CHAPTER 3

METHODOLOGY

3.1 Introduction

During the development of the project, the uses of methodologies are important to make sure the flows of the process are smooth and completed. **Figure 3.1** shows the methodologies phases for this project.

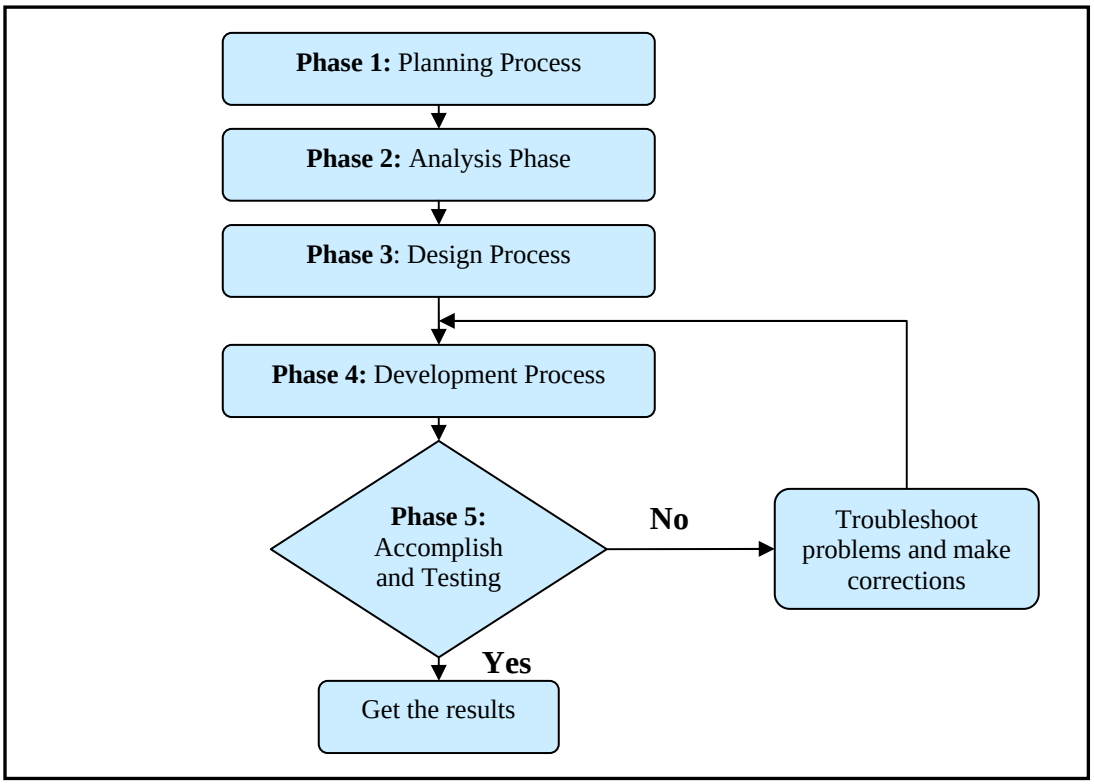

Figure 3.1 The Prototype Model

3.2 Phases in Methodology and the Process

The methodology used for this project is a prototype model. There are five phases that involve to complete the development of this project, which are:

I. Planning phase
II. Analysis phase
III. Design phase
IV. Development phase
V. Accomplish and testing phase

3.2.1 Planning phase

In the beginning of this project planning, I gathered and collected as much information and knowledge about everything that interconnected and relevance. So I can analyze for the problem statement of this project.

This phase aimed to decide and equable the objectives, scopes and steps to be taken to accomplish the project. During the planning, I analyze which routing software that might be used for the development of my PC based router, as well as the needed systems and hardware too.

3.2.2 Analysis Phase

This phase included an in-depth study of routers and its routing process. Also an in-depth study of Internet Protocol, IP routing algorithms such as RIP, OSPF and BGP. Besides, collecting and understanding the software that will be use are important in this phase.

All the essential components that will be required by the project need were estimated at this phase and should be configured accordingly and maintained throughout life cycle of the project.

3.2.2.1 Analysis on Software Used

3.2.2.1.1 Quagga

Quagga is a routing software suite, providing implementations of OSPF (version 2 & version 3), RIP (version 1, version 2 & version 3) and BGP (version 4) for Unix platforms, particularly FreeBSD, Linux, Solaris and NetBSD. Quagga is a fork of the GNU Zebra project (inactive since 2003) which was developed by Kunihiro Ishiguro. The Quagga tree aims to build a more involved community around Quagga than the current centralized model of GNU Zebra.

The Quagga architecture consists of a core daemon (zebra) which acts as an abstraction layer to the underlying Unix kernel and presents the Zserv API over a Unix or TCP stream to Quagga clients. It is these Zserv clients, which typically implement a routing protocol and communicate routing updates to the zebra daemon [21]. Existing Zserv clients are: ospfd (implementing OSPF version 2); ripd (implementing RIP version 1 and Version2); ospf6d (implementing OSPFv3 - (IPv6)); ripngd (implementing RIP v3 (IPv6)); bgpd (implementing BGPv4+ (including address family support for multicast and IPv6)).

Additionally, the Quagga architecture has a rich development library to facilitate the implementation of protocol/client daemons, coherent in configuration and administrative behavior.

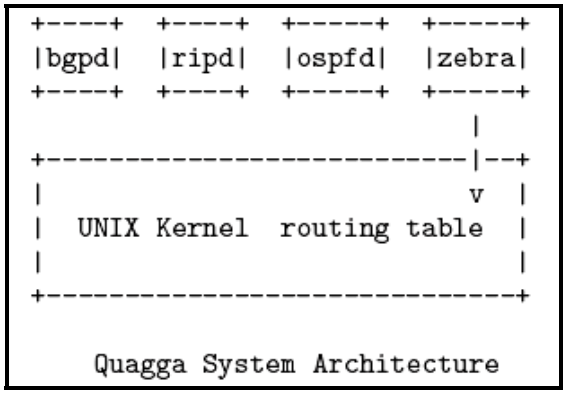

Figure 3.2 Quagga System Architecture [22]

According to **Figure 3.2**, we can explain here the Quagga Routing Architecture consists:

- Modular Design
- One process per protocol

I. BGPD – Border Gateway Protocol (v1, v2, and RIPng)

BGPD is a Border Gateway Protocol 4 (BGP-4) protocol daemon. BGP-4 is described in **RFC1771**. BGPD also supports Multiprotocol Extension for BGP-4 (sometimes known as BGP-4+ or MBGP) which is described in **RFC2283**. BGP-4 is one of the EGPs (Exterior Gateway Protocols) and is used for inter-domain routing.

II. OSPFD – Open Shortest Path First

OSPFD provides an OSPF version 2 routing protocol as described in **RFC2178**. OSPF is one of **IGPs** (Interior Gateway Protocols). Compared with RIP, OSPF can serve much more networks and period of convergence is very short. OSPF is widely used in large networks such as **ISP** backbone and enterprise network.

III. RIPD – Routing Information Protocol

RIP - Routing Information Protocol is widely deployed interior gateway protocol. RIP is a *distance-vector* protocol and based on the *Bellman-Ford* algorithms. As a distance-vector protocol, RIP router send updates to its neighbors periodically, thus allowing the convergence to a known topology. In each update, the distance to any given network will broadcast to its neighboring router. RIPD supports RIP version 2 as described in **RFC2453** and RIP version 1 as described in **RFC1058**.

- One main controlling process

IV. Zebra

Zebra is a routing software package that provides TCP/IP based routing services with routing protocols support such as **RIPv1**, **RIPv2**, **RIPng**, **OSPFv2**, **OSPFv3**, **BGP-4**, and **BGP-4+**. Zebra also supports special **BGP** behavior Route Reflector and Route Server. Adding to traditional **IPv4** routing protocols, Zebra also supports **IPv6** routing protocols. With SNMP daemon, which supports SMUX protocol, Zebra provides routing protocol MIBs.

- Extensible
- IS-IS – Intermediate system to system (for UNIX platforms)

Quagga is not a 'router' like e.g. Cisco or Juniper. Although syntax and general look is similar to Cisco, Quagga will not let us to configure different router aspects, like filters or system daemons. Some of the Cisco configuration directives will not work either.

3.2.2.1.2 Windows Netmeeting

NetMeeting is software that allows one PC to broadcast what is on its screen to other PCs. This software comes already installed on Windows 2000 and XP operating systems. This software will enable us to have free voice (and video) conversations with other computers via the internet and LAN network [23]. The conversations will be held with other people who also have the NetMeeting program installed on their computers. The advantages using this program are:

• There's no need to log in to a central server - i.e., we can connect directly to another computer on the internet without needing to 'sign-in' to anything.

• We can use a common 'whiteboard' to draw diagrams, etc in real time. All parties in the meeting will be able to see and contribute to the whiteboard drawing.

• NetMeeting supports 'remote desktop sharing' in which we can view the desktop of the person to whom you're talking (useful in assisting people with their computer from a remote location)

• NetMeeting works very well on low bandwidth connections (such as dial-up) even with video enabled.

Figure 3.3 About Netmeeting

Figure 3.4 Making a Call

3.2.2.1.3 LanFLOW

LanFlow will help to create and maintain LAN, WAN, Network, Phone system diagrams or any similar type of diagram consisting of devices and their connections. The devices are generally referred to within this product as objects and their representations as figures. They are typically computers, workstations, servers, routers, hubs, and so on. The connectors can be cables, wires, power, or simply representations of logical connections such as workgroup links or communication paths [24].

3.2.2.1.4 Advanced LAN Scanner

Advanced Lan Scanner is a small, easy-to-use, highly configurable network scanner for Win32. This program scans very fast. Advanced Lan Scanner uses multithreading technique, which gives it ability to scan more than 1000 elements per second. If used to scan ports, Advanced Lan Scanner can scan all 65536 ports in less that minute [25]. Also, fast scan is not only Advanced Lan Scanner good feature. It performs very deep scan upon each computer you wish, extracting users, services, shares and a lot of over useful information. It can connect to target machine using default user rights, or we can specify login and password to use. It also has a powerful export options with script language to describe our own save format.

3.2.2.2 Analysis on Hardware and System Used

The operating systems used for this project development are both Windows OS and Linux OS. The Windows OS is for the client PCs and Linux OS is for the router PC.

The following are needed hardwares used for this project according to the design used and shown in the **Table 3.1** below;

Design Type	The Hardware Needed	The Purpose	Bil.
A	Desktop PC	Act as the router PC, and client PCs	3
	NIC	To communicate over the computer network	4
	Network cross-cable	Connects PC to PC	2

B	Desktop PC	Act as the router PC, and client PCs	7
	NIC	To communicate over the computer network	8
	Switch	Connect the Ethernet	2
	Network straight-cable	Connects PCs to switches	8

Table 3.1 The Needed Hardwares

3.2.3 Design Phase

This phase is about the design of the router. To design the project that will be develop, I used the LanFLOW software. The software makes me design the LAN network much easier. There are two designs that I have to complete for the development of this project. **Figure 3.5** shows the Design A.

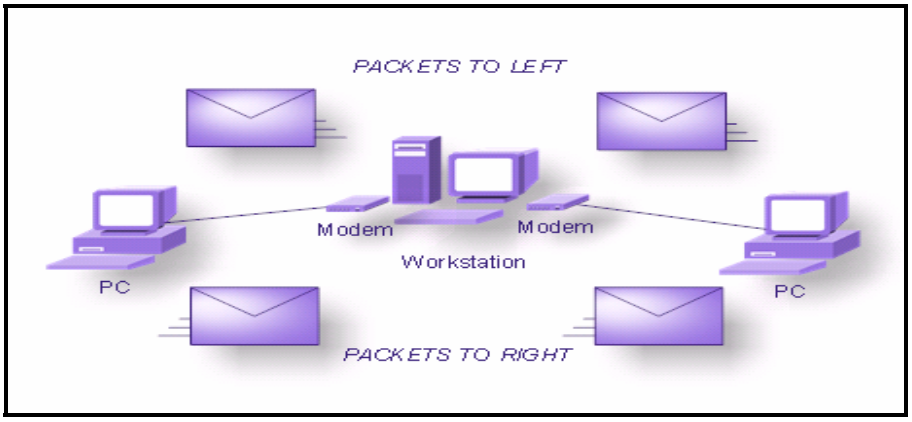

Figure 3.5 Design A

Design A included the development of a PC based router, and connected to two PCs. Design A is a simple one, and before continue to Design B, I must complete this stage first and make sure that the PCs can PING each other as the result. The design is success when the packets sent and received have 0% loss.

Design B in **Figure 3.6** shows the PC as a router, connected to two switches and six PCs, PC A – PC F. Design B is to prove that the router that have been develop is correct and doing its routing process in the right and expected way.

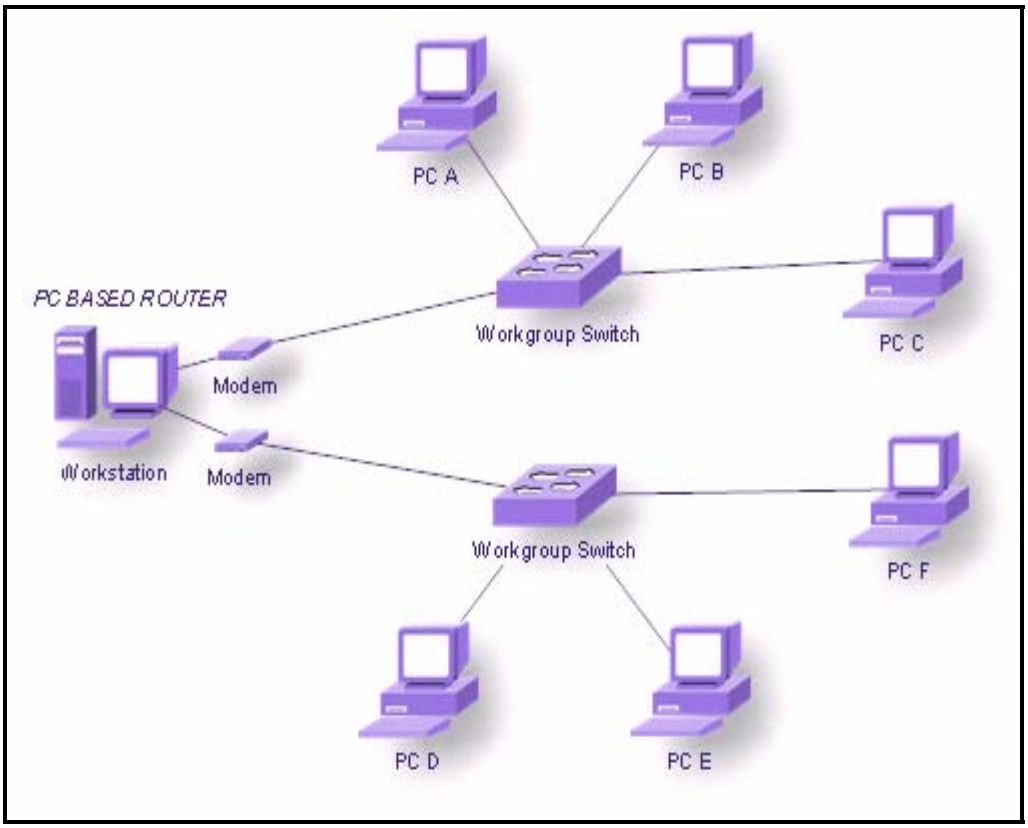

Figure 3.6 Design B

3.2.4 Development Phase

3.2.4.1 Installing NIC

The first stage of this phase is to install the new network interface card, NIC into the desktop computer that will be used to develop the router. To install a standard NIC, I have to take my computer apart a little. I unplug it first and then disconnect all the cables that are attached to the ports at the back of it. Then, move the chassis (the box that holds all of the PC's gooey innards) to a worktable and remove the exterior case. Next, remove the metal back plate (see **Figure 3.7**) at the end of the bus slot (look at the back of the computer).

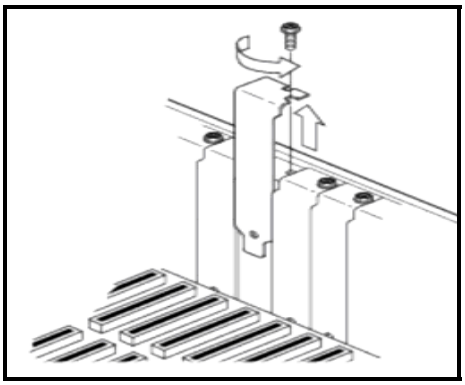

Figure 3.7 Remove the piece of metal that covers the slot [26]

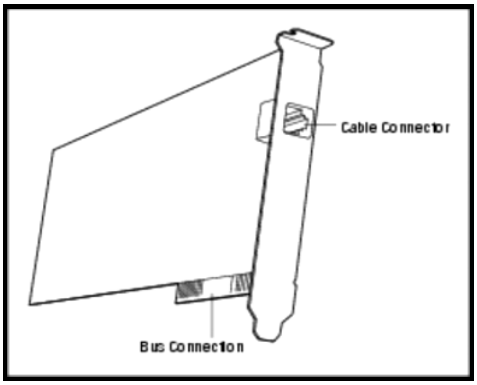

Figure 3.8 The NIC is shaped to match the bus and slot [27]

Now I follow the steps to insert the NIC in the bus. (It is easy and will not be confused about which way it fits into the bus because the back edge of the NIC replaces the metal plate that I removed from the back of the computer, as shown in **Figure 3.8**.)

When I restart the computer, Windows notices that I have installed new hardware and displays a message offering to complete the software side of the installation of the network adapter. I am using Windows XP, so I just see a message telling that the software has been installed. Besides, when I using the Linux OS, the NIC will be detected automatically too.

3.2.4.2 Make Own Network Cables

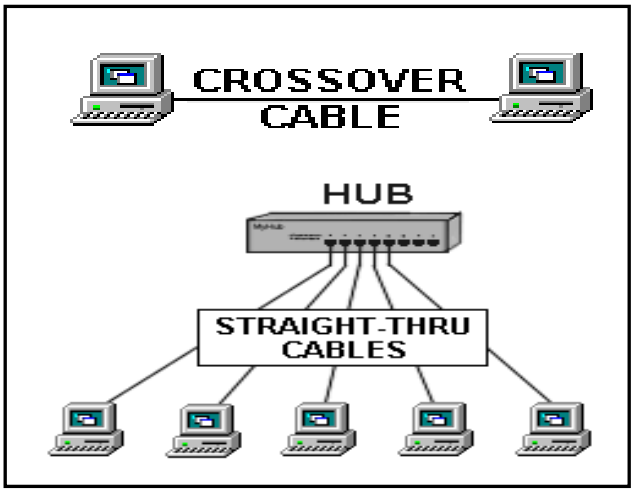

Figure 3.9 LAN Cables Connection [28]

Refer to **Figure 3.9**, a LAN can be as simple as two computers, each having a network interface card (NIC) or network adapter and running network software, connected together with a **crossover cable**. The next set up would be a network consisting of three or more computers and a hub. Each of the computers is plugged into the hub with a **straight-thru cable** (the crossover function is performed by the hub) [29].

For first design of this project, I used to connect PC to PC, so the cable that will be used is the crossover cable. To make my own crossover network cable, I refer to the Color-Code Standards.

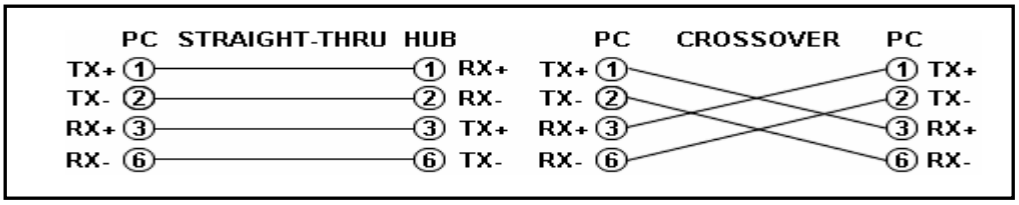

Figure 3.10 Simple pin-out diagrams of the two types of UTP Ethernet cables [30]

The TX (transmitter) pins are connected to corresponding RX (receiver) pins, plus to plus and minus to minus. I must use a crossover cable to connect units with identical interfaces. Two wires color-code standards apply; EIA/TIA 568A and EIA/TIA 568B.

The codes are commonly depicted with RJ-45 jacks as follows (the view is from the front of the jacks):

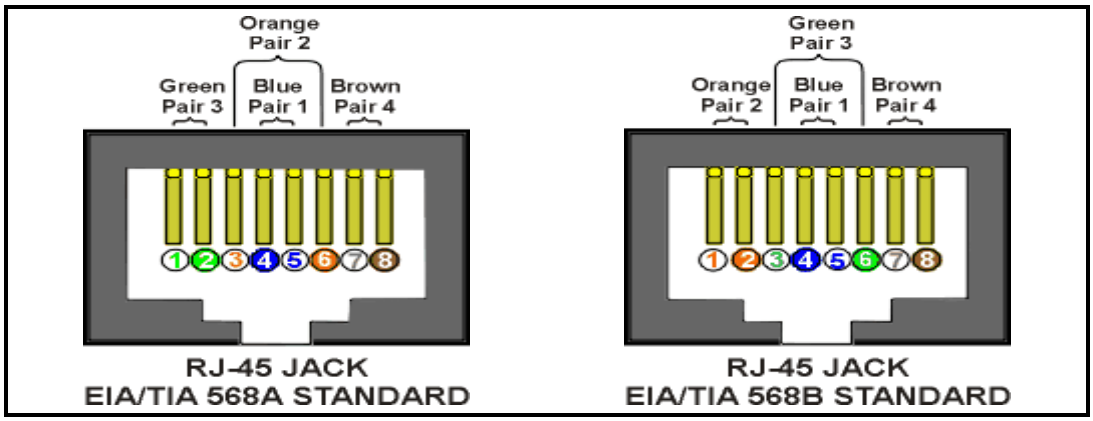

Figure 3.11 The Color-Code Standards [31]

If we apply the 568A color code and show all eight wires, our pin-out looks like this:

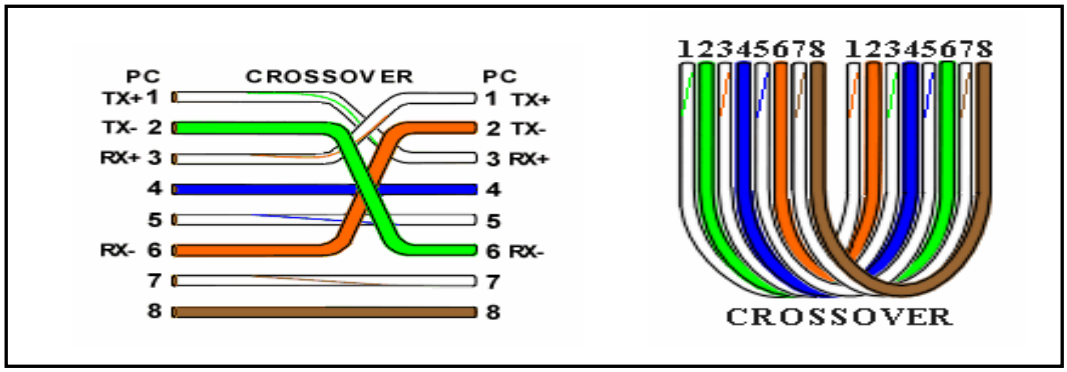

Figure 3.12 The Crossover pins, TX and RX [32]

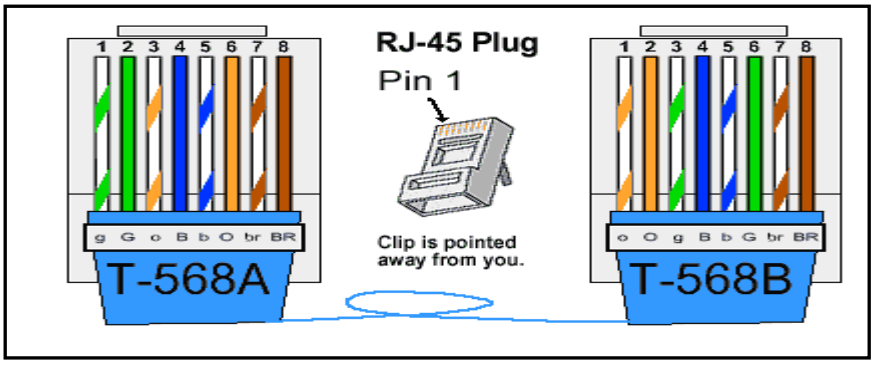

Figure 3.13 RJ-45 Crossover Ethernet Cable [33]

A good way of remembering how to wire a Crossover Ethernet cable is to wire one end using the T-568A standard and the other end using the T-568B standard. Another way of remembering the color-coding is to simply switch the Green set of wires in place with the Orange set of wires. Specifically, switch the solid Green (G) with the solid Orange, and switch the green/white with the orange/white[34].

3.2.4.3 Installing Linux OS

The next step is to install the Linux OS into the PC that will act as the router for this project. The Linux OS that I chose to install is SUSE. S.u.S.E is an acronym for the German phrase "**S**oftware- **u**nd **S**ystem-**E**ntwicklung" ("Software and system development") [35]. SUSE includes an installation and administration program called YaST2 which handles hard disk partitioning, system setup, RPM package management, online updates, network and firewall configuration, user administration and more in an integrated interface. This is the reason I chose to use the SUSE Linux for the development.

3.2.4.4 Installing Quagga Software

There are three steps for installing the software; *configuration, compilation, and installation*. The step to get the Quagga running is to issue the following commands:

% configure

% make

% make install

However, by using SUSE Linux, and get the Quagga software in .rpm file, I just easily used YaST; and chose to download new software. Before continue to the next step, I have to make sure that the folder for Quagga is there in '/etc', as shown below in **Figure 3.14.**

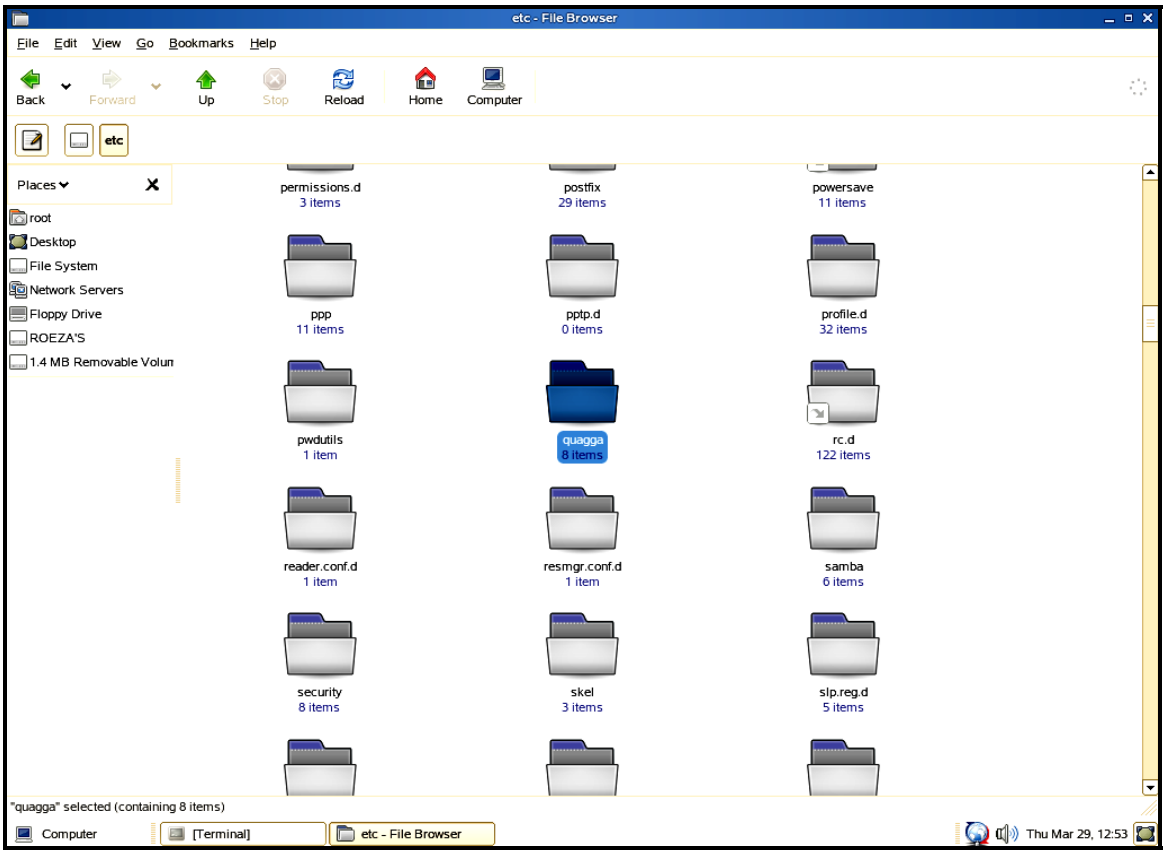

Figure 3.14 Browsing the files for Quagga

Quagga daemons have their own terminal interface or VTY. After the installation, I setup beasts's port number to connect to them. I add the following entries to '/etc/services', shown in **Figure 3.15** ;

- zebrasrv 2600/tcp # zebra service
- zebra 2601/tcp # zebra vty
- ripd 2602/tcp # RIPd vty
- ripngd 2603/tcp # RIPngd vty
- ospfd 2604/tcp #OSPFd vty
- bgpd 2605/tcp #BGPd vty
- ospf6d 2606/tcp #OSPF6d vty
- ospfapi 2607/tcp # ospfapi
- isisd 2608/tcp #ISISd vty

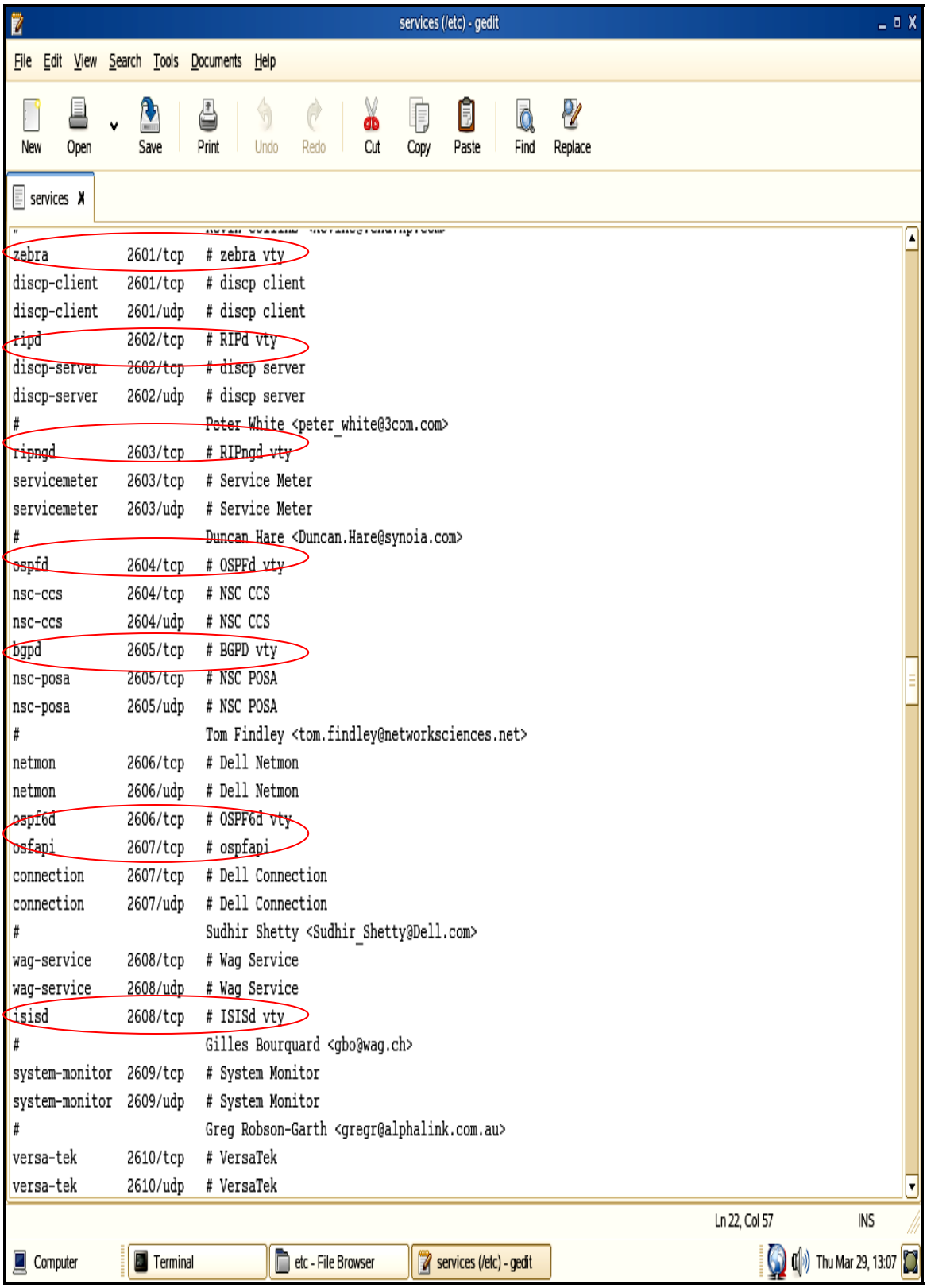

Figure 3.15 The daemons added

3.2.4.5 Write the configuration files

There are five routing daemons in use, and there is one manager daemon. These daemons may be located on separate machines from the manager daemon. Each of these daemons will listen on a particular port for incoming VTY connections. The routing daemons are:

- ripd, ripngd, ospfd, ospf6d, bgpd
- zebra

The following are discuss about the commands and configuration files for all the routing daemons. **Figure 3.16** shows the configuration files in the router PC. The configuration commands for all daemons included in this report in **APPENDICES.**

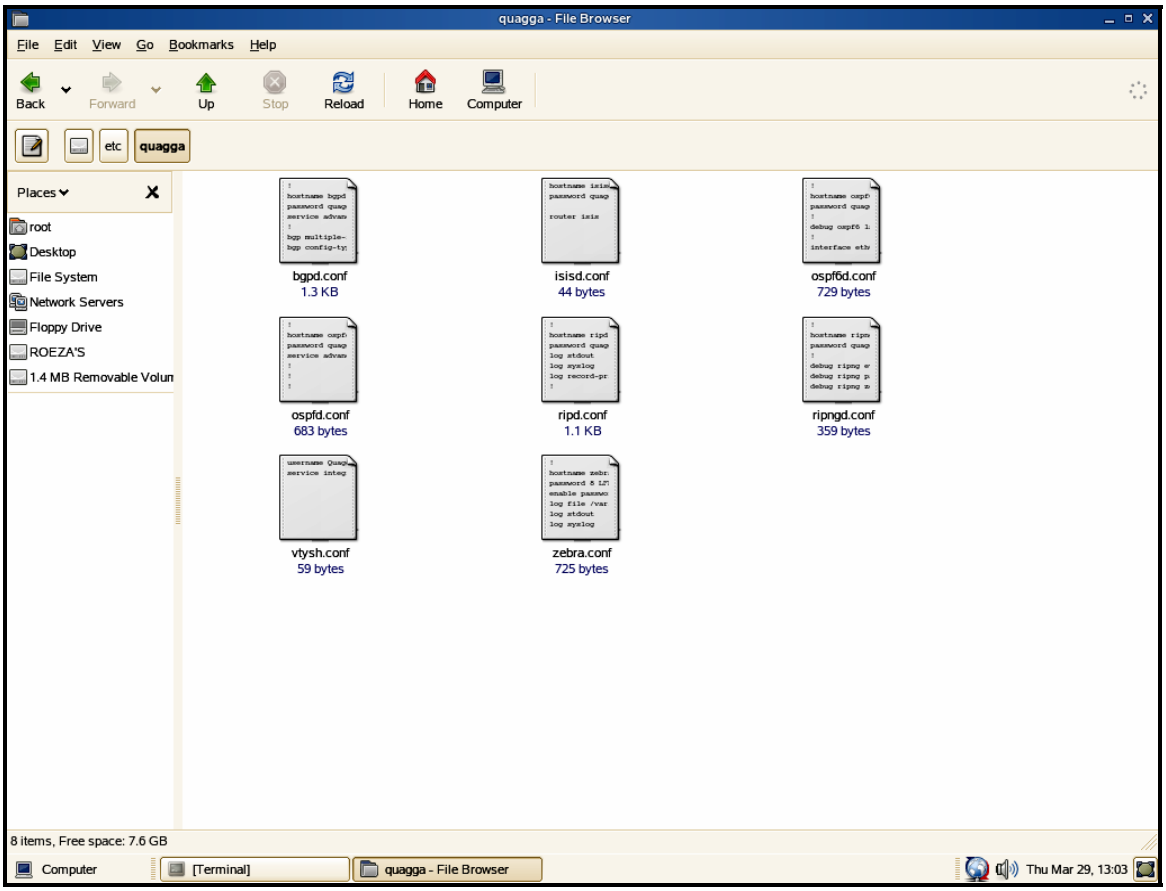

Figure 3.16 The Configuration Files

In a config file, I write the debugging options, a vty's password, routing daemon configurations, a log file name, and so forth. The config files are generally found in:

- `/etc/quagga/*.conf'

As said above, each of the daemons has its own config file. The daemon name plus `.conf' is the default config file name. For example, zebra's default config file name is:

- `/etc/quagga/zebra.conf'

The basic conf. commands used for the development of this router project are:

COMMANDS	**FUNCTIONS**
hostname *hostname*	Set hostname of the router.
password *password*	Set password for vty interface. If there is no password, a vty won't accept connections.
enable password *password*	Set enable password.
log trap *level* **no log trap**	The log trap command sets the current logging level for all enabled logging destinations, and it sets the default for all future logging commands that do not specify a level. The normal default logging level is debugging. The *no* form of the command resets the default level for future logging commands to debugging, but it does not change the logging level of existing logging destinations.
log stdout **log stdout *level*** **no log stdout**	Enable logging output to stdout. If the optional second argument specifying the logging level is not present, the default logging level will be used. The *no* form of the command disables logging to stdout. The *level* argument must have one of these values: emergencies, alerts, critical, errors, warnings, notifications, informational, or debugging.

log file *filename* **log file *filename level*** **no log file**	To log into a file, need to specify *filename* as in this example: *log file /var/log/quagga/bgpd.log informational* If the optional second argument specifying the logging level is not present, the default logging level will be used. The *no* form of the command disables logging to a file.
log syslog **log syslog *level*** **no log syslog**	Enable logging output to syslog. If the optional second argument specifying the logging level is not present, the default logging level will be used. The *no* form of the command disables logging to syslog.
log monitor **log monitor *level*** **no log monitor**	Enable logging output to vty terminals that have enabled logging using the *terminal monitor* command. By default, monitor logging is enabled at the debugging level, but this command (or the deprecated *log trap* command) can be used to change the monitor logging level. If the optional second argument specifying the logging level is not present, the default logging level (typically debugging, but can be changed using the deprecated *log trap* command) will be used. The *no* form of the command disables logging to terminal monitors.
log facility *facility* **no log facility**	This command changes the facility used in syslog messages. The default facility is *daemon*. The *no* form of the command resets the facility to the default *daemon* facility.
log record-priority **no log record-priority**	To include the severity in all messages logged to a file, to stdout, or to a terminal monitor (i.e. anything except syslog), use the *log record-priority* global

	configuration command. To disable this option, use the *no* form of the command. By default, the severity level is not included in logged messages.
service password-encryption	Encrypt password.
service advanced-vty	Enable advanced mode VTY.
service terminal-length ***<0-512>***	Set system wide line configuration. This configuration command applies to all VTY interfaces.
line vty	Enter vty configuration mode.
banner motd default	Set default motd string.
no banner motd	No motd banner string will be printed.
exec-timeout ***minute*** **exec-timeout** ***minute second***	Set VTY connection timeout value. When only one argument is specified it is used for timeout value in minutes. Optional second argument is used for timeout value in seconds. Default timeout value is 10 minutes. When timeout value is zero, it means no timeout.
no exec-timeout	Do not perform timeout at all. This command is as same as *exec-timeout 0 0.*
access-class ***access-list***	Restrict vty connections with an access list.

Table 3.2 The basic configuration commands

3.2.5 Accomplish and Testing Phase

The next phase is to accomplish and testing the design of the project. The LAN network address of both NIC attached in the router must be sets statically and it shown in **Table 3.3**;

Network device: **eth0**	Network device: **eth1**
Hardware address: 00:19:e0:0c:10:3e	Hardware address: 00:0a:e6:8f:1f:9e
IP address: 192.168.0.10	IP address: 10.0.0.2
Netmask: 255.255.255.0	Netmask: 255.255.255.0
Broadcast: 192.168.0.255	Broadcast: 10.0.0.255

Table 3.3 The LAN network addresses for eth0 and eth1

For the Design A, the network cables used are cross-cables. The LAN IP address of both PCs connected to the router PC is sets statically as shown in **Table 3.4**;

PC A	**PC B**
Ethernet adapter Local Area Connection:	Ethernet adapter Local Area Connection:
Connection-specific DNS Suffix . :	Connection-specific DNS Suffix . :
IP Address. . . . . : 192.168.0.20	IP Address. . . . . : 10.0.0.5
Subnet Mask . . . . . : 255.255.255.0	Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.0.10	Default Gateway . . . . . : 10.0.0.2

Table 3.4 The Connected PCs LAN IP network addresses for Design A

The next is for Design B. The network cables used are straight-cables connected to all PCs. **Table 3.5** shows the LAN IP network addresses for all PCs. The testing for the designs included here separate in two ways. For the Design A, I was used the command **Ping** to the IP addresses to from PC to PC. **Ping** is a computer network tool used to test

whether a particular host is reachable across an IP network. However, the **Windows Netmeeting** program was used to test the connection between all PCs for Design B.

Name of PCs	**LAN IP network addresses**
PC A	IP Address. . . . . : 192.168.0.100 Subnet Mask . . . . . : 255.255.255.0 Default Gateway . . . . . : 192.168.0.10
PC B	IP Address. . . . . : 192.168.0.40 Subnet Mask . . . . . : 255.255.255.0 Default Gateway . . . . . : 192.168.0.10
PC C	IP Address. . . . . : 192.168.0.20 Subnet Mask . . . . . : 255.255.255.0 Default Gateway . . . . . : 192.168.0.10
PC D	IP Address. . . . . : 10.0.0.5 Subnet Mask . . . . . : 255.255.255.0 Default Gateway . . . . . : 10.0.0.2
PC E	IP Address. . . . . : 10.0.0.15 Subnet Mask . . . . . : 255.255.255.0 Default Gateway . . . . . : 10.0.0.2
PC F	IP Address. . . . . : 10.0.0.20 Subnet Mask . . . . . : 255.255.255.0 Default Gateway . . . . . : 10.0.0.20

Table 3.5 The Connected PCs LAN IP network addresses for Design B