of the router configuration emulation was given by my current supervisor, Dr. Bill Buchanan, who has included it, as part of an honours project on his web site. As I am really interested on the networking field, this project appears to fully fit my course and enable me to apply practical knowledge taught during my degree. I decided to contact Dr. Bill Buchanan for further questions about the project; my supervisor has agreed to allow me to carry on with this project and share his thoughts about possible development of the project.

The aim of this project is to deliver an emulator which could enhance the learning of router programming. In order to perform router configurations, IOS language is required to be interpreted by these devices in order to program them. One of the important parts of the research undertaken has been focused on the IOS (Internetwork Operating System) as will be shown during this report.
In the introduction part it has been seen that routers are complex networking devices that forward data traffic by checking the data address of the packets sent, the router compares with its tables of IP addresses and forward the packets to the intended networking device. Cisco academy is running on Napier University location (Merchiston) and it is composed of a five routers configuration. The Cisco facilities available in Napier University, with the help of some existing emulators have been really useful designing the software application.

The router emulator is entirely programmed in Java, mainly as its ability provides a range of Graphical User Interfaces specially the Java Swing components which provide a good and professional look and feel of the application. The strength of this language will be given during this report.

For the development environment Forte for java has been used as this is a powerful tool entirely written in Java beans, which has been learned in order to program the router configuration. The advantages of forte are based on its powerful integrated components to be added and its multiple options and parameters which make it professional tool to be used when developing Java applications. Furthermore the code generated is not over crowded with useless information as is often the case with some professional tools, the code is easy to compile for another application and can launched using a simple java virtual machine and notepad for editing.

Some products are already been developed on the market, a critical appraisal of these tools will be developed and compared with the router emulator application developed for this project.

Finally, further development of the product has been implemented such as the possibility of the load and save configuration, as it would be possible to save the actual configuration and load it later for more implementations.

A graphical representation of the commands entered by the users appears to be a very useful extension as the emulator is intended for educational purposes, it could provide a faster understanding of the different commands by relying on a graphical view of the configuration. After the introduction part the theory background related with this project will be examined in the next part of this report.

![Route Configuration](image-url)

*Figure 2: Graphical representation of the actual router configuration*
2 Theory

2.1 Introduction

This part of the report will highlight the theoretical background needed for the actual implementation of the Router emulator and possible extensions will be highlighted.

An introduction to TCP/IP will be given in order to define the context of the application currently developed for this project.

The theory will be based upon academic papers written in this field and request for comments for the two widely used protocols (TCP and IP).

A model of the OSI model will be described in order to introduce a general background of networking essential for the context of the project. An overview of the IP addressing (IP version 4) will be viewed including subnet mask.

A general overview will be given of IPv6 according to recent papers published in order to provide general knowledge in the next generation of IP addressing, which is considered as a very important step in the networking field. This research will provide guidelines for a suggestion for future work.

Router are using routed protocols (such as IP, IPX, AppleTalk, and so on) to route information. Routing protocols determine the path that routed protocols follow in order to reach their destinations. An overview of the major routing protocols will be investigated.

A study of router fundamentals will be described in the next part, including booting sequences, type of memory used, router connections, passwords prompts and IOS commands. Java has been selected for the language used for the implementation of this project. All software development projects are managed according to software engineering methods that will be described in this report. Finally the strength of the programming language chosen: Java will be given and a conclusion will highlight the theory part of this report.

2.2 TCP/IP

2.2.1 Historical factors relating to the popularity TCP/IP

TCP / IP are incontestably the most used protocols due to the growth explosion of the internet and the World Wide Web where the protocols are mainly used. The development of Internet is the result of the Department of Defence (DOD).

At the end of the 60’s the DOD realised that a common protocol was needed to overcome with a lack of communications due to the wide variety of machines and proprietary protocols that did not allow communication between all these computers. A protocol that allows communications within this wide range of machines was needed regardless of the type of hardware machine or operating system used.

At the same time the ARPAnet (The Advanced Research Projects Agency network) which was a part of the DOD was looking for ways to improve communications between computers and provide a redundant path of connectivity between machines.

There is a military direct application behind the development of TCP / IP, in case of an eventual nuclear attack; if one of the points in the network fails the communications within the whole network would not be compromised; that would help to maintain communications in case of an attack.

The first protocol developed was Network Control Protocol (NCP), replaced later by TCP / IP due to the increase of the network traffic. One of the success of the adoption of TCP / IP is
due to the adoption by the NSF (National Science Foundation) which needed to create a network dedicated to the research environment, the foundation adopted ARPAnet protocol which then lead to the development of the Internet and resulted in the explosion of the TCP / IP that we know now.

After exposing the factors that lead TCP and IP protocols, due to the wide variety of platforms and different operating systems running over the world and the impressive growth of the World Wide Web and the Internet, to the actual world wide use, a discussion detailing TCP and IP protocols will be given. [B8]

2.2.2 TCP

According to the RFC 793 created by DARPA, giving specification for the TCP protocol, TCP was intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks.

The TCP protocol occurs at the transport layer of the OSI model, which will be defined later on in this chapter. The aim of TCP is to provide connections TCP provides a connection-oriented when creating a remote connection with one or more hosts and is highly reliable. Furthermore TCP provides flow and error control. [5]

TCP as described before forms part of the transport layer of the OSI model. We can note at this point that routers have nothing to do with the TCP, transport layer, as the transport layer is above the network layer that the router use to forward data packets, they then deliver IP datagram’s. Nevertheless, the IP layer delivers information of control will be exploited by the transport layer of the host machine.

TCP provides multiple advantages such as:

- Connection-oriented
- Reliable full-duplex data transmission
- Test of the received data
- Flow control
- Reassembles messages at the destination station
- Reassembles messages from incoming segments
- Timing function in the case that the receiver did not send back an acknowledgement the data is retransmitted
- Divides outgoing messages into segments

2.2.3 TCP Header

Figure 3 shows the header of TCP, which demonstrates the complexity and the important functions that TCP does provide. The description of each the fields is detailed below:

- **Source Port** (16 bits). The numerical value is indicating the source port.
- **Destination Port** (16 bits). The numerical value is indicating the destination port
- **Sequence number**. The sequence number of the first octet ensure correct sequencing of the arriving data
- **Acknowledgement** (32 bits). Next sequence number that the sender of the segment is expecting to receive.
- **Header length** or HLEN (4 bits). Contain the length of the TCP header
- **Reserved** (6 bits). Reserved for future functions; all bits are equals to 0
- **Control Bits:**
  - URG: If the flag is set to 1 it is indicating an urgent value
  - ACK: If the bit is set to 1 the field number need to be taken under account
  - PSH: Push the data into the destination of the necessary application,
RST: Reset the connection if the bit value is set to 1
SYN: Synchronisation is made and will indicate the beginning of a connection
FIN: End if the value is set up to 1, client has no more information to transmit. The connection is then closed.
- **Window** (16 bits). Used to control data flow this field determine the number of octets that the sender wants to accept.
- **Checksum** (16 bits). This field holds the checksum of the header and data fields.
- **Urgent Pointer** (16 bits). This shows the value of the URG pointer in the form of a positive sequence indicating the beginning of an urgent information.
- **Options**. Define a small group of options such as the maximum TCP length.
- **Padding**. Adds a sequence of binary 0 to ensure that the length is 32 bits.
- **Data length**. Contains the data that will be send (Variable) set at the upper protocol layer.

![TCP header](image)

Figure 3: TCP header

### 2.2.4 TCP ports

TCP (as does UDP) uses ports numbers to keep track of different information exchanges in order to pass information to the upper layers. Three categories of ports can be noted reserved for specific applications (FTP: port 21; HTTP: port 80; SMTP: (mail) port 25); Telnet port 23…), some ports are used for commercial applications, and finally above 1023 the ports are unregulated.

<table>
<thead>
<tr>
<th>Port Range</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Below 255</td>
<td>Public Applications</td>
</tr>
<tr>
<td>255-1023</td>
<td>Commercial application</td>
</tr>
<tr>
<td>Above 1023</td>
<td>Unregulated</td>
</tr>
</tbody>
</table>

Table 2: TCP ports numbers

### 2.2.5 TCP three-way handshake and window size.

As we have seen previously TCP is a connection oriented protocol. It exchanges data using a method called **three-way handshake**. The three-way handshake method is a method using synchronisation flags between the machines that want to exchange information in order to ensure the link between them.

The window size determines the amount of data that the source can transmit before an acknowledgement. This window size ensures the flow control of data that is transmitted. The window size can be adjusted in order to increase the flow of information or on the contrary the widow size can be reduce in order to ensure a reliable transmission.

According to the authors of “WINTRAC: a TCP window adjustment scheme for bandwidth management” [1], who explain in their recent paper that there is a difference between the window size and the bandwidth of the network which can cause underutilisation.
of the bandwidth. The solution presented in this paper is the usage of a recursive algorithm used to control the rate. The idea that the authors came up with, is that they could match the network load rate with the window that the receiver is willing to accept. The authors demonstrate their theory with simulation of the implemented algorithm which results in “significant improvements in delay, packet loss rates, fairness in the distribution of the maximum achievable window size, and link utilization.” This clearly demonstrates that TCP is widely used and constantly, in respect of the original specifications, improved.

All the functions available in the TCP header demonstrate that TCP presents a lot of advantages and an amount of research is carried on in the networking field, all those elements explained its wide usage. [4] TCP is normally associated with IP which is a layer 3 protocol of the OSI model. TCP/IP is a combination of two individual protocols. The Internet is then based on the layer 3/4 protocol represented by TCP/IP.

![Three ways handshake for oriented connection mode](image)

**Figure 4:** Three ways handshake and windows size in connection-oriented protocol: TCP

### 2.3 UDP (User Datagram Protocol)

Routers allow forwarding data packets using IP addresses; therefore they perform it at layer three of the OSI model, TCP is not the only protocols standing at layer 4 of the OSI model; UDP is the other protocol. User Datagram Protocol (UDP) is the connectionless transport protocol. UDP exchanges datagrams, without acknowledgments or guaranteed delivery; furthermore UDP does not handles errors or retransmission. UDP can be used on the top of the IP addresses layer.

The UDP will not be examined in this report as the TCP is sufficient here in order to introduce one of the protocols set by this next layer (Layer 4 Transport) of the OSI model, that is to introduce IP (layer 3 of the OSI model) addresses theory and sub-netting mask.

The application does not emulate the behaviour of the layer 4 of the OSI model as real Router operate at layer 3 of the OSI model. An overview of the OSI model will be viewed in order to give an overview of an essential networking model and introduce IP addresses and subnet masks.

### 2.4 The OSI and TCP/IP models

In the 80’s a wide variety of hardware and software were present in the computing field. The I.S.O. (International Standards of Organisation) creates the OSI model to facilitate the interconnections between those different platforms which reach a common standard for intercommunications and facilitate the software developer by providing a set of guidelines. The OSI model will be detailed below giving an account of the seven different layers that the model uses.
Data are passed from the top layer of the OSI model (Application) from the sender to the bottom layer (Physical layer) and the other way round is done from the bottom layer to the top layer of the OSI model.

The OSI model will be given in correspondence of the TCP/IP model. The model presented here is a four layer model, this model is not the only one, according to the RFC 871 three layers are described (network layer, host to host layer and processing/application layer). As the four model layer is the most commonly used, that the one which will be compared to the OSI model below. RFC 871. RFC793

2.5 TCP & OSI layers

A brief description of each of the OSI layers are explained below:

**Layer 1: Physical Layer.** This layer is the bottom layer of the OSI model; this layer is responsible for converting numerical data into electrical signals and transmitting them using bit streaming and receiving incoming data stream by transforming them into electrical signal.

**Layer 2: Data Link Layer.** This layer is the second layer of the OSI model, it ensures the reliability of the data received and transmitted by adding extra data to the frame. The data frame is also responsible for detecting and correcting eventual errors during transmission. Sometimes the Data link layer is split into two different sub layers: (LLC: Logical link control responsible for encapsulation and MAC which is the other sub layer that deals with protocols that a host follows to have access to the physical media). The IEEE 802.3 appears to violate the OSI as the LLC is forming one layer with its own PDU (protocol data unit) and the Mac layer cross over the layer 2 / 1 standard, but the OSI is here to
Layer 3: Network Layer. This layer is responsible for establishing the route to be used between the host and destination computers. Router are operating at this layer of the OSI layer they route all data frames through the network according to the logical address.

Layer 4: Transport Layer. This layer is responsible for data flow control and handles errors. TCP and UDP are directly involved with this layer and they have been explained previously.

Layer 5: Session Layer. This layer is responsible for handling the flow of information between interconnected machines.

Layer 6: Presentation Layer. This layer is responsible for managing the way that data is encoded. This layer is related to the translation between different systems, for security encryption, decryption and responsible for data compression.

Layer 7: Application Layer. This layer is the top layer of the OSI model it provides interfacing between applications by providing network services to applications programs. The Router Configuration Emulation is a program which operates at this layer as well as various applications.

After studying the different layers of the OSI model it is time to state the different layers that compose the TCP/IP model. This description is assuming that the four layers model is used.

Network interface layer. This layer is also known as the data-link layer. This layer provides an interface between the physical media and the network. This layer is used to format data according to the network specifications and this layer is also responsible for errors control that occurs during within the interfacing of the physical media.

Internet layer. This layer is also known as the network Layer. This layer is responsible for the distribution of packets around the network. The router occurs at this layer using standard protocols; an overview of these protocols will be detailed later in this report.

Transport layer. This layer is responsible for ensuring the flow of data between two hosts, the control layer and the acknowledgment between interconnected networks. This layer is also used to provide interfacing between networking applications.

Application layer. This layer is responsible for handling the details of a particular application, offering file transfer and responding to Internet needs. Furthermore this layer does support network API( Application Programming Interfaces) which enable access to applications to the network.
the OSI model. This presentation was needed to introduce the transport layer and next the Network layer at which physical routers operates.

![Figure 6: A which layers of the OSI model does the application developed fit in, and a representation at which OSI layers routers operates.](image)

IP is one of the logical addresses used by the router to forward and make decision. The IP addresses are based on layer 3 of the OSI model this will examined in the next part.

### 2.6 IP Addresses

One of the router protocols will be examined: IP is a part of a routed protocols and it will be studied in this part. Some other routed protocols exist and are used by routers to forward information. Examples of other protocols used (AppleTalk, Novel Netware, Xerox Network System, DECnet). This part will present the format of IP addresses as they are the most widely used.

An IP address is composed of a 32 bit long, commonly written for ease of reading by four separate dots (4 octets in 8 bits each), usually known as a dotted decimal format. The IP affected are assigned to hosts that want to communicate within network or the Internet, often assigned by an I.S.P. (Internet Service Provider).

Three main classes are used in IP addressing, Class A, B, C. These classes are used to identify network and host. IPs addresses are received from ARIN (American Registry for Internet Numbers).

Two other classes exists such as Class D which is a class reserved for multicast. Finally, the class E has been reserved for experiment and future use. [6] RFC791 IP - RFC 894 IP

<table>
<thead>
<tr>
<th>Class A</th>
<th>Decimal range: 1-126</th>
<th>First Octet (binary)</th>
<th>Network First Octet (7)</th>
<th>Host Three last octets(24)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class B</td>
<td>Decimal range: 128-191</td>
<td>First Octet (binary)</td>
<td>Network Two first octets (14)</td>
<td>Host Two last octets (16)</td>
</tr>
</tbody>
</table>
Class C
Decimal range: 192-223
First Octet (binary) 110
Network
Three first octets (21)
Host
Last octet (8)

Class D
Decimal range: 224-239
First Octet (binary) 1110
Multicasting

Class E
Decimal range: 240-254
First Octet (binary) 11110
Reserved for experiment and a future use

Figure 7: Structure of the different classes of IP address

The IP address is composed as we can see above of a network part and the host part. The table below show the number of usable host and network capacities.

<table>
<thead>
<tr>
<th>Class</th>
<th>Number of usable networks:</th>
<th>Number of usable hosts:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>$126 \times (2^7 - 2)$</td>
<td>$16.777.214 \times (2^{24} - 2)$</td>
</tr>
<tr>
<td>Class B</td>
<td>$16.382 \times (2^{14} - 2)$</td>
<td>$65.534 \times (2^{16} - 2)$</td>
</tr>
<tr>
<td>Class C</td>
<td>$2.097.150 \times (2^{21} - 2)$</td>
<td>$254 \times (2^8 - 2)$</td>
</tr>
</tbody>
</table>

Table 3: Usable host and network capacities

The ‘-2’ is corresponding to the number of actual usable network corresponding to the network address and the broadcast address which are unusable for affecting IP addresses. The class A IP address 127 is reserved for loop back and diagnostic functions.

Some classes are also used for private allocation, for example if an ATM machine is needed to be interconnected but for obvious security reasons, the machine does not need to be connected to the Internet, these addresses can then be used.

- Class A: 10.x.x.x
- Class B: 172.16.x.x \(\rightarrow\) 172.31.x.x
- Class C: 192.168.0.x \(\rightarrow\) 192.168.255.x

Routers does not normally forward traffic coming from these addresses as they are reserved for private usage.

2.7 Subnet mask

As has been demonstrated previously the IP address is composed of two fields the Network address and the Host address. A subnet mask is used to extract network and subnet information from the IP address in order to route information.

Many reasons determine the need of applying a subnet mask:

- Reduce the size of the broadcast
- Split up local area network into different locations
- Security reasons for separating traffic

The default subnet mask is created by borrowing some bits from the host’s portion of the IP address. Router use the ANDing process in order to determine if the destination host is in the
same network. The ANDing process consist of applying an AND logical function of the IP address and the subnet mask, the result giving the network where the IP address is located.

<table>
<thead>
<tr>
<th>Host IP address:</th>
<th>193.1.1.12</th>
<th>11000001.00000001.00000001.00001100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet Mask:</td>
<td>Class C</td>
<td>11111111.11111111.11111111.00000000</td>
</tr>
<tr>
<td></td>
<td>255.255.255.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>193.1.1.0</td>
<td>11000001.00000001.00000000.0000000</td>
</tr>
</tbody>
</table>

Table 4: Example of the ANDing process

The result is then: in decimal 193.1.1.0 the host seen previously is then attached to this network.

Customs subnet mask can be created by modifying the subnet mask and adapting the subnet process to the specifications needed. Unfortunately this method is *lossy*, due to the result of the unusable first and last address for each subnet created using a custom subnet mask.

### 2.7.1 IP version 6 (IPv6) / IPng

IPv6 is also known as the IPng where *ng* stands for next generation. Currently the version 4 (IPv4) is used by layer 3 of the OSI model; some vendors have already released hardware and software compliant with IPv6. IPv6. This is the period of cohabitation of the IPv4 and IPv6 (Figure 8).

The IPv6 is presented here in order to give an overview of tomorrow’s technologies concerning router hardware and to provide guidelines for a possible evolution of the application Router Configuration Emulation realised.

IP version 6 was first recommended by the I.E.T.F. (Internet Engineering Task Force) in 94. The (IPNGW G.) Next Generation Working Group has been contracted by the previous organisation to develop this next version of IP addressing.

The IPv6 is composed of 128 bits length in comparison with IPv4, as it has been shown previously, which contain 32 bits length.

According to a population prediction there will be 9 billion people in 2050. The IPv4 is coping with a maximum of over 4 billion ($2^{32}$) of addresses. With the emerging development of mobile phones, personal assistants and wearable computing, the need of IP addresses will increase within few years. That is one of the reason that a new format of IP addresses is needed, as the IPv6 allow over an impressive number of 340 trillion trillion trillion addresses.

![Figure 8: Cohabitation of IPv4 and IPv6 occurring at layer 3 of the OSI model](image)

Furthermore the specification of IPv6 allow great options available such as the “Anycast” in addition of existing methods of addressing (Multicast(one host to multiple hosts) and Unicast(one host to one other host)) method described by Harnish Patel as the possibility for example, that a router to update its routing tables and the opportunity for a host to forward a message to the nearest hosts with the aim that any hosts can manage the forwarding decision.

IPv6 provides mechanisms for security aspects; the IPv6 specifications include two cryptographic methods. The first method is A.H. (Authentication header) and the E.S.P.
(Encapsulating Security Payload). The first one provides data integrity and authentication method and the second one provides integrity and security of contents.

According to the paper published by Bill Hancock “IPV6 Security Enhancements Still Not Everything You Need” the IPv6 suffers from security weaknesses at the moment. Even if many Operating systems (Linux, Digital Unix, Microsoft IPv6, FreeBSD…) and some hardware are already compliant with IPv6, various aspects must then be solved such as the security problem, technical problems…, which can be ensured by new standards specifications. IPv6 appears to offer great inevitable advantages and it appears to be the next big step in the network area. In the context of the application built along this project the theory based on the IPv6 made along with this report can be used for a possible extension of the software, as we will discuss later, some Cisco routers IOS are already compliant with IPv6. [8] [9] [10] [11] RFC2080

2.8 Routing protocols

A previous part has introduced a routed protocol (IP). Confusion between routing protocols and routed protocols need to be avoided: “Routing protocols determine the path that routed protocols need to follow to their destination. “ Source: Cisco Semester 1. An overview of the actual routing protocols will be studied they are necessary for the implementation of the Router Configuration Emulation software.

Routing protocols enable router to create an internal map, according to mathematical algorithms in order to be able to route information. [7] This map will enable the router to forward information by selecting the best path for the data packets.

The majors routing protocols will be examined, examples of such protocols being: RIP, IGRP EIGRP, and OSPF. [12]

2.8.1 RIP (Routing Information Protocol)

Rip is one of the oldest routing protocols, the research on which the algorithms are based on dating from 1957. An upgrade of the original version of RIP has been made in 1994; this upgrade is known under the name of RIP 2. This last implementation has enabled the implementation of security on the routing tables, the ability to carry more information and finally the support of subnet masks previously shown within the theory part.

RIP is a IGP (Interior Gateway Protocol) it calculates the distances to the receiver in terms of hops. A hop is a metric value, for example between the source and the receiver if only one node separate the two nodes the value for the distance will be 2, see the example bellow:

![Simple example of the Hops count using RIP algorithm and directed graph](image)

Assuming that the node one is the sender and the node three the receiver, and the weight of the arc are equals, the number of hops between one and three will be two. RIP is limited to a maximum count of 15 hops (The value is set to 16 for hops count distance but the maximum that the number of hops that occur is 15), over this value the destination is considered as unreachable. [B3] RFC1083.
RIP protocol, as we seen previously is an IGP protocol, it calculates the distance from a sender to the destination. Information is stored on their routing tables. RIP is known as a distance vector protocol and it sends periodically information to each of the closer router. The type of update used by RIP is known as broadcast, routers transmit their information to other router at regular intervals, according to a timer included on the router, often set to 30s.

Unfortunately this technique presents two major disadvantage such as the bandwidth problems, but it is still widely used due to ease of configuration, stability and the major support by both software and hardware. This protocol is implemented in the Router Configuration Emulation as this is the basic routing protocol and the most used for the reasons explored previously.

As the RIP protocol is the protocol implemented in the Router Configuration Emulation Software, an account of some other protocols will be described for a possible expansion of the project.

![Routing Protocol](Rip)

<table>
<thead>
<tr>
<th>Routing Protocol</th>
<th>Rip</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.5.5.0</td>
<td></td>
</tr>
<tr>
<td>205.7.5.0</td>
<td></td>
</tr>
<tr>
<td>201.100.11.0</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 10:** Implementation of the RIP protocol in the Router Configuration Emulation Software

### 2.8.2 IGRP / EIGRP

IGRP stands for (Interior Gateway Routing Protocol) is as we seen previously like RIP protocol a distance vector protocol, which periodically sends information to each of its neighbours for updating its routing table. Like RIP IGRP presents problem of bandwidth when an upgrade of routing table occurs and the upgrade is made bit by bit, no overall upgrade is made.

Unlike the RIP protocol, IGRP does not limit its decision on the number of Hops counted, which is not necessarily the best method to forward data packets. IGRP takes account of the bandwidth, the load of the network, the reliability of the hosts and the propagation delay. Furthermore the maximum number of Hops is 255.

E.IGRP stands for (Enhanced Interior Gateway Routing Protocol) is a new protocol created to enhance the efficiency of the IGRP by taking advantages of the distance protocols and those of link state protocol.

A link state is a protocol which allows each router to transmit routing information to all other router only when changes occur within the routing tables. This method overcomes the problem of distance vector protocol, by forwarding routing table only when changes occur and a major table update is done at the beginning. This protocol does present disadvantages such as an initial high bandwidth requirement and processing and memory needed is important. RFC2896. RFC 1700.

### 2.8.3 OSPF

OSPF stands for (Open Short Path first) which is a determination of the optimum path. OSPF is a link state protocol which presents the same advantages and disadvantages as the IGRP presented before. In addition, OSPF contains criteria which include the speed of the Route taken to forward packets, the actual traffic. This protocol provides reliability and security, and was mainly designed to address the needs of large network that RIP could not handle. RFC1060.
2.9 Router Fundamentals

This part of the report will present the router fundamental, which are necessary to give an overall presentation of the step used to configure a real router (Called the Boot), which type of memory does the configuration use and an overall presentation of the I.O.S (Internetwork Operating System) which is used to configure the router will be explored as this is one of the most important part of the implementation of the Router Configuration Emulation software.

![Cisco 2500 Router](www.cisco.com)

The software is based on the Cisco 2519 as it provides 2 auxiliary ports and 2 serial ports, the rear of the router will be detailed below with an explanation of the router port following:

![Rear view of the Router 2519](www.cisco.com)

The two AUI (Attachment Unit Interface) placed on the left of the Router are used for an Ethernet connection at 10Mbps. These ports are used to connect the router to a hub or a patch panel (using the appropriate connectors).

Two serial ports are also available on the routers. Theses connections are used for a connection to a W.A.N. (Wide Area Network). Typically other routers are connected to these ports. The cable is a DB-25 connector. [3]

![DB25 cable](www.cisco.com)
Two Connectors Console and Aux appear on the right side of the rear of the router. The console port is an RJ-45 used to connect and configure the router. Typically, a RJ-45 connector is plug into the router and at the other end the signal is converted to a com1 connector in order to be plug to a computer. A program such as hyperterminal can be used to configure the router.

The second connector is also a RJ-45 connector which is used for connecting an auxiliary device such as a modem. The auxiliary connector will be used to configure the router using modem commands. This is possible to configure the router using a dial up device such as a modem to perform remote router configuration.

At the right hand of the router appear one switch to power on/off the router and a power connection to supply power to the router.

2.10 Router memory

Different memories are used into the electronics of the router. These memories will be studied in order to show where and how the configuration of the router is actually stored and accessed.

When the router is powered, this will load the IOS (Internetwork Operating System) from an EEPROM (Electronically Erasable Programmable Read Only Memory). This EEPROM is flash memory which allows erasing the actual configuration and replaces it by a new configuration. If a configuration file is present in the EEPROM the IOS will load and then look for a valid connection (Start up Config) that is stored in a (NVRAM) Non volatile Random Access Memory.

If no configuration file is present on the NVRAM the router will use a setup mode. This setup mode is used to have access to global commands. In the context of this emulator we assume that a file configuration is present on the NVRAM or the prompt to enter in this mode has been negative so the CLI (Command Line Interface). This is the best way to configure a router as it offers a more flexible way to configure a router and it is at this stage that our emulator is starting.

Figure 14 demonstrates the different steps performed by the router once it is powered on. The configuration of the router will be performed at the CLI (Command Line Interface) for this emulator (Such as all the emulators available on the market: Cisco, Sim, and Boson). This mode is the one which allows the most flexibility. The other mode shown in the initial configuration dialog which is useful to rapidly perform a configuration but it is fairly limited. The CLI will be the most powerful mode of internal router configuration environment. Router Configuration Emulation is using this mode as this is the mode which gives the most flexibility in the configuration process.
Honours Project (CO42019) Report – Router Configuration Emulation –

Power on Router

Load the I.O.S. (Internetwork Operating System) from an EEPROM memory.

The I.O.S. will load and look for a valid configuration stored into a NVRAM

If the configuration file is found the router launch the actual I.O.S. configuration from the NVRAM memory

If no configuration file is present on the NVRAM the set up mode is launch.

The user is prompted to enter in the initial configuration. “Would you like to enter into the initial configuration dialog?”

If the answer is positive the user will be asked to perform a Basic management or extended set up.

If the answer is negative the user enters into the C.L.I. (Command Line Interface).

Emulation is starting at this point using the C.L.I. interface.

If at the end of the configuration the user discard the running config configuration and decide to go to the C.L.I. prompt.

If the configuration is saved each time that the router is rebooted the configuration saved will be loaded from the NVRAM

Figure 14: Description of the Booting sequence: memories, configuration interfaces and situation of the emulation by the Router Configuration Emulation software.

2.10.1 CLI

As we have seen previously this mode is accessed when the router is first powered on and the answer to the first question “Would you like to enter the initial configuration?” is NO and the confirmation is entered by the command yes. The other way to enter into the CLI
configuration is when the initial configuration has been performed, more generally if a configuration file is present on the NVRAM.

After pressing the enter key

![Figure 15: By pressing the Enter key, the CLI mode is launched.](image)

The interface status message appears in order to show the state of the different interfaces the CLI prompt appears, the configuration can then been started.

2.11 CLI prompts

The notion of prompts is really important as part of the router programming. According to the prompt displayed, different command are available. The graph below shows the different CLI prompts. Note that the different commands available at each prompts can be listed by typing the “help” command. Within the implementation part of this report the commands associated with the different prompts presented here and their usage will be detailed.

Figure 16 presents the different prompts available in the CLI mode. The First mode is the initial mode shown previously in Figure 15, no commands apart from the enter key are allowed in this mode.

Once the enter key is pressed, if no passwords are activated the prompt “Router>” is displayed, this mode allows very limited commands such as ping, show certain aspect of configuration and trace route to check the configuration. Once the command enable is entered from the user mode, the privileged mode is activated (if no password is set to enter in this mode.). This mode allows further investigation of the configuration made, allow telnet, advanced show configuration etc. Once the command “config t” or “config terminal” is entered, the Router(config)# is displayed. This mode allows the resolution of the hostname, plus advanced feature for router configuration such as the possibility of setting the hostname table.

This mode allows also launching one of the other modes:

- **Router(config-if)#** is mainly used to set the IP addresses on the ports;
- **Router(config-subif)#** is not emulated but its used to configure the sub interfaces and allows creation of virtual interfaces within the router.
- **Router(config-line)#** is used for configuring auxiliary line, console, and passwords.
- **Router(config-router)#** prompt is mainly used to configure the routing protocol

Note that the description above explains the major uses of these CLI prompts. All the commands available for each of the prompts can be used using the help command of the router. Also, the word “Router” is here to refer to the hostname configured for the router, if the hostname is changed all the prompts are affected with the new hostname (e.g.
Honours Project (CO42019) Report – Router Configuration Emulation –

Atlanta(config)# is the global configuration prompt for the hostname configured: “Atlanta”.

Figure 16 shows the different prompts and their links:

![Figure 16: Representation of the different CLI prompts and their connections.](image)

2.12 IOS

The last part of this theory section is dedicated to the IOS; IOS (Internetwork Operating System) is the kernel of the router language. IOS is a set of commands recognisable by router and certain switches.

These commands are entered at the end of each of the prompts shown before in order to configure the intended device, in our case the IOS will be based on router configuration. The set of commands can be obtained by typing “Help” of each of the prompts. The implementation in the area of this project will be shown in chapter four of this report. RFC1010.

2.13 Conclusion

This conclusion based on the theory part will highlight the outcomes of the theory part. After an introduction based on the different aspects that will be researched in this report, TCP has been viewed. Historical factors have highlighted the origins of TCP/IP and given an explanation of its popularity. A general presentation of TCP has been viewed in order to give a good background to TCP in order to introduce IP TCP is also the most widely protocol used mainly due to the impressive growth of Internet. TCP header, ports and the ways of synchronising and sending data with the “window” notion have been viewed. [1]

UDP protocol has been briefly viewed. These parts have been reviewed according to RFCs and academic papers published. A presentation of the OSI model in relation of the TCP model has been examined in order to show and explain at which layers of the OSI model routers are working to forward data packets.

One of the important parts of this theory: IP has been introduced such as subnet masks, in order to divide networks into sub nets according to the requirements made.

A presentation on the different routing protocols has been shown in respect to RFCs and various books and academic papers published. The protocols investigated have been RIP, IGRP / EIGRP and OSPF.
The last part of the theory investigated the router fundamentals such as the router boot, the different memory involved, the router connection, passwords, router prompts and a presentation of IOS.

The next part of this report will be dedicated to the Design / Modelling of the application implemented in the context of this project.
3 Design/Modelling

3.1 Introduction

This part of the report will highlight the modelling of the application and the set of rules applied to develop the Router Configuration Emulation software. An overview of the requirements will be given by a general view of the application. The language chosen for developing this application is Java. This is a high level programming language, and the strength of Java will be reviewed in more detail in the next part of this report, within the implementation part.

Different software engineering methods will be studied, and the selected method of development will be highlighted. After a view of the software engineering method, the main specifications within the high level design part will be examined; this part will highlight the global input/output of the system. Finally, a more specific view such as the relationship within the components will be highlighted. A conclusion will provide an overview of the different subjects developed during the Design / Modelling part.

3.2 Requirements

The final application should provide a fully implemented package programmed in the Java language, used to enhance the learning of router programming.

Most of the IOS commands need to be interpreted and emulated in order to affect the IP addresses and subnet masks and affect the different passwords inherent to router programming.

Hostname resolution and the implementation of the routing protocol needs to be emulated in the context of this application. The application will need to be able to perform the interpretation of most of the IOS commands, to perform router configuration. Extra parts have been added such as the feature to open and to save the file configuration of the actual router programming. A better feedback using a graphical design drawing of what has been configured using the IOS commands in the emulator has been emulated.

3.3 Software engineering methods

Most of the time when software is produced, the progress follows a set of rules used to allow management and control of the software which is being produced.

According to the book *Software engineering* by Ian Sommerville, “Like all engineering, software engineering is not just about producing products but involves producing products in a cost effective way”. In addition the software engineering methods will be followed in order to limit the cost, improve reliability, requirements, improve the quality of the software produced and to improve re-use of the software produced and to allow further development of the application.

In the context of this application different software engineering methods will be described, and the justification of the method selected will be described. Three major software engineering methods will be defined here, the *Waterfall Model*, *Evolutionary development* and *Boehm’s spiral model*. [B7]
3.3.1 Waterfall Model

The waterfall life-cycle model is essentially a set of five steps that the project follows leading to the operation and maintenance step. Simplistically these are: determine user needs, define requirements, design the system, and implement the system, test, fix, and deliver the system.

![Figure 17: Waterfall model, software life cycle.](image)

The advantages of the waterfall method is:

- Well-studied, well-understood, and well-defined.
- Easy to model and understand.
- Easy to plan and monitor.
- Many management tools exist to support this life-cycle model.

but it suffers from several disadvantages, including:

- Most if not all requirements must be known up front.
- Does not readily accommodate requirements changes.
- Product is not available for initial use until the project is nearly done.

This software engineering method does not appear to be valid in the context of the Router Configuration Emulation, mainly due to the fact that all the requirements must be known at the beginning of the project. [B7] [B9]

3.3.2 Evolutionary development

This method of development is based on developing an initial prototype and cycling through the different steps such as the specification, development and validation until the final version of the software is created. Figure 18 shows the evolutionary development method:
The advantages of this method are:

- Development of a small system gives a faster development cycle.
- Suits systems with a short lifetime.
- Development of systems where it is impossible to express detailed specifications in advance.

But the drawbacks include:

- The process is not visible.
- Systems are usually poorly structured.
- Special motivation of the programmers is needed as the development is not visible and no global structure is followed.

Once more the evolutionary development will not be the technique followed for this project due to the non-visible process development. [B7] [B9]

### 3.3.3 Boehm’s spiral model

The main idea behind the development of the spiral is a need for an improved process of the two generic development methods exposed before. Boehm’s model takes the form of a spiral, starting from the centre. Each time a loop is made a phase of the development is achieved. There are no fixed phases in this model which allow adapting the model according to the requirements or needs of the software development. Each phase of the spiral contains risk analysis, planning, requirements analysis, engineering and evaluation.

Its advantages are:

- Good visibility of the process
- Possibility of adapting the phases according to the requirements
- Allows the possibility of creating the prototype and re-cycling in order to redefine the requirements, the development, the test, until the final software is developed.

and its drawbacks include:

- This model provides guidelines
- A set of procedures is not clearly stated
This method will be selected over the different software engineering methods due to the ability of adapting the software development model when it fits with the need of development of this software. The cycle of the development chosen is: review of the project, risk analysis, prototype, requirement analysis and the cycle loop again.

This model has been chosen mainly due to the adaptation of the requirements to the development of the software. Furthermore this model contains the advantages of the two previous methods. It also takes account of risk analysis. After a phase of the requirement and the risk analysis cycle, a prototype has been produced and validated in the Requirement analysis and validation phase. This method was also chosen as the first prototype was not adequate due to technology constraints.

![Boehm's spiral model used for the development of the project](image)

**Figure 19:** Boehm’s spiral model used for the development of the project

### 3.3.4 Risk analysis phase

This part involves an initial risk analysis encountered during the development of the application due to the technology used is displayed in Figure 20. The first part of the implementation has been to develop a Java applet in order to be able to run the emulator via a web site and avoid different problems, such as the installation of the product on each machine, and platform dependence…

Unfortunately, this solution has been abandoned due to the major problem encountered. The first problem is that not all the browsers supported the key listener and key events, which is the first command which is entered when a router need to be programmed. Furthermore, the snapshot above shows the applet running under Windows XP. After the decision of Microsoft, does not include JVM with Windows XP, java was not supported by internet explorer. Even after upgrading the JVM from Microsoft and sun, the browser supported java, but the key event was not working.

After agreement with my supervisor Dr. Bill Buchanan, a Java application has been created instead of an applet, which avoid too many compatibility browser problems and it enables the uses of Java Swing components. This clearly demonstrates the need of using Boehm’s spiral as the software engineering methods due to the problem inherent within the development. After the review with my supervisor the requirements have been adapted and the cycle has looped until the final version of the application. [B7] [B9]
3.4 High-level design

As explained within the requirement part the aim is to create a router emulator which performs the following:

- A package programmed in Java used to enhance the learning of router programming.
- Most of the IOS commands need to be interpreted and emulated, here is a non exhaustive list of the needs required by the application:
  
  - IP addresses
  - Hostname
  - Subnet masks
  - Different passwords
  - Routing protocol
  - Routing table
  - Different CLI(Command Line Interface) prompts

The application will need to be able to perform the interpretation of most of the IOS commands, to perform router configuration. The possibility of opening and saving configuration files has been added to the project and implemented, such as providing a better feedback of the IOS commands typed when configuring the router (Note that this facility is NOT available in a real router and it has been implemented mainly to enhance router programming skills.)

This part of the report will detail at a high level design the different components of the application.

According to Meyer (1987) "The construction of software systems as structured collections of abstract data type implementations". This part will show the methods used in this program and the relationship between these methods. [B7]

In object-oriented programming, the functionality of the program will be described using objects which interacts with each others. The different methods used in this program
demonstrate the model used during the development of the application. The model of an object which is the definition of the entity modelled.

![Diagram of methods interaction](image)

**Figure 21:** Model of an object and the interaction of the methods

[Source modified and adapted from Software Development by Dr. John Owens]

### 3.4.1 Modelling the software

The following part will give a similar figure for the methods used for the Router Configuration Emulation software and highlight the different interactions between the methods created. A deeper description will be given in the next part of the design: Component design.

This part develops the main methods implemented in the context of this software. All the methods will be described in the next part, this part only highlighting the input / output of the program.

An account of the method chosen will be studied in order to explain (Based on in Figure 22) the high-level design of the program. This part will highlight the main input / output of the program.

### 3.4.2 Global Input / Output of the system

This part will highlight the global Input / Output of the software according to the model shown above. The model above is a representation of an object: RouterConfigEmu. The different parts which compose this model will be viewed in order to show, at a high level of the design, the input / output of the program.

1) **Main class**
   This Class is always present in each program developed; this is the starting point of the application. The main class contain global variables which can be accessed by any methods within the same objects.

2) **Exit form**
   This class is very simple and straightforward, it controls the close button based on the form created. If the main form is closed, the program closes.

   If this method is omitted, it results to an error exception when the current form exits.

3) **Initializations (Variables; Graphic environment)**
   This class contains the different variables, mainly for the graphic environment as this provides the exact position on the screen of the different Java Swings components which need to be launched at the beginning such as others global variables.

4) **Java Swing Graphic environment**
   This part contains the different elements integrated for the form and the different elements which are generated by Forte for Java, which is the program used to develop this application.
5) Key Event Listener
This class will detect the ‘Enter’ key. The Enter key is detected each time a user presses it, the program will launch the CLI (Command Line Interface). (Refer to the Theory part on the CLI prompts). This is the way that the program interprets the command typed by detecting at each stage of the router programming which commands the user has typed. Once the user has pressed enter the relation to the interpretation of the command is immediate and follows in the next part (no 6,7,8).

6) CLI Prompts
The program produces the different CLI prompts according to the level of the programming and the command typed. (Remainder from the theory part example of a CLI prompt Router(config-line)#)

7) Interpretation of the command typed
This method provides an interpretation of the command typed. Once the user has pressed the ‘Enter’ key, or if the user has loaded a saved configuration, the method performs strings manipulations in order to extract the command typed by the user. The command is then interpreted and the next part is launched (no 8).

8) Feedback of the Command Typed
Once the command has been interpreted by the program the user receives feedback of the command typed. The feedback is different according to the level of the CLI (Command Line Interface) and the command typed.
9) Variables Affected after the Interpretation of the commands
Once the interpretation of the command typed is made, variables can be from two different natures, global variables or graphics variables. These variables are affected accordingly and the user perceives immediate feedback of the commands typed, either graphics or command line based.

10) Java Swing Menu
The other way to program the router is to load a router file configuration using the Java Swing menu. This implementation was not required but it has been implemented later on in order to enhance router programming skills and to enable to saving and loading of router configurations files. The about menu gives feedback of the software by giving the name of the author, and my supervisor with the contacts associated. This states also that the router has been based on the IOS commands performed by Cisco routers.

11) Action Performed associated with the menu
This step jumps the detection of the ‘Enter’ key as this clearly appears in the model of the object. The stage 5 is not performed if the user decides to launch the router configuration using the launch menu. This is the other way to perform the configuration of the router if a previous file has been saved. Note that in the context of this application some files that will be detailed within the Test part of this report, have been saved by default. The screen shot of the application developed is used to associate the methods explained before. The different methods can then been associated with the application developed.

After giving an account of the global input / output of the different methods that compose the program, an analysis on the different components which compose this model will be highlighted.

Figure 23: Representation of the different methods which compose the application using a screen shot of the software developed.
3.5 Components design analysis

This part will demonstrate the different methods associated with the object (RouterConfigEmu). This analysis represents the diverse parts associated with the methods which have been highlighted before. This is more specific than the previous part; the different methods associated will be explained in more details:

3.5.1 Main class

This part of the object contains only one single class which are mainly global variables which can be accessed by the different methods.

- Public static void main(java.lang.String[] args){}

This class contain one argument which is not used in the context of our program as the application needs to be straightforward to use for the intended users.

3.5.2 Exit form

The Exit form once more contains a single method which is launched when the program detects that the user has clicked on the top right hand corner of the form in order to exit it:

- private void exitForm(java.awt.event.WindowEvent evt) {}

This method uses the AWT, and provides some components which fire events when a user interacts with the components. In this case the AWTEvent is used when users close the form and exit correctly the program.

3.5.3 Initialisations (Variables; Graphic environment)

This method provides different functions associated with it, this is mainly a declaration of global variables either for the graphics part and the global variables. The definition of these methods are not generated by Forte for Java.

- public class RouterConfigEmu extends javax.swing.JFrame{}

This method is launched by the main method viewed previously. It is extended from the package javax.swing which contains the different Java swing components used in the context of this application (Frame, menu…) gives a much better look and feel of the application. This class provides the main global variables used by the other methods described below.

- public RouterConfigEmu() {

This class provides another set of more specific variable declarations which are not needed by all the classes. This class launches the init() and initComponents() methods in order to start the application.

The init(){} is inherited from the method RouterConfigEmu and contains the global variables which are not generated using Forte. This set of definitions has been included at the end of the development of the application in order to provide the graphics feedback to the user when IOS commands are entered. This enhances the router programming skills by providing better feedback. This will be demonstrated within the Test part of this application.
3.5.4 Java Swing Graphic environment

After drawing the different areas of the application using forge, the program creates a range of component declarations according to the graphics made (Position of the frame on the screen, position of the menu, the different elements…)

This is composed by one method:

- initcomponents()

3.5.5 Key Event Listener

This method is accessed by the program if and only if the user presses the enter key.

- private void DisplayRouterKeyPressed(java.awt.event.KeyEvent evt) {}

The method uses (like the exit method viewed previously) the AWT event, there is a key listener where the commands are typed which is checking whether the enter key is pressed or not. If the enter key is pressed the method is launched and the program goes to the next step.

3.5.6 CLI Prompts

The CLI prompts are displayed within the main Panel where the commands are typed. Once the enter key is pressed the prompt is displayed. The prompts are organised in a way that provides different levels of the router configuration according where the router is based. (See the theory part for more details). This part is composed of different methods which display a different prompt. These methods are:

The command typed is first interpreted and the prompt is displayed accordingly:

- PromptModeUser public void PromptModeUser()

  Used to display the normal prompt user: Router>

  This mode is the basic mode, this allows limited configuration, and is used more to check the current configuration of the router and perform commands such as ping…

- ExecutivePassPrompt public void ExecutivePassPrompt()

  Used to display the executive prompt: Router#

  This mode allows further investigations of the configuration made, allows telnet plus advanced show configuration IOS commands

- public void GlobalConfig()

  Used to display the global configuration prompt: Router(config)#

  This mode allows the resolution of the hostname, plus advanced feature for router configuration such as the possibility of setting the hostname table. This prompt is also used to launch the other configuration prompts.

- InterfaceConfigPrompt public void InterfaceConfigPrompt()
Used to display the normal prompt user: Router(config-if)#

This CLI mode is mainly used to set up IP addresses on the ports of the router; it is the short term for interface (In the context of this application 2 Ethernet interfaces and 2 serials interfaces)

- **RouterConfigPrompt**
  - `public void RouterConfigPrompt()`

  Used to display the normal prompt user: Router (config-router)#

  This prompt is mainly used to configure the routing protocol and the routing table.

- **LineConfigPrompt**
  - `public void LineConfigPrompt()`

  Used to display the normal prompt user: Router (config-line)#

  This prompt is used for configuring auxiliary line, console, and passwords.

- **ExecutivePassPrompt**
  - `public void ExecutivePassPrompt()`

  Used to display the normal prompt user: Password:

  When the Executive password is activated, when the user tries to enable this mode the program prompts the user in order to have access to the Executive mode. If the password is not set, the executive password is accessed.

- **ConsolePassPrompt**
  - `public void ConsolePassPrompt()`

  Used to display the console password prompt "User Access Verification" "Password:"

  When the password has been activated, the user is asked about the console password (In a regular basis this event occurs when the user decides to type the command logout or exit.

- **ConsolePassPrompt2**
  - `public void ConsolePassPrompt2()`

  Used to display the console password prompt: “Password:"

  This method is similar than the previous one, this mode is different as the user has three tries to type the correct password, the message “User access Verification does not need to be displayed once more when it is time for the second and third try.

### 3.5.7 Interpretation of Commands typed

Once the prompt is displayed, different functions are used in order to extract from the Panel used to perform the configuration, the command typed by the user. This part of the report does not explain how the extraction of the string is implemented but presents the different methods which extract the command typed. The next part of this report will highlight the way that the program extracts the data.

These commands provide the full extraction of the command typed. The command knows at which stage of the CLI prompts the user is, extracts the command by removing the spaces
and uses the tool string tokenizer. The usage of the tool string tokenizer will be explained in
the next part of this report. Once the command is extracted, the method checks using a Map to
store the different recognised commands, and affects the different variables and provides
feedback to the user.

**InterpretCommand public void InterpretCommand()**

The interpret command will extract a command typed and interpret it according to the CLI:
Router>

As all the different commands in this part behave in the same way a principal explanation is
provided above. Note that the different commands included inside the Map are different
according to the CLI prompt. This allows limited configuration, this mode is used more to
check the actual configuration of the router.

**InterpretCommandPrivileged public void InterpretCommandPrivileged()**

The interpret command will extract the command typed and interpret it according to the CLI:
Router#

As with all the methods implemented in this part this will store data according to its own map
and interpret the command typed accordingly.

**public void GlobalConfigCommand()**

The global command will extract command typed and interpret it according to the CLI:
Router(config)#. This mode allows further investigations of the configuration made, allowing
telnet plus advanced show configuration IOS commands that this command can recognise and
interpret.

**public void ExecutivePassPromptCommand()**

The executive pass prompt command will extract the command typed and check if the password is correct
according to a global variable declared at the beginning of the Object RouterConfigEmu, if the variable
is null, as the user did not configure this variable, this command is not executed.
After three try, such as in real router behaviour, the program exits the password prompt and
returns to the previous CLI prompt (in this case the user prompt). This has been implemented
mainly due to security reasons in real hardware router. No real security reasons are affected
using this emulator but the aim is to provide an emulator as close as possible to the reality.

**public void InterfaceConfigCommand()**

The interface config command will extract the command typed and interpret it according to
the CLI: Router(config-if)#

This method is mainly used to set up IP addresses on the ports and the clock rate for the two
serial interfaces.
public void RouterConfigCommand()

The router config command will extract the command typed and interpret it according to the CLI: Router(config-router)#

This method is used to configure the routing protocol and affect the routing table values.

public void LineConfigCommand()

The line config command will extract the command typed and interpret it according to the CLI: Router(config-line)#

This method is used to extract the commands which are used to configure auxiliary line, console, and associated passwords.

public void ConsolePassPromptCommand()

The console pass prompt command will extract the command typed and interpret it according to the CLI: Password:

This method is very similar to the Executive pass prompt command, the user only having three attempts to access to the user mode. This password is mainly set in order to restrict access to any user event the user mode does not enable which disable access to unwanted users. The only difference is that after three tries the user will see the main screen has if the user had typed logout functions. The number of attempts will then been reset when the user presses the enter key again.

3.5.8 Feedback of the Command Typed and Variables Affected after the Interpretation of the commands

As explained before, when one of the commands is typed and interpreted by the different methods explained above, the global variables are affected accordingly. The user receives a feedback of the command typed. This feedback can be differentiated by two different approaches, the graphical view and the normal feedback given by the command line. This graphical view as explained before is not a component of a real router but this is mainly used to enhance the router programming skills.

3.5.9 Java Swing Menu and ActionPerformed associated with the menu

This is once more implemented but it was not required in the requirement specifications, and is mainly to provide a facility used to create a more powerful environment and to allow the loading and saving the configuration files.

These methods are used to display a Java Swing menu at the top left hand corner of the application. The Java Swing menu gives a much better look and feel of the application. This is composed of two different submenus: “File” with open file configuration, save file configuration and exit; the second submenu is composed of “?” with about.

The declarations of the environment (Variables) are handled at the beginning of the program as explained before. Different functions are associated with these menus.

private void jMenuItemOpenActionPerformed(java.awt.event.ActionEvent evt) {
This command is used to open a configuration file saved previously. When the user loads a configuration file, a new window pops up on the screen asking the user to define the location and the name of the file that the user wants to load.

If the file and the directory are valid (An account of the exception has been implemented, such as if the user does not have the right access to a specific file or is the file is corrupted, or the path is invalid.)

![Image](image_url)

**Figure 24:** The open action associated with the Java Swing menu JOpen.

This method bypasses the “enter” method (Step 5) Key Event Listener) as the user does not need to press the enter key, each command is loaded automatically according to the configuration previously performed. The way that the commands are loaded will be explained in more detail in the next part of this report (Implementation).

```java
private void jMenuItemSaveActionPerformed(java.awt.event.ActionEvent evt) {
    This method is associated with the command save file configuration inside the Java Swing menu. Each time that the user is typing a valid command, these are copied into the preview save area, as shown below.

![Image](image_url)

**Figure 25:** The method used to save the configuration file

Once the user decides to save this file configuration, after clicking on the save button included inside the Java swing menu, this method will select the entire text and prompt the user what name and path the file will be named and take place. Once the user indicated the filename and the path the program will parse all the commands included into the “Save Router Configuration Preview” (See figure above) and create the file. The file created is viewable using notepad to edit it or is perfectly possible to load it using the command load in order to perform further configuration.

```java
private void jMenuItemExitActionPerformed(java.awt.event.ActionEvent evt) {
    This method is very similar than the main exit method, but the difference is that this is accessible using the Java Swing menu.
private void jMenuItemAboutActionPerformed(java.awt.event.ActionEvent evt) {

This method is forming part of the other submenu:”?” which provides information about the author, my supervisor, e-mail addresses contact and the copyright issues mainly due to the fact that the software has been developed according to the IOS developed by Cisco.

All the methods have been described in this section giving a more specific view of the different methods used, their relationship and their usages. The following will give a partial conclusion that will highlight the learning outcomes of this Design / Modelling part of this report.

### 3.6 Conclusion

This part of the report will discuss the learning outcomes of the Design / Modelling part. The general modelling of the application has been investigated by describing the set of rules applied in the development of this application.

In order to demonstrate and to justify the modelling and design of the application a definition of the requirement has been given by highlighting the important points of what the final application needs to provide.

The different engineering methods used within software programming development environment have been reviewed and the justification for the software engineering method used: Boehm’s spiral model has been given.

An overview of the high level design of the programming has been given in order to provide a justification of the model used. This part has highlighted the different Input / Output of the software.

After giving an overview of the high level design the different methods which compose the application have been given and a relationship within the components used have been shown.

The language chosen for developing this application is Java. This is a high level programming language, and the strength of Java will viewed in the next part of this report within the implementation part. The implementation part is based on the model presented in the context of the Modelling / Design part. The Global input /output and the component design background developed during this part will be essential in order to explain the implementation part.
4 Implementation

4.1 Introduction

This part of the report is intended to study the implementation of the Router Configuration Software. The presentation of the implementation is related to the theory and design / modelling parts of this report as well as the theory part.

As Java is the language used to program the Router Configuration Emulation software an account of the strength of this programming language will be given in order to justify this choice. The implementation of the different functions is related to the modelling / design part. This implementation will make reference to the main model given in the previous part.

The different methods which are used to implement this software will be detailed, such as the way of implementing the CLI prompts and extracting the command typed. Not all the commands emulated will be described; few commands will be detailed in order to demonstrate the implementation of these. This is mainly due to the length of the program which is around 3500 lines of code.

The next part of the implementation is to show how the program handles Open and Save for the configuration files. A partial conclusion will be given before the evaluation / testing of the application, in the next part of the report.

4.2 Programming language chosen: Java

Historically Java has been developed to construct software for electronic chips. The programming languages available at this time were C and C++; Java then responding to the need to the programmers mainly due to its platform independence, as this was easy to convey a code developed for a specific electronic chip architecture to another one without the need of recreate the program for this other chip.

The justification of the programming language chosen, Java, will be made by highlighting the strength of this language. Java is a powerful programming language which provides many libraries. Theses libraries are imported at the beginning of the program. They give a lot of functionality such as the graphics interfaces, mainly used in this application, and also the ordinary programming language functions. Different libraries have been used to develop this project such as the Java Swing components which are now well supported by the Java Virtual Machine under which the java code is running. The input / output library has already been used and the different tools to manipulate string and store information have been widely used (Hash map and String Tokenizer).

Java is a robust programming language which enables the programmer to create a powerful and professional application using a wide range of features provided. It is an object-oriented language in which the implementer designs objects and then the programmer calls a set of methods of theses objects to program the application. A model of the main object is shown in the previous part with the different methods associated.

Java is platform independent; this is one of the reasons why this programming language has been adopted. In the context of this application, the tests that will show in the next part that the application can run on another platform (Linux Mandrake 8.1 is the platform tested). This is really interesting, to have the possibility of using the same application using different platforms simply by installing a Java Virtual Machine. The application can be launched and can enhance the router programming of a wider range of users it is not only reserved for
windows based PC. Users with Macintosh, Linux, FreeBSD etc can use the application without the need of programming a new application based on the different architecture.

The following will summarise the main reasons why Java has been used in order to develop this application:

- Powerful
- Impressive range of libraries available
- Robust
- Object-oriented
- Very good support using Online Java JDK help features
- Platform independent

[B1] [B4] [B5] [B6]

4.3 Implementation of the router commands

This part of the report will detail the different methods which compose the programmed application. The four first methods, which refers to the modelling of the application studied in the previous part of the report, will not be completely detailed. They mainly contain variable declarations which are essential for the other methods of the application, used as global variables and also used for graphical generation.

4.3.1 Libraries declaration

```java
import java.util;
import java.io.*;
import java.awt.*;
import javax.swing.*;
```

The description above shows the four principal libraries used for implementing the application. The Java.util is used to store the different recognised command into a hash map. This is a really powerful tool in java in order to store and retrieve information. The description of the hash map will be detailed later on.

The Java util library is also used for the string tokenizer which performs advanced string manipulation used to extract the commands entered by the user and compare them in order to see if it does match with the hash map. The java io stands for Input Output this is mainly used for the open and saved file configuration as they use streams of data to open from a file and store into a file.

The AWT is the Abstract Window toolkit which is used to carry out output to the screen. The last library imported is the set of the Java Swing components which is a set of attractive components (In comparison with the normal components) which give a definitive professional look and feel to the application.
4.3.2 Main types of declaration

The following describes one of the most important components of this application. This is where the user will type the IOS commands in order to configure the router. This declaration is very similar to the other components of this application but will not be detailed here as they are similar to the following presented.

```java
private javax.swing.JTextArea DisplayRouter;
DisplayRouter = new javax.swing.JTextArea();
jScrollPane1.setViewportView(DisplayRouter);
```

A private method is giving a definition of the Component used and the second line asks the program to create a new text area. Finally the JTextArea is added using the third line of code.

Another type of declaration is concerned with the graphic such as:

```java
logoNapier.setIcon(new javax.swing.ImageIcon(getClass().getResource("/NapierLogo.JPG");
logoNapier.setText("jLabel5");
getContentPane().add(logoNapier);
logoNapier.setBounds(290, 480, 140, 30);
```

This type of declaration is similar that the previous one, the general concept is still the same, create a component, define its property and display it on the screen. This declaration will get the icon, in this case the directory where the picture reside, include it on the main panel and finally declare its properties such as the size wanted.

The last type of declaration is concerned with global declarations. This is very straightforward one example of this type of declaration:

```java
String InterfaceE0State = "Down";
```

This case is a global declaration of a string which is initially set to down, as long as the interface Ethernet 0 is not activated the value will remain the same.

All the declarations will not be examined due to their impressive number. The three examples above describe the different kinds of declaration implemented in this software.
4.4 Exit method

The second method (exit) only contains one line of code which is used to properly exit the program when the user clicks on the top right hand corner of the application. This method is used in order to avoid the program and/or computer crashing when the user closes the application. This method is straightforward as this contains a single line of code.

```java
private void exitForm(java.awt.event.WindowEvent evt) { System.exit(0); }
```

In the case of the event monitored on the exit button the system will exit the program.

4.5 Key event listener

This is a very important method implemented in this program as this is used to detect if the user presses the enter key. That would mean that a command has been typed and the program needs to extract it according to its command table and the CLI prompt where the user is.

```java
private void DisplayRouterKeyPressed(java.awt.event.KeyEvent evt)
{FIRST:event_DisplayRouterKeyPressed    {
int code = evt.getKeyCode();
if (code == evt.VK_ENTER) {
    switch(turn){
        case 0 : /* initialization */
            turn = 1; /* enable the user mode */
            String output = "" + '
';
            DisplayRouter.append(output);
            PromptModeUser();
            DisplayRouter.setCaretPosition(-'\n');
            break;
        case 1 : /* user mode */
            InterpretCommand();
            DisplayRouter.setCaretPosition('-'n');
            break;
        case 2 : /* privileged mode */
            InterpretCommandPrivilieged();
            DisplayRouter.setCaretPosition('-'n');
            break;
        case 3 : /* privileged mode */
            GlobalConfigCommand();
            DisplayRouter.setCaretPosition('-'n');
            break;
        case 4 : /* Executive Password Prompt */
            ExecutivePassPromptCommand();
            DisplayRouter.setCaretPosition('-'n');
            break;
        case 5 : /* Interface Config Prompt */
            InterfaceConfigCommand();
            DisplayRouter.setCaretPosition('-'n');
            break;
        case 6 : /* Interface Config Prompt */
            RouterConfigCommand();
            DisplayRouter.setCaretPosition('-'n');
            break;
        case 7: /* Line Config Prompt */
            LineConfigCommand();
            DisplayRouter.setCaretPosition('-'n');
            break;
        Case 8: /* Console Password Check */
            ConsolePassPromptCommand();
            DisplayRouter.setCaretPosition('-'n');
            break;  }  }  }   }
```
An account of the internationalisation of the program has been made due to the difference of calling the ‘Enter key’ according to the keyboard used. The method is checking each key typed in the keyboard while the application is on use. The value code will detect the entire key typed, if the code returned by the VK (Virtual Keyboard). The VK is used in order to follow platform independence as the ‘enter’ key might be different from one language to another (eg in French enter key ‘Entrée’ in English ‘Enter’). Rather than implementing a method for each type of Enter key, Java provides the Virtual Keyboard tool which ensures keyboard compatibility.

The value of the switch (turn) is used to launch the interpretation of the command according to the CLI prompt where the user is performing the configuration/. An account of the methods used will be described later. The different turns corresponds to the CLI prompts:

- **Case 0**: This is the first mode launched this is used to enter in the router screen which is prompting the user to “Press return to get started”; at this point no commands need to be interpreted. The case 1 is then activated (If the console password is not activated)

Below the different functions will be launched according to which CLI prompt does the user is performing the configuration.

- **Case 1**: Prompt mode user: Router>; launch the interpretation of the command. (User mode)
- **Case 2**: Privileged mode prompt Router#; launch the interpretation of the command. (Privileged mode).
- **Case 3**: Global Config prompt Router(Config)# launch the interpretation of the command. (Global config mode).
- **Case 4**: Executive Password prompts Password: launch the interpretation of the password typed.
- **Case 5**: Interface Config mode: Router(config-if)# launch the interpretation of the interface command.
- **Case 6**: Router Config mode: Router(config-router)# launch the interpretation of the router command in order to implement the routing protocol and the hostname table.
- **Case 7**: Line Config mode :Router(config-line)# launch the interpretation of the line command (Telnet, auxiliary and console mainly used to set up passwords)
- **Case 8**: Console Password Prompt: Password: , launch the interpretation of the console password.

Note that the hostname can vary according to the name implemented within the application. The “DisplayRouter.setCaretPosition('-\n');” is used to remove a carriage return which is the result of the enter key within a text area. If this is omitted, the command selected will be added to a carriage return which would lead to an error.

### 4.5.1 Displaying CLI prompts

This part of the implementation demonstrates the way that the application is handling the different CLI prompts. Note that not all the prompts will be described here as they are constructed in the same model. An example of the CLI prompts will be described; the interface CLI prompt which is mainly used to configure the IP addresses on the ports of the router, and the console password prompt will be also viewed below.
public void InterfaceConfigPrompt()
{
    String PromptAdd = Hostname + "(config-if)#";
    DisplayRouter.append(PromptAdd);
    DisplayRouter.setCaretPosition(DisplayRouter.getText().length());
}

The code above is called by the method Interface command mode; an account of this method will be given later. A new string is created using the global variable hostname, which is initially Router but the hostname can be modified. This command then takes account of the change of the hostname.

The string is then added to the principal JText Area highlighte before; this is the main text area where the commands are entered in order to configure the router. The last line is used to put the cursor at the proper place on the screen, just after the string. If this line is omitted the string is displayed and the cursor is placed before the string parsed.

The next bit of the program will demonstrate the console password. This prompt is only displayed when the user has activated it and is trying to have access to the user mode prompt, in other words if the console password is activated and the user types the command ‘exit’, ‘logout’ or ‘disable’ the router asks for the console password if the user want to access to the router console.

public void ConsolePassPrompt()
{
    String PromptAddConsolePass = ":" + '
' + "User Access Verification" + '
' + "Password:";
    DisplayRouter.append(PromptAddConsolePass);
    DisplayRouter.setCaretPosition(DisplayRouter.getText().length());
}

public void ConsolePassPrompt2()
{
    String PromptAddConsolePass2 = "Password:";
    DisplayRouter.append(PromptAddConsolePass2);
    DisplayRouter.setCaretPosition(DisplayRouter.getText().length());
}

Two prompts are presented here; the first prompt password informs the user that a password needs to be entered. If the user enters the proper password at the first attempt the router will display the user console prompt (Router>). The user has three attempts in order to type the correct password. For the second and third try, the user console password prompt 2 is used as there is no need to repeat the sentence “User access Verification”. As explained before the prompts are added to the main router configuration screen in order for the user to type the password. The last line is also used in order to put the cursor after the prompt of the string “Password”.

4.5.2 Interpretation of the command typed

This part will demonstrate the way that one of the methods performs the string extraction typed by the user and interpret it in order to provide feedback. Once again not all of the interpretations made by the different functions of this program will be detailed. The extraction and interpretation made by the function InterpretCommand() will be detailed such as the extraction of the password which is slightly different.

The extraction of the command typed is made once the CLI prompt has been included and when the user has pressed the enter key. The other way implemented in this program is when a router configuration file is opened, the interpretation of the command typed is launched when a carriage return is found inside this file, and therefore this interpretation will be detailed later on in this report.
public void InterfaceConfigCommand()
{
    int RelationGlobalConfigInterfaceHashMapInt = 0;
    String AllTextInterfaceConfig = DisplayRouter.getText();
    /* Calculate the Length of the string */
    int StringLengthInterfaceConfig = AllTextInterfaceConfig.length();
    /* Calculate the beginning of the command typed */
    int BeginOfCommandInterfaceConfig = AllTextInterfaceConfig.lastIndexOf("#");
    int BeginOfCommand2InterfaceConfig = BeginOfCommandInterfaceConfig + 1;
    DisplayRouter.select(BeginOfCommand2InterfaceConfig,StringLengthInterfaceConfig);
    /* Take the selection of the text and save it as a string */
    String CommandConfigInterface = DisplayRouter.getSelectedText();
    HashMap GlobalConfigInterfaceCommand = new HashMap();
    String RelationGlobalConfigInterfaceHashMap = "";
    GlobalConfigInterfaceCommand.put(null,"0"); /* if the user has only pressed the 
        enter key without including any commands */
    GlobalConfigInterfaceCommand.put("help", "33");
    GlobalConfigInterfaceCommand.put("exit", "4"); //exit the privileged mode
    GlobalConfigInterfaceCommand.put("ip", "5"); // Program ports with the Ip @ddress
    GlobalConfigInterfaceCommand.put("no", "6"); // Starts up ports
    GlobalConfigInterfaceCommand.put("shutdown", "7"); // Shut down the ports
    GlobalConfigInterfaceCommand.put("clock", "8"); //Configure the DCE Interface clock rate
    The bit of code shown above forms part of the method interpret command; this is the 
    beginning of the method implemented. This representation has been split up in more than one 
    part in order to help understanding, the code shown above is forming the first part of the 
    method implemented and is explained below. The variable RelationGlobalConfigInterfaceHashMapInt is initialised at the beginning of the method, this 
    variable is very important as this will tell which part of the program to launch according to the 
    command typed using this variable associated with a switch.

    The method is then implementing the string extraction, the entire text contained into the 
    main J Text Area, is selected. The length of this text is then calculated by returning the value 
    of the selected text. An integer will then been used in order to detect the last ‘#’ which is 
    forming part of the CLI prompt.

    (Router(config-if)# is the prompt shown for this command extraction).

    The integer value will be added to 1 as we do not want that the command extracted start with 
    the sign #. The selection of the command typed by the user is then made using the values of 
    the entire selected text less the value of the length of the command.

    • DisplayRouter.select(BeginOfCommand2InterfaceConfig,StringLengthInterfaceConfig).

    The selected text is then stored into a string. This string represents the value typed by the user, 
    at this point the extraction of the value has been performed by the program.

    Once the extraction of the command typed is made, the command needs to be compared to 
    the different commands that the program can interpret. The component used here in order to 
    store a data collection of the different commands recognisable by the program is a hash map.
In this case the hash map can store a collection of objects associated with a key. The key is a simple integer which is associated with a string, this string represents the first part of the command that the router can interpret.

```java
GlobalConfigInterfaceCommand.put("help", "33");
```

In this example the command help is associated with the key 33, in order to interpret this command a switch statement will then be employed in order to execute the corresponding code used to interpret this command.

Note that the command implemented is not necessarily the complete command. This is checking the first token of the command, in other words the first part of the command is checked and the corresponding program is launched.

```java
if(CommandConfigInterface == null){
   //   RelationGlobalConfigHashMap = (String)GlobalConfigCommand.get(Command);
   RelationGlobalConfigInterfaceHashMap = "0";
} else {
   // The string tokenizer component is used in order
   // to separate the elements
   StringTokenizer st= new StringTokenizer(CommandConfigInterface);
   if(st.hasMoreTokens()) {
      token11 = st.nextToken();
   }
   if(st.hasMoreTokens()) {
      token22 = st.nextToken();
   }
   if(st.hasMoreTokens()) {
      token33 = st.nextToken();
   }
   if(st.hasMoreTokens()) {
      token44 = st.nextToken();
   }
   if(st.hasMoreTokens()) {
      token55 = st.nextToken();
   }
   if(st.hasMoreTokens()) {
      token66 = st.nextToken();
   }
   RelationGlobalConfigInterfaceHashMap =
   (String)GlobalConfigInterfaceCommand.get(token11);
   RelationGlobalConfigInterfaceHashMapInt =
   Integer.parseInt(RelationGlobalConfigInterfaceHashMap);
}
switch(RelationGlobalConfigInterfaceHashMapInt){
   case 0 :
```

The part of the program shown above describes the second part of the Interface Config Command. If the command typed by the user is void the user has simply pressed the enter key without typing any commands the integer affected is then equal to 0, the switch statement will launch the part of the program corresponding to this integer.

If the command is not void the program uses a string tokenizer. This is a powerful java tool which is used to extract the main string into sub-strings by removing the spaces inside the main string and parsing the elements into new strings called tokens.

E.g. the user has typed: ip address 192.5.5.1 255.255.255.0

```
Token 1    Token 2    Token 3    Token 4
```
The command typed is used in order to affect an IP address and a subnet mask on the router’s port. The string tokenizer will detect all the spaces in-between and save each token as a new string. This set of strings will enhance the way of processing the command typed by the user as each of the sub-string can be checked individually and the corresponding code can then be launched. If the value of the hash map is null the corresponding key number is affected. That means that the user has typed a command which is not recognised by the Router Configuration Emulation software and the exception needs to be handled by providing the appropriate feedback to the user. The key of the hash map is then cast as an integer in order to be used inside the switch statement; the corresponding code will then be launched according to this key.

4.5.3 Example of an emulated IOS command

The explanation of the extraction of the command has been shown above. Once the command is matched with the hash map, the corresponding code is then used to interpret the command typed. The program below shows the interpretation of the command if the user decides to configure one of the interfaces of the router. In this case the interface configured is Ethernet 0. Not all the commands will be shown here due to the length of the program developed. This part of the program does not detail the results if the command typed by the user is not complete or if some of the elements are wrong (The program implements these features), in this case an exception informs the user that the command is incomplete ("% Incomplete Command") or not recognised by the program. ("% Unknown command or computer name, or unable to find computer address").

```java
case 5:
    Test.append(CommandConfigInterface + '\n');
    if(token22.equals("address")){
        if(token33 != ""){
            if(token44 != ""){
                if(PortToAdministerConfigInterface.equals("e0")){
                    InternetAddressE0 = token33;
                    SubnetMaskE0 = token44;
                   IpAddressE0JLabel.setText(InternetAddressE0);
                    SubNetE0JLabel.setText(SubnetMaskE0);
                    token22 = ""; token33 = ""; token44="";
                // Initialization of the variables
                String JumpLineConfigInt = "" + InternetAddressE0 + "T" + "Interface e0 configured: " + "Internet address is:" + InternetAddressE0 + " Subnetmask is:" + SubnetMaskE0 + "\n";
                DisplayRouter.append(JumpLineConfigInt);
                turn = 5;
                InterfaceConfigPrompt();
            }else{}
        }
    }
    break;
```

We assume that the user has reached the CLI prompt Router(config-if)# and has decided to configure the IP address and subnet mask on the port Ethernet 0. The command typed is:

```
ip address 192.5.5.1 255.255.255.0
```

The program then extracts the command, parsed the different elements into sub-strings and the hash map key is 5 according to the first substring ip which is related to the integer 5 in the hash map. The command typed is then copied into the Test Area in order to be saved into a router configuration file if the user decides to do so.

If the second token is equal to address and the two other tokens are not void the command typed is assumed valid. The program checks if the user entered in this prompt with the intention of configuring the port E0 (Ethernet 0). The value of the IP address and subnet mask associated with the token 33 and 44 are assigned to these global values. Also, these values are
assigned to the two labels in order to create a graphical representation of the router configuration performed by the user. The user also receives a feedback inside the configuration panel (Interface e0 configured: Internet address is: 192.5.5.1 Subnet mask is: 255.255.255.0) in addition to the graphical representation. The program then displays a new prompt and is ready to accept a new command.

<table>
<thead>
<tr>
<th>IP Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>e0</td>
</tr>
</tbody>
</table>

**Figure 27:** Graphical representation of the Ethernet 0 router port configuration

### 4.5.4 Executive password verification

The part of the program below represents the way that the application handles the password verification. This example only shows the executive password. The console password verification is also emulated but will not be detailed here as the console password verification follows a similar procedure. The executive password prompt is displayed if the user has set it up and activated it.

```java
if(CommandExecPass.equals(ExecutivePassword)){
    Test.append(CommandExecPass + '\n');
    turn = 2;
    PasswordExecutiveCount = 0; // reset the number of tries
    String output = '\n' + "The password is correct" + '\n';
    DisplayRouter.append(output);
    PrivilegedMode();
} else{
    turn = 4;
    PasswordExecutiveCount = PasswordExecutiveCount + 1;
    if(PasswordExecutiveCount != 3){
        String output = '' + '\n';
        DisplayRouter.append(output);
        ExecutivePassPrompt();
        ExecutivePassPromptCommand();
    } else{
        turn = 4;
        PasswordExecutiveCount = PasswordExecutiveCount + 1;
        if(PasswordExecutiveCount != 3){
            String output = '' + '\n';
            DisplayRouter.append(output);
            ExecutivePassPrompt();
            ExecutivePassPromptCommand();
        } else{
            PasswordExecutiveCount = 0; // reset the number of tries
            String output = '\n' + "%Bad Password" + '\n';
            DisplayRouter.append(output);
            InterpretCommand();
        }
    }
}
```

Once the executive password prompt is displayed the user gets three attempts to enter the correct password. The extraction of the password typed by the user has been made before and is not shown here as the processing is similar than extracting an IOS command which has been explained before. If the password is correct the user has access to the executive prompt (Router#), the password is saved and the number of tries reset to zero. If the password is wrong and the number of tries is less than three, the program displays the password prompt and the user can try again to enter the password. If the user has made three attempts to enter into the executive prompt and the password is incorrect, the program then displays ("%Bad password") and the normal user prompt is once again displayed: (Router>). The number of tries is then reset.

Note that the program is not trying to extract the password if the user only presses the enter key without typing any command, as the string selected is equal to 0.
4.5.5 Java Swing menu: (Open & save files configuration)

The program has been extended to be able to save, load configuration files of the Router Configuration Emulation software. The different methods implemented to implement these functions are explained in the following part. The declaration of the Java swing menu variables will not be defined as the declarations of variables have been seen at the beginning of the implementation part.

**Save file Configuration**

Each time a valid command is entered either by the load function or by the user, the command is copied into a JText area named test. A preview is available on the application in order to be able to see what is actually saved. The code below describes the way that the program is implemented in order to save a configuration file from the application.

```java
private void jMenuItemSaveActionPerformed(java.awt.event.ActionEvent evt) {
    String FileNameSave;
    FileDialog fd = new FileDialog(this, "Save Router Configuration File", 
    FileDialog.SAVE);
    fd.show();
    FileNameSave = fd.getFile();
    String dir = fd.getDirectory();
    try{
        File f = new File(dir, FileNameSave);
        FileWriter fw = new FileWriter(f);
        fw.write(Test.getText());
        fw.close();
    }
    catch (IOException e) {
        System.err.println("File Error:" + e.toString());
        System.exit(1);
    }
}
```

The save file is associated with the Java swing menu item; an action event is associated with the button “Save”. If the user click on the save configuration file, the program is then launched. A string is declared in order to be able to store the file name that the user has entered in order to store the file. A dialogue asks the user for a filename and a location where the user wants to store the file. The program uses a file writer and selects the entire text which is contained in the preview area and saves it according to the name and location. Once the file is written the file written is closed in order to be able to save another file. An error is thrown if the filename is invalid or the pathname is incorrect, more generally if the program can not save the file, the program then exits the application. The diagram below shows the sequences that the program follows in order to save the file. We assume that no error occurs during the procedure of the save file.
Figure 28: The procedure that the program follows in order to save the router configuration file

Open file configuration

The open file configuration procedure is detailed below. In the same way that the save configuration file described before, an event listener checks if the user loads a Router Configuration file. If the user clicks on the load configuration file inside the Java swing menu, the code below is launched.

```java
private void jMenuItemOpenActionPerformed(java.awt.event.ActionEvent evt) {
    FileDialog fdLoad = new FileDialog(this, "Load Router Configuration File", FileDialog.LOAD);
    fdLoad.show();
    String FileNameLoad = fdLoad.getFile();
    String dir = fdLoad.getDirectory();
    try{
        File f = new File(dir, FileNameLoad);
        String CommandLoad = null;
        FileReader inputFile = new FileReader(f);
        turn = 1;            // enable the user mode
        String output = "\n";
        DisplayRouter.append(output);
        PromptModeUser();
        while (true){
            int i = inputFile.read();
            char c = (char) i;
            if(i == -1)
                if(c != '
'){
                    CommandLoad = String.valueOf(c);
                    jTextArea2.append(CommandLoad);
                }
            else{
                String CommandFromLoad = null;
                String ParseCommandLoad1 = null;
                String CommandFromLoadRest1 = null;
                int IntFirstCr = 0;
                int IntFirstCr2 = 0;
                int IntFirstCr3 = 0;
                CommandFromLoad = jTextArea2.getText();
                IntFirstCr = CommandFromLoad.indexOf("\n");
                IntFirstCr2 = IntFirstCr - 1;
                IntFirstCr3 = IntFirstCr + 1;
                ParseCommandLoad1 = CommandFromLoad.substring(0, IntFirstCr);
                DisplayRouter.append(ParseCommandLoad1);
                CommandFromLoadRest1 = CommandFromLoad.substring(IntFirstCr3);
                jTextArea2.setText(CommandFromLoadRest1);
                switch(turn){
```
Once the code is launched the open router configuration dialog appears on the screen. The program pauses until the user points to a file. The filename and the location where the file is located are stored in two variables. The method “try” is here to try if it is possible to open a file. If the program can not open the file an exception is thrown. A file reader method will point to the file in order to access information which is included in the configuration file. The user prompt is then added to the configuration panel (Router>) in order to be able to parse the different commands which are included in the file. Each character is then read individually. A ‘n’ means that this is the end of the command, and the command needs to be interpreted. The process continues until the last character return the value ‘-1’ which means that this is the end of the file and the program exits from the loop. The character is then tested and the command is added character by character to the configuration panel. Once the program detects a carriage return, the command needs to be interpreted. The program then enters into the else loop in order to parse the complete command onto the configuration panel and launch the interpretation. The program performs the extraction of the command included by selecting the text included between two carriage returns, saves it into a string and parses it into the configuration panel. Once the command is parsed, the switch statement launches the appropriate program according to the CLI prompt where the configuration is made. This process is the same as if the user pressed the enter command. This will not be examined again as this as been already viewed with the ‘enter key event’. The process is then followed until the program detects a character which has a ‘-1’ value which mean that this is the EOF (End Of File). The while loop is then exited, the open save file processes is then over.

4.6 Conclusion

This part of the report has highlighted the way that the program has been implemented related to the theory and the modelling / design part examined in the previous parts. As we saw in the implementation, Java is the programming language used, a justification of this choice as been highlighted by demonstrating the strength of this programming language. After describing within the theory part the models used for this program, the implementation of the different functions that the design represents have been studied. An account of the different libraries used for this program have been described, also the different types of declarations made in the context of this program.

The key event listener, has been viewed and demonstrates the step that the program follow when the user press the enter key. An example of a CLI prompt that the program displays has also been viewed such as the way of extracting and interpreted the commands performed.

An example of the emulation of an IOS commands has also been viewed. Finally, the extra parts added to the program used to enhance and facilitate the learning of router programming have been viewed. These parts represent the graphical representation of the commands interpreted by the user in order to improve feedback and learn, using this method, in an easier way to learn IOS commands. The facilities that allow the users to load and save file configuration have been also detailed.

Note that not all the commands implemented in the context of this program have been examined due to the length of the program. Nevertheless, the major implementations have been highlighted according to the model developed in the design / implementation part. The next part of the report will be dedicated to the tests performed using the software in order to demonstrate the validation of the requirements.
5 Tests/Evaluation

5.1 Introduction

This part of the report will demonstrate the test and evaluation performed on the Router Configuration software in order to show that the software developed does fit with the requirements. As has been said in the abstract, the software has been modelled by studying the behaviour of real router environments.

This study has been based on the observation and analysis of the Cisco laboratory available in Napier University. The test performed is to take the router configuration of each of the five routers and open them using the router emulator in order to demonstrate that the software can interpret all the commands used to configure these real routers.

The second test will be based on the achievement of the tutorial given in Advanced Distributed Systems and be able to perform the configuration of the router based on this tutorial. The functions of opening and saving the router configurations files and the graphical representation of the router will be viewed.

As it has been argued in the implementation part that Java is a platform independent language, a test will be performed on another operating system, Linux (Version Used Mandrake 8.1, Graphical environment Gnome, and Java Virtual Machine used: JDK IBM 1.3-8) in order to demonstrate that the software programmed is platform independent.

As the software is intended to enhance the router programming skills, typical users will be asked to attempt the tutorial of Advanced Distributed Systems and collect their thoughts based on monitoring evaluation and task performance. A comparison with other emulators existing on the market will be made based on different points: Usability, Easiness, Robustness, facilities offered, speed and price. A conclusion will be finally made in order to summarise the learning outcomes of the test / evaluation part.

5.2 Test based on the interpretation of real router configurations

The software realised has been modelled by studying the behaviour of real router environment, as the Cisco laboratory in Napier University provides a five router environments. The test performed is to take the router configuration of each of the five routers and open them using the router emulator in order to demonstrate that the software can interpret all the commands used to configure these real routers. After receiving the five router configurations by my supervisor I decided to use these configurations in order to validate the Router Configuration Emulator requirements and demonstrate that the software can handle a set of IOS commands by interpreting these configurations.

The topology of the Cisco Lab is presented in Figure 29. Each of the five routers holds a router configuration, in order to forward data packets according to the topology used. Each of the configuration has been tested using the Router Configuration Emulation, the screenshot below demonstrate that the router emulator has interpreted all the commands which are included inside the router configuration. The following will not show the screen shot for all the routers as the configuration of the different routers is similar. Therefore all the configurations have been tested and all of them are perfectly working using the emulator.
5.3 Test performed using the Advanced Distributed Systems tutorial on router configuration

The tutorial based on Advanced Distributed systems has also been performed, the tutorial has been followed and all the commands explained in this tutorial are fully working, furthermore the facilities that the Router Configuration Emulation software provide such as the file open and save have been tested and are working and also the graphical representation of the configuration performed by the user.

The possibility of taking a file saved by the emulator and downloading it into a real router is also possible as the files saved do not contain any extra information. The only information...
contained include the IOS commands interpreted and carriage return at the end of each commands which enable the router to detect the end of a command typed.

5.4 Java platform independent?

During the implementation it has been argued that Java has been selected and its strength has been highlighted. One of the strengths of Java is to be platform independent due to its Java Virtual Machine that makes the link between the hardware and the programs and leads to a platform independent programming language.

A test will be performed on another operating system. The operating system tested is Linux, the version used Mandrake 8.1 with the graphical environment Gnome, and Java Virtual Machine installed is JDK IBM 1.3-8.

The tests performed show that the program is working properly under Linux, the different files configurations (The five routers configuration files plus the Advanced Distributed systems tutorial file) have been opened and the program has interpreted all the commands included in the different files and the graphical environment has been drawn according to the configuration present inside the different files. The application works fine and the facility of loading the files and saved them is working. The graphical representation of the actual router configuration (The LAB_B router configuration has been tested) is working properly. Some slight changes appears in the resolution of the application and some of the text displayed are not complete but we can conclude that the application is fully compliant under Linux due to the platform independence offered by Java and all the facilities are working.

![Figure 31: The application running under Linux Mandrake](image-url)
5.5 Users evaluation

As the software is intended to enhance router programming skills, some typical user has been asked to attempt the tutorial of Advanced Distributed Systems. The results of this evaluation have been based on monitoring evaluation and task performance.

The idea is to distribute an evaluation sheet to the person which is going to make the test. An overview of general networking background has been explained to the users and also an overview of router configuration in order to perform the test.

Three persons from different backgrounds have selected for the evaluation. The three representatives are three users with different levels of networking background and router programming of the typical users for the software realised. The first user got a background in management. He is currently doing a Master in Information Technology. He has never programmed a router before and the networking basis needed to be explained. Therefore an interest for the router programming has been shown, that is why I have selected this person to be part of the evaluation. The second user has a computing background in software engineering; he has a good background in networking but has never programmed a router before. Finally, the last user is currently doing a computing honours degree, he has got a very good background in networking and he possesses a Cisco certification (Semester 1 and 2). He has already used emulators and real routers to perform routers configurations.

The details of the users have been asked and then a general overview of what the system is intended for has been given. In order to perform the evaluation in good conditions a basic networking overview and a router configuration explanation have been given.

A direct evaluation of the candidates has been done and the feelings of the user while performing the configuration have been noted. In order to evaluate the task performed a usability engineering method has been made and the time to perform the tutorial has been noted. A questionnaire survey and general comments have been added in order to receive the general feelings of the users concerning the applications. The evaluation sheets are included into the appendix.

The evaluation has shown that the intended users have all performed the tasks required for doing the tutorial in reasonable time. Some questions have been asked about the different IOS commands typed and the users managed to finish the tutorial without problems.

The questionnaire / survey part has shown that the Router Configuration Emulation is well noted according to the other emulators shown and the tasks required has been relatively easy for them to perform and the facilities that the program possess (Load / Save configuration files can improve the environment when router programming is learned and the graphical representation of the router programming help to understand the commands typed. The most confident of the three users in router programming would like that the software to present more IOS commands emulated. All the users argued that the software realised could improve by a great deal their programming skills mainly due to the graphical representation implemented.

5.6 Comparison with other router emulators.

Others router emulators currently exist on the market. Three emulators will be compared with the router configuration emulation software developed. The criteria of evaluation will be based on: (Usability, Ease of use, Robustness, facilities offered, efficiency and price).
The others router software emulators are: Boson Router Simulator from Boson Software, Inc., E-sim from Cisco and Sim from Kenn Thompson. The table below show the different emulators used for evaluating the software.

<table>
<thead>
<tr>
<th>Usability</th>
<th>Ease of use</th>
<th>Robustness</th>
<th>Facilities offered</th>
<th>Efficiency</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>N/A</td>
<td>✔️</td>
<td>Licence $100</td>
</tr>
<tr>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>x</td>
<td>✔️</td>
<td>Licensed</td>
</tr>
<tr>
<td>--</td>
<td>xx</td>
<td>xx</td>
<td>xx</td>
<td>--</td>
<td>Freeware</td>
</tr>
<tr>
<td>✔️ ✔️ ✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>Freeware</td>
</tr>
</tbody>
</table>

**Table 5:** Comparison of the Router Configuration Emulation software with existing products

The table above present the different criteria tested for the different software in comparison of the Router Configuration emulation software.

- **Boson Router Simulator:** This program is a professional tool which appears easy to use, is robust and offers ‘save’ and ‘load’ of the configuration file. Not all the IOS commands could be tested due to the high fee of the licence $100. The version tested is only a demo version.
- **E-sim:** E-sim is developed by Cisco and provides a good set of IOS commands which are sufficient to perform the different tutorials. Therefore the software does not implement any facilities for load/save configuration files and the topology shown is a simple static picture. The configuration is limited to the tutorials required for a given topology.
- **Sim:** This is freeware developed by an independent programmer. The software does not appear to provide a wide range of IOS emulated commands. This software is so slow that the tutorial of Advanced Distributed Systems has not been performed. It is difficult to enter a command, as it is sometimes needed to type two times the first letter of the command in order to be printed. Furthermore the software does not provide any facilities which could enhance the router configuration skills.
Router Configuration Emulation: According to the user’s evaluation and feedback the emulator does provides a reasonable set in order to perform the Advanced Distributed Systems tutorial. As this has been viewed before the software is also able to interpret the different router configuration included in the five routers of Cisco topology. The facilities provided, load and save configuration file help to test real router configurations and allow the users to save their configurations in order to implement them later on. The dynamic graphical representation of the router configuration performed, has been agreed by the evaluators to give the possibility to enhance router programming skills.

5.7 Conclusion

This partial conclusion will give an overview of the different learning outcomes of the tests / evaluation part. The different tests performed demonstrate that the Router Configuration Emulation software does fit with requirements. As a study has been based on the observation and analysis of the Cisco laboratory available in Napier University, the first test was to load the five configuration files of the different routers of the laboratory into the Router Configuration Emulation software.

The tests were achieved and the software handled all the commands included in the configuration files using the load/ save facilities and the graphical representation of the software has also been validated. The Advanced Distributed Systems tutorial based on router configurations has been achieved. The test was conclusive and validated the software requirements.

During the implementation part it has been argued, after highlighting the language strengths used for this application, that Java is a platform independent. A test to prove this statement has been made using Linux Mandrake 8.1 and a Java Virtual Machine JDK IBM 1.3-8, that the application was working properly under Linux. Some tests such as the file and load facilities have been performed, the interpretation of the IOS commands typed and finally the graphical representation of the actual configuration have also been experienced.

As the software is intended to enhance the router programming skills, three typical users have been asked to do the tutorial of Advanced Distributed Systems and collect their thoughts based on monitoring evaluation and task performance analysis.

Finally a comparison with other software emulators currently existing in the market have been viewed based on different criteria in order to demonstrate the strength of the software against other emulators currently on the market.
6 Conclusion

6.1 Appraisal of Achievement

This last part of the report is intended to make an appraisal of the achievement of the Router Configuration Emulation software. The theory part has been highlighted by giving a general background essential to understanding the implementation of the software developed. A background of networking has been highlighted by giving an account of the TCP and IP, the OSI model, the IP addressing and the new version of IP addresses IPv6 have been viewed according to different RFCs and academic papers published.

A more specific background has then been focused on the different routing protocols (RIP, IGRP, EIGRP and OSPF). The router fundamental has then been viewed by giving an account of the different specification of a router, such as boot, memory, router connection, passwords, router prompts. A comment on IOS has been given as this is the set of language used in order to configure a router.

The second part of this report has been focused on the design / modelling of the application built. The specification and analysis of the software has been viewed. Different engineering methods have been detailed and the justification for the method used, has also been viewed. The high level design of the emulator has been given in order to justify the implementation and highlight the global input / output of the software. A more specific analysis based on the component design has been given.

The implementation of the software realised has been given during the next part of the report. A justification of the selection of the programming language used has been made by giving the strengths of the language used. The component design model given in the previous part which is composed of the different methods implemented in the software as been used in order to understudy the implementation. The different methods implemented such as the way of displaying the different CLI prompts and the way of extracting and interpreting the command typed as also been viewed. An example of one of the IOS commands emulated has been viewed. A description of the implementation of facilities used to enhance the router programming skill have been developed such as the possibility of opening and saving files configurations and also the graphical representation of the router configuration performed.

The test part has been the last part of the report and the evaluation made on the Router Configuration Emulation software has been examined. Different router configurations of the Cisco laboratory available in Napier University have been loaded into the Router Configuration Emulation software in order to demonstrate that the software is able to interpret all the commands contained on these router configurations. The facilities of load / save file configuration have also been tested. As one of the strengths of Java, the programming language used for implementing this application, is to be platform independent, the previous tests have been performed under another operating system: Linux. The test validates that Java is platform independent as it has been demonstrated that the application is working perfectly under Linux, without the need to adapt the program to this new software architecture. As the software is intended to enhance the router programming skills some intended users have evaluate the software. The evaluation was conclusive as the Advanced Distributed Systems tutorial has been perform without major problems by all the evaluators. Finally, a comparison of the software with different products available on the market has been made in order to highlight the strength of the application.

The objectives stated in the introduction were to create an emulator of a router which implements major IOS (Internetwork operating system) commands, such as (Resolve the hostname, Configure the console, auxiliary telnet and executive passwords, affect the IP
addresses and subnet masks on the different ports of the router, configure routing tables and configure hostname tables)

The different objectives have been reached; furthermore more IOS commands have been implemented. The testing part of the application has demonstrated that the Router Configuration Emulation software was able to perform all of the IOS commands included into real router configuration. The different configuration have been loaded and interpreted by the router configuration emulation.

Another aspect of the emulator has been to produce facilities such as the possibility of loading and saving router configuration using the Router Configuration Emulation software. The other part that has been implemented in addition of the requirements was to produce a dynamic graphical representation of the configuration currently performed by the users. Among the different software tested, the Router Configuration Emulation is the only software which enables platform independence and this has been argued within the testing part that the dynamic graphical representation of the configuration and the extra facilities provided help router programming skills a great deal.

6.2 Suggestions for future work

This last part of the report is intended to state the possible improvements for future work on the Router Configuration Emulation software. A better H.C.I. design of the application could be made. H.C.I. is a recent science in computing. This is the study of how a user interacts with specific software. Professional H.C.I. tools such as monitoring users in a separate room using cameras, as the evaluators would be less conscious that they are monitored could be performed. This study could bring a new set of requirements by performing an analysis on a wider range and number of evaluators. A full set of IOS commands could be implemented for this software in order to produce a powerful emulator fully compliant with the latest versions of IOS produced by the different router manufacturers. As the networking and more generally computing area is evolving so fast that the software needs to adapt to these constant moving technologies by the different improvement and changes made in software and hardware architecture [2]. A possibility of defining a topology by adding more than one router or more networks components in order to create a virtual network can be investigated. Connections between the different elements can be made by improving the dynamic graphical representation implemented and be able to view the configuration made for a given topology. With this objective in mind, a presentation of the IP version 6 has been made as this is commonly agreed that this new format of IP addresses will be the next big step in the networking field, some hardware developed by major manufacturers are already available but they are most of the time reserved for researchers and very limited in number. Future work appears to focused on this emerging technologies and adapt the software accordingly. To conclude, this project has been really interesting as it has permitted me to develop practical knowledge taught during my degree by implementing this Router Configuration Emulation software; personal improvements in programming, networking, software development, Linux have been made thorough the development of this project.
7 Appendix A

The following gives a copy of the project Gantt chart.

<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>2002</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Honours Project: Router Configuration Emulation</td>
<td>99%</td>
</tr>
<tr>
<td>2</td>
<td>Project Dissertation</td>
<td>100%</td>
</tr>
<tr>
<td>3</td>
<td>Timetable Key Events</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Select a project topic &amp; supervisor</td>
<td>100%</td>
</tr>
<tr>
<td>5</td>
<td>Project Contract agreed &amp; signed by student &amp; supervisor &amp; second marker</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Project review with supervisor and second marker</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2nd semester start</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>A good draft of the project should have been submitted to your supervisor</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Final date for the submission of the project dissertation/Final presentation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Final date for the demonstration of the project</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Creation of the router interface</td>
<td>100%</td>
</tr>
<tr>
<td>11</td>
<td>Continuity of the Java applet (profects)</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Fixing problem of browser compatibility</td>
<td>100%</td>
</tr>
<tr>
<td>13</td>
<td>Learning the new software: Tools for Java</td>
<td>100%</td>
</tr>
<tr>
<td>14</td>
<td>Creation of the new router interface in Java Swing</td>
<td>100%</td>
</tr>
<tr>
<td>15</td>
<td>Interpret the typical command</td>
<td>100%</td>
</tr>
<tr>
<td>16</td>
<td>Implement enable commands, disable, logout</td>
<td>100%</td>
</tr>
<tr>
<td>17</td>
<td>Change the prompt aspect (executive prompt)</td>
<td>100%</td>
</tr>
<tr>
<td>18</td>
<td>Setting password for executive name</td>
<td>100%</td>
</tr>
<tr>
<td>19</td>
<td>Getting the IP addresses on the different ports of the routers</td>
<td>100%</td>
</tr>
<tr>
<td>20</td>
<td>Emulate command &quot;the shutdown&quot; to activate ports in address space</td>
<td>100%</td>
</tr>
<tr>
<td>21</td>
<td>Emulate clock rate</td>
<td>100%</td>
</tr>
<tr>
<td>22</td>
<td>Setting a routing protocol (RIP) + Investigate different type of routing.</td>
<td>100%</td>
</tr>
<tr>
<td>23</td>
<td>Configure the VTY (telnet) password</td>
<td>100%</td>
</tr>
<tr>
<td>24</td>
<td>Emulate the telnet command to login into the other routers</td>
<td>100%</td>
</tr>
<tr>
<td>25</td>
<td>Setting the hostname table with the IP of the different parts</td>
<td>100%</td>
</tr>
<tr>
<td>26</td>
<td>Emulate the commands (Show protocols,Show Host)</td>
<td>100%</td>
</tr>
<tr>
<td>27</td>
<td>Finalisation of the project</td>
<td>100%</td>
</tr>
<tr>
<td>28</td>
<td>Investigate further implementation of the others IOS commands &amp; emulate them</td>
<td>100%</td>
</tr>
<tr>
<td>29</td>
<td>Test of the router interface &amp; verify</td>
<td>100%</td>
</tr>
<tr>
<td>30</td>
<td>Creation of the project documentation</td>
<td>100%</td>
</tr>
<tr>
<td>31</td>
<td>Further ideas of implementations</td>
<td>100%</td>
</tr>
<tr>
<td>32</td>
<td>Extended IOS commands</td>
<td>100%</td>
</tr>
<tr>
<td>33</td>
<td>Load and save configuration files</td>
<td>100%</td>
</tr>
<tr>
<td>34</td>
<td>Graphical representation of the configuration entered</td>
<td>100%</td>
</tr>
<tr>
<td>35</td>
<td>Connection to a real router / Download the configuration file</td>
<td>100%</td>
</tr>
</tbody>
</table>
8 Appendix B

This is a copy of the Advanced Distributed System given for the evaluation of the program.
(Source: Advanced Distributed Systems.)

Advanced Distributed Systems Worksheet 1 (Router Programming)

Procedure

The basic tasks are:

- Set-up hostname. This uses the hostname command.
- Set-up IP addresses and subnet masks for each of the ports. This uses the ip address command.
- Set-up an executive password. This uses the enable secret command.
- Set-up login (console) and telnet (vty) passwords. This uses the password and login commands.
- Set-up a routing protocol. This uses the network command.
- Set-up a hosts name table. This uses the ip host command.
- Show the status of the ports, and the running configuration. The command used for these are show protocols and show running-config.

Programming Router A

Getting in privileged mode The router has two main modes:

- User mode. This is the initial mode that the user goes into when they log onto the router. In this mode it is not possible to configure the router, and it is only possible to perform simple commands such as telnet and ping.
- Executive mode. In this mode the full range of commands can be used, and the router can be programmed. The command which is used to go from user mode into executive mode is enable. If a password is set for the executive mode, the user must enter this before they can enter into the executive mode. The prompt should change from a ‘>’ to a ‘#’.

The following gives an example:

```
Router> enable
Router#
```

If at any time you cannot remember the syntax of a command type-in the command and then press the ‘?’ key.

Setting hostname

The hostname is set using the hostname command. This name is reflected in the prompt of the router, and makes it easier to identify the current router.

For example to set the host of LAB_A to LAB_A:
Honours Project (CO42019) Report – Router Configuration Emulation –

Router# config t
Enter configuration commands, one per line. End with END.
Router (config)# hostname LAB_A
LAB_A (config)#

Setting the secret password
The secret password is used to enter into the executive mode. In the following case the password is defined as class.

LAB_A (config)# enable secret class
LAB_A (config)# exit
LAB_A# exit
LAB_A> enable
Password: ccc
Password: class
LAB_A#.

Setting the IP addresses on the ports
One of the most important things to set on the router is the IP address of each of the ports. These ports will be used as gateways out of the network segment to which they connect to. The interface command (or int for short) programs each of the interfaces.

In the following example the three ports on the router are programmed with the required IP addresses, and subnet masks. The ports will not automatically come on-line, and will start in a shutdown mode. Thus the no shutdown command is used to start them up.

LAB_A# config t
LAB_A (config)# int e0
LAB_A (config-if)# ip address 192.5.5.1 255.255.255.0
LAB_A (config-if)# no shutdown
LAB_A (config-if)# exit
LAB_A (config)# int e1
LAB_A (config-if)# ip address 205.7.5.1 255.255.255.0
LAB_A (config-if)# no shutdown
LAB_A (config-if)# exit
LAB_A (config)# int s0
LAB_A (config-if)# ip address 201.100.11.1 255.255.255.0
LAB_A (config-if)# clock rate 56000
LAB_A (config-if)# no shutdown
LAB_A (config-if)# exit

Setting a routing protocol
The router will not be able to connect to other routers unless it runs a routing protocol which is the same as the other routers. This allows the routers to determine the best path to a remote device. In this case the RIP protocol is set-up with the router rip command, and then each of the networks in which the router will broadcast its routing table are defined (using the network command):

LAB_A (config)# router rip
LAB_A (config-router)# network 192.5.5.0
LAB_A (config-router)# network 205.7.5.0
LAB_A (config-router)# network 201.100.11.0
LAB_A (config-router)# exit
LAB_A (config)#

Setting the line console password
Passwords are important in providing a degree of security to the router. There are three main passwords to set: the executive password, the console password, and the remote login (vty) password. The vty password defines the telnet password, and the console password defines the initial login password.
Setting the hostnames table
It is often difficult to remember the IP address of each of the ports, thus a router can be setup with its own hosts table, as shown next:

| LAB_A (config)# ip host LAB_A 192.5.5.1 205.7.5.1 201.100.11.1 |
| LAB_A (config)# ip host LAB_B 201.100.11.2 219.17.100.1 199.6.13.1 |
| LAB_A (config)# ip host LAB_C 223.8.151.1 204.204.7.1 199.6.13.1 |
| LAB_A (config)# ip host LAB_D 210.93.105.1 204.204.7.2 |
| LAB_A (config)# ip host LAB_E 210.93.105.2 |
| LAB_A (config)# exit |

Determining if the ports are operating
To see if the ports are up, and if they are connected to the other routers, the show protocols command is used. For example:

| LAB_A# show protocols |
| E0 is up, line protocol is up |
| Internet address is 192.5.5.1 |
| E1 is up, line protocol is up |
| Internet address is 205.7.5.1 |
| S0 is up, line protocol is up |
| Internet address is 201.100.11.1 |

The hosts table can also be viewed with show hosts.
This following is an example of a real router configuration that the Router Configuration Emulation software can interpret:

```plaintext
(Router configuration : LAB_A)
enable
cfgt
hostname LAB-A
enable secret class

terface Ethernet0
ip address 192.5.5.1 255.255.255.0
no shutdown
exit
terface Ethernet1
ip address 205.7.5.1 255.255.255.0
no shutdown
exit
terface Serial0
ip address 201.100.11.1 255.255.255.0
clockrate 56000
no shutdown
exit
terface Serial1
no ip address
shutdown
exit
router rip
network 192.5.5.0
network 205.7.5.0
network 201.100.11.0
exit
ip host LAB-B 201.100.11.2 219.17.100.1 199.6.13.1
ip host LAB-C 199.6.13.2 223.8.151.1 204.204.7.1
ip host LAB-D 204.204.7.2 210.93.105.1
ip host LAB-E 210.93.105.2
ip host LAB-A 192.5.5.1 205.7.5.1 201.100.11.1
no ip classless
line con 0
password cisco
login
exit
line aux 0
exit
line vty 0 4
password cisco
login
exit
exit
```
10 Appendix D

The following pages define the evaluation documents.
Evaluation
Router Configuration Emulation software

Please complete the details below:

First Name: ... Occupational Title: ... 
Surname: ... Course: ... 

1) Direct Evaluation:
(Part reserved for the evaluator)
- The evaluation has been performed without any problems. Some questions about specific configuration commands have been made. The evaluator agreed that the software developed can improve route programmation skills.

2) Usability Engineering:

Please complete the Advanced Distributed Systems tutorial using the Router Configuration Emulation software.
A general overview of router configurations and basic networking explanations have been given to the users before performing the tutorial using the software developed.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Measuring Concept</th>
<th>Measuring Method</th>
<th>Worst Case</th>
<th>Plan Level</th>
<th>Best Case</th>
<th>Now Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Distributed system based on the router configuration tutorial</td>
<td>task</td>
<td>See usability engineering part 2</td>
<td>1 hour</td>
<td>20 min</td>
<td>10 min</td>
<td>5.7 min</td>
</tr>
</tbody>
</table>

3) Questionnaire / survey
Please complete the following: (Tick your answer)
- Compared to the other software presented (E-sim, Boson Emulator, sim) how would you classify the Router Configuration Emulation software from 1 (best) to 4 (worst)?
- Do you think that the software developed can perform all the tasks presented in this tutorial?
  - Strongly agree □ agree □ Neutral □ Disagree □ Strongly disagree □
  - Do the facilities provided (Load and Save files configuration and the graphical representation of the software) enhance your router programming skills?
    - Strongly agree □ agree □ Neutral □ Disagree □ Strongly disagree □

Please write you comments below to inform me how the system can enhance your router programming skills and state your general comments about the program developed.

Nice interface. Help with improving...
Evaluation
Router Configuration Emulation software

Please complete the details below:

First Name: ………………………………………………………………………… Occupation: ………………………………………………………………………
Surname: …………………………………………………………………………… Course: ……………………………………………………………………………

1) Direct Evaluation (Part reserved for the evaluator)
   - Some questions have been asked in order to clarify specific points of the router programming tools. The user does not have a specific computing background but the task required has been performed within a normal time.

2) Usability engineering

Please complete the Advanced Distributed Systems tutorial using the Router Configuration Emulation software.
A general overview of router configurations and basic networking explanations have been given to the users before performing the tutorial using the software developed.

(Part reserved for the evaluator )

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Measuring Concepts</th>
<th>Measuring Method</th>
<th>Worst Case</th>
<th>Plan Level</th>
<th>Best Case</th>
<th>New Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Distributed system based on the router configuration tutorial</td>
<td>task</td>
<td>See usability engineering part 2</td>
<td>1 hour</td>
<td>20 mn</td>
<td>10 mn</td>
<td>3.5 mn</td>
</tr>
</tbody>
</table>

3) Questionnaire / survey
   Please complete the following: (Tick your answer)
   - Compared to the other software presented (E-sim, Boson Emulator, sim) how would you classify the Router Configuration Emulation software from (1 (best)) to 4 (worst)?

   - Do you think that the software developed can perform all the tasks presented in this tutorial?
     Strongly agree ☐ agree ☐ Neutral ☐ Disagree ☐ Strongly disagree ☐

   - Do the facilities provided (Load and Save files configuration and the graphical representation of the software) enhance your router programming skills?
     Strongly agree ☐ agree ☐ Neutral ☐ Disagree ☐ Strongly disagree ☐

Please write your comments below to inform me how the system can enhance your router programming skills and state your general comments about the program developed.

...
Honours Project (CO42019) Report – Router Configuration Emulation –

Evaluation
Router Configuration Emulation software

Please complete the details below:

First Name: *Thierry*  
Surname: Thierry

Occupation: Student
Course: BSc. Computing

1) Direct Evaluation:

(Part reserved for the evaluator)

- The task required has been performed without any problems hence the evaluator got a certain confidence in

- route programming based on previous experience

2) Usability engineering:

Please complete the Advanced Distributed Systems tutorial using the Router Configuration Emulation software.

A general overview of router configurations and basic networking explanations have been given to the users before performing the tutorial using the software developed.

(Part reserved for the evaluator)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Measuring Concept</th>
<th>Measuring Method</th>
<th>Worst Case</th>
<th>Plan Level</th>
<th>Best Case</th>
<th>Now Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Distributed system based on the router configuration tutorial</td>
<td>task</td>
<td>See usability engineering part 2</td>
<td>1 hour</td>
<td>20 min</td>
<td>10 min</td>
<td>12 min</td>
</tr>
</tbody>
</table>

3) Questionnaire / survey

Please complete the following: (Tick your answer)

- Compared to the other software presented (E-sim, Boson Emulator, sim) how would you classify the Router Configuration Emulation software from 1 (best) to 4 (worst)?

- Do you think that the software developed can perform all the tasks presented in this tutorial?

  - Strongly agree: ☑
  - Agree: ☐
  - Neutral: ☐
  - Disagree: ☐
  - Strongly disagree: ☐

- Do the facilities provided (Load and Save files configuration and the graphical representation of the software) enhance your router programming skills?

  - Strongly agree: ☑
  - Agree: ☐
  - Neutral: ☐
  - Disagree: ☐
  - Strongly disagree: ☐

Please, write you comments below to inform me, how the system can enhance your router programming skills and state your general comments about the program developed.

The system provided a good opportunity to learn about the main

Some... on... the... commands... were... not easy... to... be... programmed... to

Some... on... the... commands... were... not easy... to... be... programmed... to

become... the... best... of... all... the... ones... that... were... tested.
11 Appendix E

This is the partial code shown below in order to demonstrate the shape of the program implemented. **The entire code is not included** as this contains over 3500 lines of code and it would be too large to be included in this report.

```java
private void jMenuItemOpenActionPerformed(java.awt.event.ActionEvent evt) {
    FileDialog fdLoad = new FileDialog(this, "Load Router Configuration File", FileDialog.LOAD);
    fdLoad.show();
    String FileNameLoad = fdLoad.getFile();
    String dir = fdLoad.getDirectory();
    try{
        File f = new File(dir, FileNameLoad);
        String CommandLoad = null;
        FileReader inputFile = new FileReader(f);
        // enable the user mode
        turn = 1;
        String output = "" + '
';
        DisplayRouter.append(output);
        PromptModeUser();
        while (true){
            int i = inputFile.read();
            char c = (char) i;
            if(i == -1)
                break;
            if(c != '
'){
                CommandLoad = String.valueOf(c);
                jTextArea2.append(CommandLoad);
            }
            else{
                jTextArea2.append("" + '
');
                String CommandFromLoad = null;
                String ParseCommandLoad1 = null;
                String CommandFromLoadRest1 = null;
                int IntFirstCr = 0;
                int IntFirstCr2 = 0;
                int IntFirstCr3 = 0;
                CommandFromLoad = jTextArea2.getText();
                IntFirstCr = CommandFromLoad.indexOf('"\n');
                IntFirstCr2 = IntFirstCr - 1;
                IntFirstCr3 = IntFirstCr + 1;
                ParseCommandLoad1 = CommandFromLoad.substring(0, IntFirstCr);
                DisplayRouter.append(ParseCommandLoad1);
                CommandFromLoadRest1 = CommandFromLoad.substring(IntFirstCr3);
                jTextArea2.setText(CommandFromLoadRest1);
                switch(turn){
                    case 0 : /* initialization */
                        turn = 1; /* enable the user mode */
                        PromptModeUser();
                        break;
                    case 1 : /* user mode */
                        InterpretCommand();
                        break;
                    case 2 : /* privileged mode */
                        InterpretCommandPrivileged();
                        break;
                    case 3 : /* privileged mode */
                        GlobalConfigCommand();
                        break;
                    case 4 : /* Executive Password Prompt */
                        break;
                }
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

ExecutivePassPromptCommand();
break;

case 5 : /* Interface Config Prompt */
   InterfaceConfigCommand();
   break;

case 6 : /* Interface Config Prompt */
   RouterConfigCommand();
   break;

case 7 : /* Line Config Prompt */
   LineConfigCommand();
   break;

case 8 : /* Console Password Check */
   ConsolePassPromptCommand();
   break;
}
}
}
}
}

}  
}
catch (IOException e) {
   System.err.println("File Error:" + e.toString());
   System.exit(1);
}

private void jMenuItemSaveActionPerformed(java.awt.event.ActionEvent evt){
   String FileNameSave;
   OpenFileDialog fd = new OpenFileDialog(this, "Save Router Configuration File",
   OpenFileDialog.SAVE);
   fd.show();
   FileNameSave = fd.getFile();
   String dir = fd.getDirectory();
   try{
      File f = new File(dir, FileNameSave);
      FileWriter fw = new FileWriter(f);
      fw.write(Test.getText());
      fw.close();
   } catch (IOException e) {
      System.err.println("File Error:" + e.toString());
      System.exit(1);
   }
}

public void GlobalConfig(){
   // Display the global config prompt Router(config)#
   String PromptAdd= Hostname + "(config)#";
   DisplayRouter.append(PromptAdd);
   DisplayRouter.setCaretPosition(DisplayRouter.getText().length());
}

public void InterpretCommandPrivileged(){
   // Display the global config prompt Router(config)#
   String PromptAdd= Hostname + "(config)#";
   DisplayRouter.append(PromptAdd);
   DisplayRouter.setCaretPosition(DisplayRouter.getText().length());
}

public void InterpretCommandPrivileged(){
   String AllText = DisplayRouter.getText();
   /* Calculate the length of the string */
   int StringLength = AllText.length();
   /* Calculate the beginning of the command typed */
   int BeginOfCommand = AllText.lastIndexOf("#");
   int BeginOfCommand2 = BeginOfCommand +1;
   DisplayRouter.select(BeginOfCommand2,StringLength);
   /* Take the selection of the text and save it as a string */
   String Command = DisplayRouter.getSelectedText();
   HashMap PrivilegedCommand = new HashMap();
   String RelationPrivilegedHashMap;
   // Commands emulated for the privileged mode
   PrivilegedCommand.put("disable", "0");
   PrivilegedCommand.put("logout", "1");
// if the user has only pressed the enter key without including any commands
PrivilegedCommand.put(null, "2");
// Simply jump a line and display the privileged mode
PrivilegedCommand.put("enable", "2");
PrivilegedCommand.put("exit", "1"); //exit the router console
PrivilegedCommand.put("?", "4");
PrivilegedCommand.put("help", "33");
PrivilegedCommand.put("config t", "5");
PrivilegedCommand.put("config terminal", "5");
PrivilegedCommand.put("show running-config", "6");
PrivilegedCommand.put("sh running-config", "6");
PrivilegedCommand.put("show run", "6");
PrivilegedCommand.put("sh run", "6");
PrivilegedCommand.put("sh protocols", "7");
PrivilegedCommand.put("sh protocols", "7");
PrivilegedCommand.put("show hosts", "8");
PrivilegedCommand.put("sh hosts", "8");
PrivilegedCommand.put("show interface", "9");
PrivilegedCommand.put("sh interface", "9");
PrivilegedCommand.put("show interfaces", "9");
PrivilegedCommand.put("sh interfaces", "9");

RelationPrivilegedHashMap = (String)PrivilegedCommand.get(Command);
/* Check if the command match with the hashmap
 * if the command does not match with it, the case number is 
 * affected to display an error message on the router interface 
 */
if (RelationPrivilegedHashMap == null) {
    RelationPrivilegedHashMap = "404";
}
// Hash map, like a vector can not store primitives data types it 
// store some instance of object. The solution is to use a "warpper" class 
// in this case we need to tranform the string RelationUserhHashMap into 
// integer to match with the switch statement.
int RelationPrivilegedHashMapInt;
RelationPrivilegedHashMapInt = Integer.parseInt(RelationPrivilegedHashMap);
switch (RelationPrivilegedHashMapInt) {
    case 0 :
        Test.append(Command + '\n');
        String output = "The command typed is:" + Command + '\n';
        DisplayRouter.append(output);
        PromptModeUser();
        turn = 1;
        break;
    case 1:
        turn = 0;
        String InitRouter = "Router con0 is now available" + '\n' + "" + '\n' + "";
        String InitRouter2 = "Press return to get started" + '\n' + "" + '\n' + "";
        DisplayRouter.setText(InitRouter + InitRouter2);
        if ((ConsolePassword != "") && (LoginConsole.equals("Activated"))) {
            Test.append(Command + '\n');
            turn = 8; //Console password prompt
            ConsolePassPrompt();
        } else {
            Test.append(Command + '\n');
            Test.append("" + '\n');
            turn = 0;
        }
        break;
    case 2 :
        String output1 = "" + '\n';
        DisplayRouter.append(output1);
        PrivilegedMode();
        turn = 2;
        break;
    case 4:
Test.append(Command + '\n');
String QuestionMarkPrivi = "" + '\n' +
"clear" + ' ' + "Reset functions" + '\n' +
"clock" + ' ' + "Manage the system clock" + '\n' +
"configure" + ' ' + "Enter configuration mode" + '\n' +
"copy" + ' ' + "Copy from one file to another" + '\n' +
"disable" + ' ' + "Turn off privileged commands" + '\n' +
"disconnect" + ' ' + "Disconnect an existing network connection" + '\n' +
"enable" + ' ' + "Turn on privileged commands" + '\n' +
"erase" + ' ' + "Erase a filesystem" + '\n' +
"exit" + ' ' + "Exit from the EXEC" + '\n' +
"help" + ' ' + "Description of the interactive help system" + '\n' +
"logout" + ' ' + "Exit from the EXEC" + '\n' +
"ping" + ' ' + "Send echo messages" + '\n' +
"reload" + ' ' + "Halt and perform a cold restart" + '\n' +
"show" + ' ' + "Show running system information" + '\n' +
"telnet" + ' ' + "Open a telnet connection" + '\n' +
"terminal" + ' ' + "Set terminal line parameters" + '\n' +
"traceroute" + ' ' + "Trace route to destination" + '\n' +
"write" + ' ' + "Write running configuration to memory, network, or terminal " + '\n' +
DisplayRouter.append(QuestionMarkPrivi);
PrivilegedMode();
turn = 2;
break;
case 33:
Test.append(Command + '\n');
String outputHelpPri = "" + '\n' +
"Help may be requested at any point in a command by entering" + '\n' +
"a question mark '?' or nothing matches, the help list will" + '\n' +
"be empty and you must backup until entering a '?' shows the" + '\n' +
"available options." + '\n' +
"Two styles of help are provided:" + '\n' +
"1. Full help is available when you are ready to enter a" + '\n' +
"command argument (e.g. 'show ?') and describes each possible" + '\n' +
"argument." + '\n' +
"2. Partial help is provided when an abbreviated argument is entered" + '\n' +
"and you want to know what arguments match the input" + '\n' +
"(e.g. 'show pr?')." + '\n' +
DisplayRouter.append(outputHelpPri);
PrivilegedMode();
turn = 2;
break;
case 5:
Test.append(Command + '\n');
String ConfigT = "" + '\n' +
"Enter configuration commands, one per line. End with CNTL/Z." + '\n' +
DisplayRouter.append(ConfigT);
GlobalConfig();
turn = 3;
break;
case 6:
Test.append(Command + '\n');
String RunningConfig = "" + '\n' +
"Building configuration..." + '\n' +
"Current configuration:" + '\n' +
"12.0" + '\n' +
"service timestamps debug uptime" + '\n' +
"service timestamps log uptime" + '\n' +
"no service password-encryption" + '\n' +
"hostname Router" + '\n' +
DisplayRouter.append(RunningConfig);
if(Hostname.equals("Router")){
  String RunningConfig2 = "hostname Router" + '\n' +
  "hostname Router" + '\n' +
  DisplayRouter.append(RunningConfig2);
}
else{
  String RunningConfig3 = "hostname " + Hostname + '\n' +
  "hostname " + '\n' +
  DisplayRouter.append(RunningConfig3);
String RunningConfig4 = "enable password" + '
' + "!
" + '
';
DisplayRouter.append(RunningConfig4);
if(ExecutivePassword.equals("")) {
    String RunningConfig5 = "!
" + '
';
    DisplayRouter.append(RunningConfig5);
} else {
    String RunningConfig6 = "enable secret 5 $1$0/yw$toqA0xRicty8gh7pm06F5b" + '
' + "!
" + '
';
    DisplayRouter.append(RunningConfig6);
}
String RunningConfig7 = "ip subnet-zero" + '
';
DisplayRouter.append(RunningConfig7);
if(HostnamesTableName.equals("")) {
    String RunningConfig8 = "!
" + '
' + "!
" + '
' + "!
" + '
';
    DisplayRouter.append(RunningConfig8);
} else {
    String RunningConfig9 = "ip host " + HostnamesTableName + " " + HostnamesTableIp1 + " " + HostnamesTableIp2 + " " + HostnamesTableIp3 + " " + HostnamesTableIp4 + '
';
    DisplayRouter.append(RunningConfig9);
    if(HostnamesTableName_2.equals("")) {
        String RunningConfig10 = "!
" + '
';
        DisplayRouter.append(RunningConfig10);
    } else {
        String RunningConfig11 = "ip host " + HostnamesTableName_2 + " " + HostnamesTableIp1_2 + " " + HostnamesTableIp2_2 + " " + HostnamesTableIp3_2 + " " + HostnamesTableIp4_2 + '
';
        DisplayRouter.append(RunningConfig11);
        if(HostnamesTableName_3.equals("")) {
            String RunningConfig12 = "!
" + '
';
            DisplayRouter.append(RunningConfig12);
        } else {
            String RunningConfig13 = "ip host " + HostnamesTableName_3 + " " + HostnamesTableIp1_3 + " " + HostnamesTableIp2_3 + " " + HostnamesTableIp3_3 + " " + HostnamesTableIp4_3 + '
';
            DisplayRouter.append(RunningConfig13);
            if(HostnamesTableName_4.equals("")) {
                String RunningConfig14 = "!
" + '
' + "!
" + '
' + "!
" + '
';
                DisplayRouter.append(RunningConfig14);
            } else {
                String RunningConfig15 = "interface Ethernet0" + '
';
                DisplayRouter.append(RunningConfig15);
                if(InternetAddressE0.equals("")) {
                    String RunningConfig16 = "no ip address" + '
';
                    DisplayRouter.append(RunningConfig16);
                } else {
                    String RunningConfig17 = "ip address " + InternetAddressE0 + " " + SubnetMaskE0 + '
';
                    DisplayRouter.append(RunningConfig17);
                }
            }
        }
    }
    String RunningConfig18 = "shutdown" + '
';
    DisplayRouter.append(RunningConfig18);
} else {
    String RunningConfig19 = "!
" + '
';
    DisplayRouter.append(RunningConfig19);
String RunningConfig19b = "interface Ethernet1" + '\n';
DisplayRouter.append(RunningConfig19b);
if(InternetAddressE1.equals("")){
    String RunningConfig20 = "no ip address" + '\n';
    DisplayRouter.append(RunningConfig20);
} else{
    String RunningConfig21 = "ip address " +
    InternetAddressE1 + " " + SubnetMaskE1 + '\n';
    DisplayRouter.append(RunningConfig21);
}
if(InterfaceE1State.equals("Down")){
    String RunningConfig22 = "shutdown" + '\n' + "!" + '\n';
    DisplayRouter.append(RunningConfig22);
} else{
    String RunningConfig23 = "!" + '\n';
    DisplayRouter.append(RunningConfig23);
}
String RunningConfig23b = "interface Serial0" + '\n';
DisplayRouter.append(RunningConfig23b);
if(InternetAddressS0.equals("")){
    String RunningConfig24 = "no ip address" + '\n';
    DisplayRouter.append(RunningConfig24);
} else{
    String RunningConfig25 = "ip address " +
    InternetAddressS0 + " " + SubnetMaskS0 + '\n';
    DisplayRouter.append(RunningConfig25);
}
if(InterfaceS0State.equals("Down")){
    String RunningConfig26 = "shutdown" + '\n' + "!" + '\n';
    DisplayRouter.append(RunningConfig26);
} else{
}
if(ClockRateS0.equals("")){
}
else{
    String RunningConfig28 = "Clock rate " + ClockRateS0 +
    "!" + '\n';
    DisplayRouter.append(RunningConfig28);
}
String RunningConfig23bb = "interface Serial1" + '\n';
DisplayRouter.append(RunningConfig23bb);
if(InternetAddressS1.equals("")){
    String RunningConfig24b = "no ip address" + '\n';
    DisplayRouter.append(RunningConfig24b);
} else{
    String RunningConfig25b = "ip address " +
    InternetAddressS1 + " " + SubnetMaskS1 + '\n';
    DisplayRouter.append(RunningConfig25b);
}
if(InterfaceS1State.equals("Down")){
    String RunningConfig26b = "shutdown" + '\n' + "!" + '\n';
    DisplayRouter.append(RunningConfig26b);
} else{
}
if(ClockRateS1.equals("")){
}
else{
    String RunningConfig28b = "Clock rate " + ClockRateS1 + '\n';
    DisplayRouter.append(RunningConfig28b);
}
if(RoutingRipTableNetwork1.equals("")){
    String RunningConfig29 = "!" + '\n' + "!" + '\n' + "!" + '\n';
    DisplayRouter.append(RunningConfig29);
} else{
    String RunningConfig30 = "Router rip" + '\n' +
    "network " + RoutingRipTableNetwork1 + '\n';
    DisplayRouter.append(RunningConfig30);
if(RoutingRipTableNetwork2.equals("")){
}
else{
    String RunningConfig31 = "network " + RoutingRipTableNetwork2 + '\n';
    DisplayRouter.append(RunningConfig31);
}
if(RoutingRipTableNetwork3.equals("")){
    String RunningConfig31b = "!" + '\n';
    DisplayRouter.append(RunningConfig31b);
}
else{
    String RunningConfig32 = "network " + RoutingRipTableNetwork3 + '\n' + "!" + '\n';
    DisplayRouter.append(RunningConfig32);
}
String RunningConfig33 = "ip classless " + "!" + '\n' + "!" + '\n' + "line con 0" + "!" + "transport input none" + '\n';
DisplayRouter.append(RunningConfig33);
if(ConsolePassword.equals("")){
}
else{
    String RunningConfig34 = "password " + ConsolePassword + '\n';
    DisplayRouter.append(RunningConfig34);
}
if(LoginConsole.equals("")){
}
else{
    String RunningConfig35 = "login" + '\n';
    DisplayRouter.append(RunningConfig35);
}
String RunningConfig36 = "line aux 0" + '\n';
DisplayRouter.append(RunningConfig36);
if(AuxPassword.equals("")){
}
else{
    String RunningConfig37 = "password " + AuxPassword + '\n';
    DisplayRouter.append(RunningConfig37);
}
if(LoginAux.equals("")){
}
else{
    String RunningConfig38 = "login" + '\n';
    DisplayRouter.append(RunningConfig38);
}
String RunningConfig39 = "line vty 0 4" + "session-timeout 60" + '\n';
DisplayRouter.append(RunningConfig39);
if(VtyPassword.equals("")){
}
else{
    String RunningConfig40 = "password " + VtyPassword + '\n';
    DisplayRouter.append(RunningConfig40);
}
if(LoginVty.equals("")){
}
else{
    String RunningConfig41 = "login" + '\n';
    DisplayRouter.append(RunningConfig41);
}
String RunningConfig42 = "!" + '\n' + "end" + '\n';
DisplayRouter.append(RunningConfig42);
PrivilegedMode();
turn = 2;
break;

case 7:
    Test.append(Command + '\n');
    shProtocols();
    PrivilegedMode();
    turn = 2;
    break;

case 8:
    Test.append(Command + '\n');
    shHosts();
    PrivilegedMode();
Honours Project (CO42019) Report – Router Configuration Emulation –

```java

    turn = 2;
    break;

    case 9:
        Test.append(Command + '
');
        shInterfaces();
        PrivilegedMode();
        turn = 2;
        break;
    case 404:
        String output2 = "" + '
' + "% Unknown command or computer name, or unable to find computer address" + '
';
        DisplayRouter.append(output2);
        PrivilegedMode();
        turn = 2;
        break;
    }

    public void ExecutivePassPromptCommand(){
        String AllTextExecPass = DisplayRouter.getText();
        /* Calculate the Length of the string */
        int StringLengthExecPass = AllTextExecPass.length();
        /* Calculate the beginning of the command typed */
        int BeginOfCommandExecPass = AllTextExecPass.lastIndexOf(":");
        int BeginOfCommand2ExecPass = BeginOfCommandExecPass + 1;
        DisplayRouter.select(BeginOfCommand2ExecPass, StringLengthExecPass);
        /* Take the selection of the text and save it as a string */
        String CommandExecPass = DisplayRouter.getSelectedText();
        if(CommandExecPass.equals(ExecutivePassword)){
            Test.append(CommandExecPass + '
');
            turn = 2;
            PasswordExecutiveCount = 0; //reset the number of tries
            String output ='
' + "The password is correct" + '
';
            DisplayRouter.append(output);
            PrivilegedMode();
        }
        else{
            turn = 4;
            PasswordExecutiveCount = PasswordExecutiveCount + 1;
            if(PasswordExecutiveCount != 3){
                String output ="" + '
';
                DisplayRouter.append(output);
                ExecutivePassPrompt();
                ExecutivePassPromptCommand();
            }
            if(PasswordExecutiveCount == 3){
                PasswordExecutiveCount = 0; //reset the number of tries
                String output ="n"+ "%Bad Password" + '
';
                DisplayRouter.append(output);
                InterpretCommand();
            }
        }
    }
```

12 Appendix F

The following are the supported Router Configuration Emulation commands.

User mode (Router>)

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable</td>
</tr>
<tr>
<td>en</td>
</tr>
<tr>
<td>logout</td>
</tr>
<tr>
<td>null</td>
</tr>
<tr>
<td>?</td>
</tr>
<tr>
<td>help</td>
</tr>
<tr>
<td>exit</td>
</tr>
<tr>
<td>disable</td>
</tr>
<tr>
<td>sh protocols</td>
</tr>
<tr>
<td>show protocols</td>
</tr>
<tr>
<td>sh hosts</td>
</tr>
<tr>
<td>show hosts</td>
</tr>
<tr>
<td>show interfaces</td>
</tr>
<tr>
<td>sh interface</td>
</tr>
<tr>
<td>sh interfaces</td>
</tr>
<tr>
<td>sh interface</td>
</tr>
</tbody>
</table>

Privileged mode (Router#)

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>disable</td>
</tr>
<tr>
<td>logout</td>
</tr>
<tr>
<td>null</td>
</tr>
<tr>
<td>enable</td>
</tr>
<tr>
<td>exit</td>
</tr>
<tr>
<td>?</td>
</tr>
<tr>
<td>help</td>
</tr>
<tr>
<td>config t</td>
</tr>
<tr>
<td>config terminal</td>
</tr>
<tr>
<td>show running-config</td>
</tr>
<tr>
<td>show run</td>
</tr>
<tr>
<td>sh run</td>
</tr>
<tr>
<td>sh protocols</td>
</tr>
<tr>
<td>show protocols</td>
</tr>
<tr>
<td>sh hosts</td>
</tr>
<tr>
<td>show hosts</td>
</tr>
<tr>
<td>show interfaces</td>
</tr>
<tr>
<td>sh interface</td>
</tr>
<tr>
<td>show interface</td>
</tr>
</tbody>
</table>

Global config command (Router(config)#)

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
</tr>
<tr>
<td>?</td>
</tr>
<tr>
<td>help</td>
</tr>
<tr>
<td>exit</td>
</tr>
<tr>
<td>hostname</td>
</tr>
<tr>
<td>int (e0, Ethernet 0, e1, Ethernet 1, s0 Serial 0, s1 Serial 1)</td>
</tr>
<tr>
<td>interface (e0, Ethernet 0, e1, Ethernet 1, s0 Serial 0, s1 Serial 1)</td>
</tr>
<tr>
<td>router (rip)</td>
</tr>
<tr>
<td>line (con 0, aux 0, vty 0 4)</td>
</tr>
<tr>
<td>Line (con 0, aux 0, vty 0 4)</td>
</tr>
<tr>
<td>Ip</td>
</tr>
<tr>
<td>No shutdown</td>
</tr>
</tbody>
</table>

Executive Password Prompt / Console Password Prompt
Interpret the password typed & take account of wrong passwords.

Interface Config (Router(config-if)#)

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
</tr>
<tr>
<td>Help</td>
</tr>
<tr>
<td>?</td>
</tr>
</tbody>
</table>
exit
ip (address X.X.X.X subnet)
no shutdown
shutdown
clock (1200 – 4000000)

Router Config (Router(config-router)#)
Null
Help
?
exit
network (X.X.X.X (Ip @)

Line Config (Router(config-line)#)
Null
Help
?
exit
password
login
13 Bibliography and references

Books:
[B9] Software Management Guidebook, Nasa GB_001_96 Washington D.C.

Academic publications:
[5] TCP flow control technique for an interworking interface: hardware implementation, Ridha Ouni a,1 , Adel Soudani a,1 , Salem Nasri a,1 , Kholdoun Torki b,), Mohamed Abid a,1 , Rached Tourki. 31 July 2001.

RFCs:
RFC2080 RIPng for IPv6
RFC 871 TCP/IP
RFC791 IP 1
RFC 894 IP 2
RFC1010 IOS
RFC1083 RIP
RFC793 TCP
RFC1060 OSPF
RFC2896 IGRP Cisco routing protocols
RFC 1700 Std2 Aug 02 Cisco EIGRP
Web sites:
[W1] www.faqs.com
[W3] www.boson.com
[W4] java.sun.com/j2se/1.3/docs/api/
[W5] java.sun.com
[W6] citeseer.nj.nec.com/es