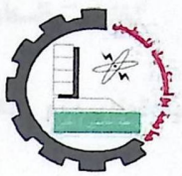
Palestine Polytechnic University



College of Engineering & Technology
Electrical & Computer Engineering Department

Graduation Project

Simulation of a Parallel Router Using a Cluster of PCs

Project Team

Tayseer Sweiti

Ola Yousef Abu Shekha

Project Supervisor
Dr. Mohammed ALdasht

Hebron-Palestine

June, 2005

Abstract

As computer networks become larger and larger routers are increasingly become more important in the network traffic. Larger networks mean larger numbers of users which mean larger number of routing requests which mean a heavy load on the router. Vast amount of requests can convert the router into a bottleneck of the network. A way to overcome this problem is by employing a more potential router with more ports and more processing power, this solution can result expensive and will reach the limit again as the network size increases.

A more efficient and economic way to overcome the bottleneck is by using a parallel machine to function as a router. Such parallel machine is a cluster of PCs which is economic to build and scalable. The scalability of clusters of PCs is an important issue because the routing problem resides in the continuous increase of the network nodes. So, this problem can simply treated by increasing the number of PCs in the cluster (size of parallel machine) to increase the performance of networks.

يتقدم قطاع الاتصالات و شبكات الكمبيوتر بشكل متسارع، و نتيجة لهذا التقدم فان عدد شبكات الكمبيوتر يزداد بشكل كبير الامر الذي ادى الى زيادة الضغط على ال (ROUTER) الذي يلعب دوراً اساسياً في ربط هذه الشبكات معاً، هذا الضغط المتزايد يؤدي الى ابطاء سرعة ال (ROUTER) وبالتالي ابطاء سرعة الشبكات بشكل عام ، احد الحلول المقترحة لهذه المشكلة هو استخدام (ROUTER) بقدرات أعلى، لكن هذا الحل يكلف كثيراً، وسرعان ما تعود المشكلة لتظهر مع زيادة عدد الشبكات.

احد الحلول الذي قد يكون مناسباً اكثر هو استخدام عدة اجهزة كمبيوتر لتقوم معاً بأداء وظيفة ال (ROUTER)، قمنا في هذا المشروع بعمل محاكاة لنظام تقوم فيه عدة اجهزة كمبيوتر بوظيفة ال (ROUTER) ، من فوائد هذا النظام انه قابل لزيادة ادائه بزيادة عدد اجهزة الكمبيوتر التي تقوم بوظيفة ال (ROUTER).

List of Contents

List of Tables

Table (2) List of Tables

List of Figures

Table (2) List of Figures

List of Figures

Table (2) List of Figures

Chapter One

Introduction

Chapter One

Introduction

1.1 Overview

As computer networks become larger and larger routers are increasingly become more important in networks. The function of routers in computer networking can be summarized in four points, which are: prioritize traffic, translate network addresses, tunnel and act as firewalls. So, routers could be a primary determinant of the overall network performance. The primary problem that can limit the performance of a router is the congestion that could be result from an aggressive demand on the router, converting the router into a bottleneck of networks.

A way to overcome this problem is by employing a more potential router with more ports and more processing power, this solution can result expensive and will reach the limit again as the network size increases.

A more efficient and economic way to overcome the bottleneck is by using a parallel machine to function as a router. Such parallel machine is a cluster of PCs which is economic to build and scalable. The scalability of clusters of PCs is an important issue because the routing problem resides in the continuous increase of network's nodes, so this problem can be simply solved by increasing the number of PCs in clusters (size of parallel machine) to increase the performance of networks.

1.2 Literature Review

Many parallel router architectures have been proposed to solve the problem of the bottleneck. Among the most popular are the Massively Parallel Router (MPR) of Pluris Inc. and the GRF IP Switch of Ascend Inc. A MPR consists of a large number of processing nodes. Each processing node is a single-board computer. These processing nodes are interconnected by a Switch. Each processing node has its own routing table. They share the routing load of the router [5].

A GRF IP Switch is composed of multiple IP Forwarding Media Cards interconnected by a switch. Each media card has its own processor, network interfaces and routing table. All media cards perform IP routing independently [5].

Despite the differences, the architectures of these parallel routers are very similar. A high-speed switch interconnects multiple Routing Nodes. Each routing node has its own processor, network interfaces and memory. It provides one or more ports of the router. All the routing nodes route packets independently. If a packet must go through the port of another routing node, that routing node will not route the packet, it simply forwards the packet through the link layer. These routers are scalable. The scalability comes from the ease to add more processing power and internal bandwidth to the router. Doubling the number of RNs and switches will double the throughput of the router [5].

The difference between these projects and our project is that, our Project depends on the use of a cluster of personal computer connected by a switch to do the routing processes, and depend on the LAM/MPI on implementing the parallel routing, but the previous projects uses multiple processing nodes.

1.3 Estimated Cost

The cost of this project varies depending on the available resources; because the project doesn't require new computing resources and can be build using unused computer recourses. But in our calculations we assume that nothing is available, and we have to buy every thing. We require the following:

- **Hardware Requirements**:
  - (5) Pentium II, III, or 4 PCs with average cost for each one 350$
  - (1)Switch with 8 ports which cost 20$
  - (10) 10/100 PCI NICs each one will cost 10$
  - Cables and Rj-45 connectors which will cost about 50$

  Total hardware resources cost is 1870$
- **Software requirements**
  - (4) Copies of Linux operating system each one will cost 30$

  Total software requirements cost is 120$
- Human requirements
  - We require two programmers to work 35 days, with 8 hours a day. Each programmer require 10$ per hour.

  Total human requirements cost is 5600$

Total project cost is **7590$**

1.4 Time Schedule

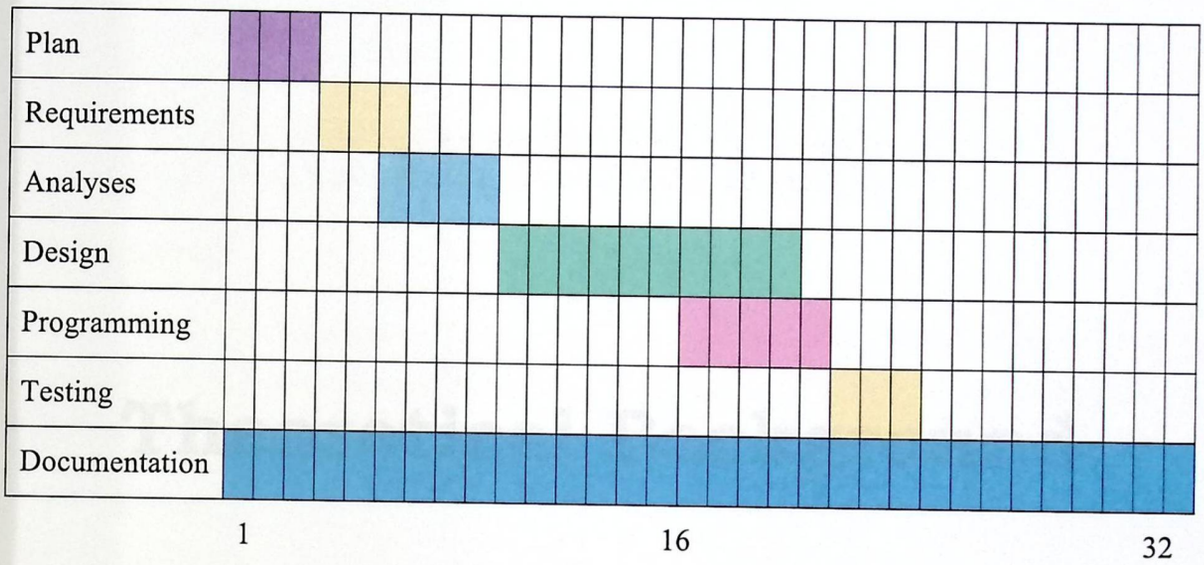


Figure (1.1) Time Schedule (Weeks)

Chapter Two

Theoretical Background

Chapter Two

Theoretical Background

In this chapter we introduce routing and parallelism as two main topics in this chapter. In addition we will show how these topics are related to each other.

2.1 Routing

Routing is the process of forwarding packet to the proper path depending on the logical address. Routing is done by devices dedicated for routing or by using off-the-shelf computers configured to perform routing.

2.1.1 Router

A router is basically a dedicated microcomputer that has a Central Processing Unit (CPU), memory, input/output interfaces including management consol interface, and operating system. Routers can have multiple ports and can connect LANs and WANs running at different speeds and different protocols. Routers operate at the network layer (layer 3) of the OSI model. The primary functions of the router in internetworking are best path determination between the source and destination depending on the logical address of the incoming packet, and switching or forwarding the packet to the next hop in the path.

2.1.1.1 Router Architecture

- CPU: The Central Processing Unit (CPU) executes instructions in the operating system. Among these functions are system initialization, routing functions, and network interface control. The CPU is a microprocessor. Large routers may have multiple CPUs [3].

- RAM/DRAM: stores routing tables, ARP cache, fast-switching cache, packet buffering (shared RAM), and packet hold queues; RAM also provides temporary and/or running memory for a router's configuration file while the router is powered; RAM content is lost during a power down or restart

- NVRAM: non-volatile RAM stores the router's backup/startup configuration file; NVRAM content is retained during power down or restart

- Flash: erasable, reprogrammable ROM that holds the operating system image and microcode; Flash memory enables software updates without removing and replacing processor chips; Flash content is retained during power down or restart; Flash memory can store multiple versions of IOS software

- ROM: contains power-on diagnostics, a bootstrap program, and operating system software; software upgrades in ROM require removing and replacing pluggable chips on the CPU

- Interfaces: network connections on the motherboard or on separate interface modules, through which packets enter and exit a router

- Consol: used to configure the router by connecting the router to another work station.

- Power Supply: The power supply provides the necessary power to operate the internal components. Larger routers may use multiple or modular power supplies. In some of the smaller routers the power supply may be external to the router

Figure (2.1) shows a router internal components.

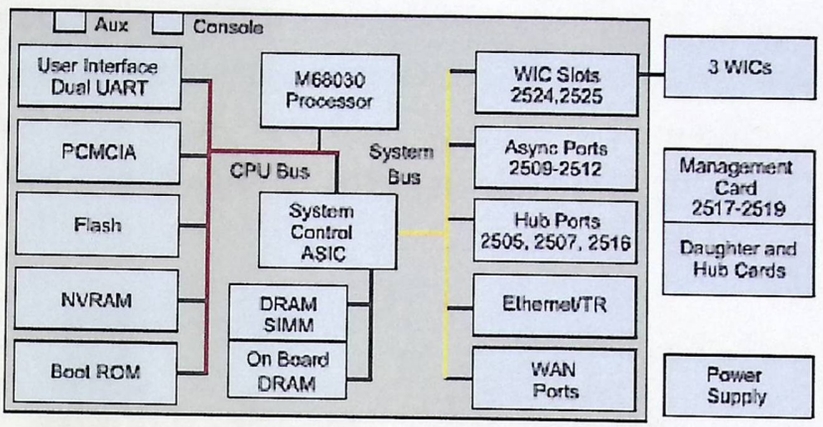


Figure (2.1) Router Internal Components [3]

2.1.1.2 Router Technologies

The Internet is composed of a large number of autonomous systems, each of which has routers that typically play one of three roles:

- Interior routers: internal to one autonomous LAN.
- Border routers: connect autonomous LAN with a WAN.
- Exterior router: direct data between nodes outside an autonomous LAN.

2.1.1.3 Router Function

Router performs two primary functions in forwarding the packet from the source to destination, path determination and packet switching.

2.1.1.3.1 Path Determination

Routers determine the best path between the source and the destination depending on metrics number which calculated by the routing algorithms depending on special characteristics of the routing path, these characteristics may typically include delay, bandwidth, load, reliability, hop count, ticks and cost, as shown in figure (2.2).

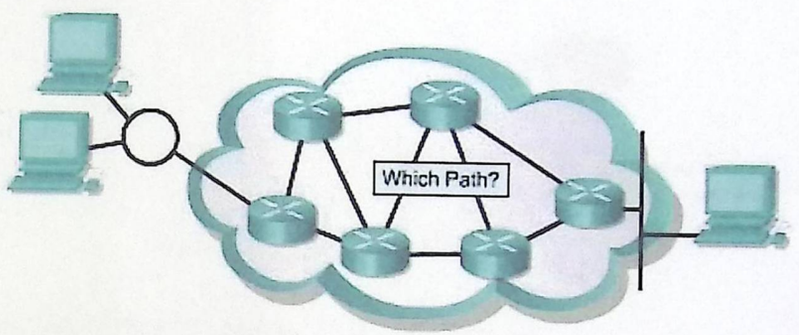


Figure (2.2) Path Determination [3]

Typically, the smaller the metric number, the better the path. To assets the best path determination routing algorithms maintain routing tables in routers .These tables contain routing information such as the network destination address and its outgoing interface.

2.1.1.3.2 Switching

When a packet reaches a router the network layer in the router determine the outing interface by matching the destinations network logical address with the outing interface. The best bath is determined using the routing table and metrics number. The destinations MAC address changed to the next hop or the final destinations MAC address. As packet travel through the routing path from one node to another the MAC address change, but the logical address stay the same, as shown in figure (2.3).

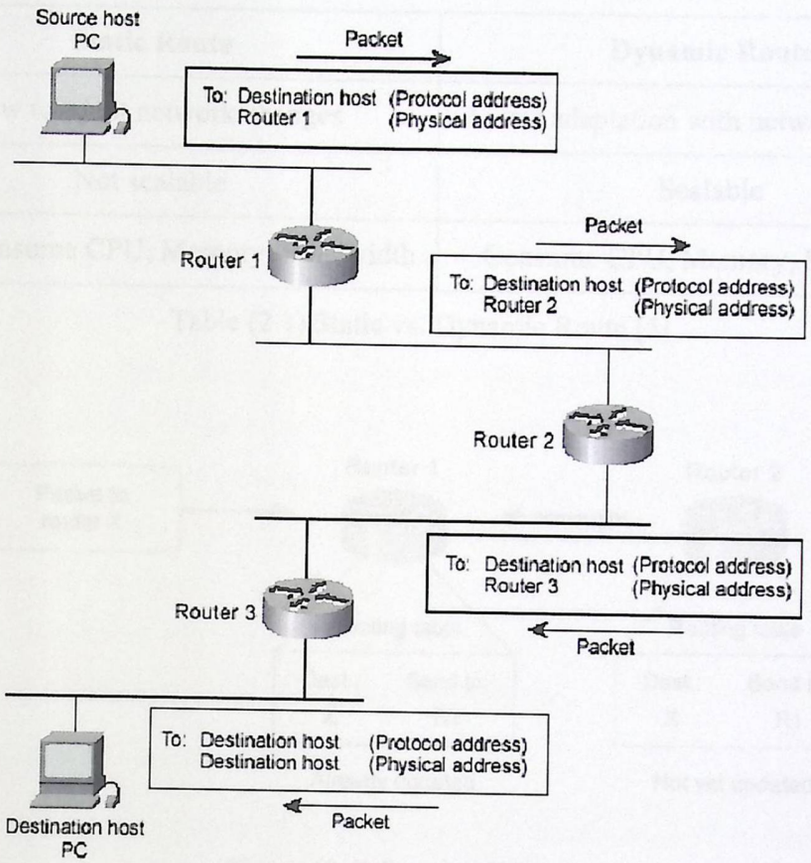


Figure (2.3) Packet Switching [4]

2.1.1.4 Routing Table

Routing table contain information necessary to the router for best path determination. There are two methods to maintain routing table; static and dynamic. In static method an administrator manually configures forwarding table entries. In dynamic method routers exchange network reachability information using routing protocols. The following is a comparison between the two methods:

Table (2.1) shows a compression between static and dynamic routing. Figure (2.5) shows an example of routing table.

Static Route	Dynamic Route
Slow to adapt network changes	Faster adaptation with network changes
Not scalable	Scalable
Don't Consume CPU, Memory, Bandwidth	Consume CPU, Memory, Bandwidth

Table (2.1) Static vs. Dynamic Route [5]

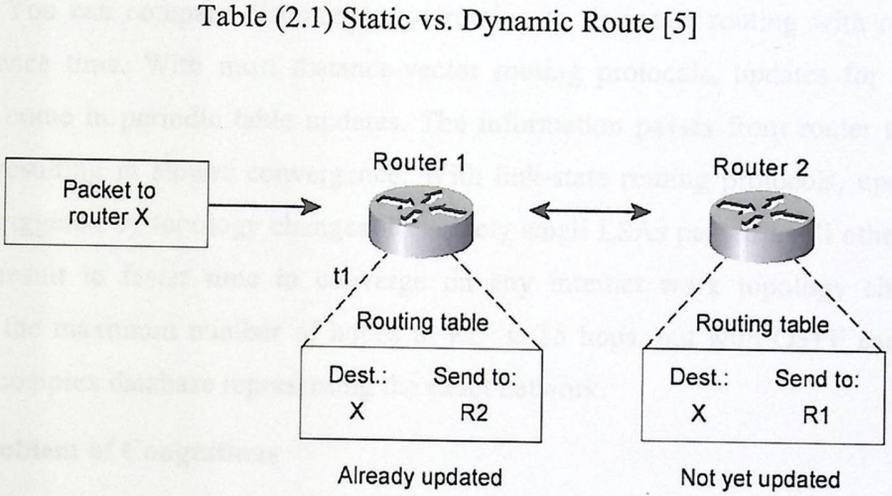


Figure (2.4) Routing Table [4]

2.1.1.5 Protocols

Protocols are rules that govern the way in which the packets are sent from a source to a destination. In routing there are two types of protocols, routed and routing protocols.

2.1.1.5.1 Routed Protocols

Routed protocols contain layer-3 addresses. These protocols used between routers to direct user traffic. It includes information to direct the packet from one node to another. In addition, it specifies the packet fields. IP, IPX, AppleTalk, and DEC net are examples of routed protocol.

2.1.1.5.2 Routing Protocols

Routing protocols are used between routers to maintain routing tables by allowing message passing between routers to share routing table updates. So it assists routed protocols to do their Job. There are two classes of routing protocols; distance vector routing protocols such as RIP, and link state routing protocols such as OSPF.

You can compare distance-vector routing to link-state routing with respect to convergence time. With most distance-vector routing protocols, updates for topology changes come in periodic table updates. The information passes from router to router, usually resulting in slower convergence. With link-state routing protocols, updates are usually triggered by topology changes. Relatively small LSAs passed to all other routers usually result in faster time to converge on any internet work topology change. In addition the maximum number of hopes in RIP is 15 hops, but with OSPF each router builds a complex database representing the exact network.

2.1.2 Problem of Congestions

In a large network, when a large number of requests reach the router at the same time. The router will not be able to handle these requests at the same time, so packets will spend more time waiting for the router to be ready; this will slow down the network and makes the router a bottleneck in the network.

One of the solutions to this problem is to use a cluster of computers connected in parallel and performs the routing process. This method will make the routing process faster by dividing the load on a number of routing nodes. This method will solve the problem of the bottleneck and increase the performance and the speed of networks.

2.2 Parallelism

Parallel computing is the simultaneous use of multiple compute resources to solve a computational problem. The computing resources could include a single computer with multiple processors, an arbitrary number of computers connected by a network, or a combination of both [1].

Parallelism could save money, because it can use available computer resources. It is faster than serial computation, because it divides the load on more than one computer resource. It solves the problem of memory constrains. We can extend the memory by adding new computer resources.

2.2.1 Parallel Memory Architecture

Memory architecture is concerned of the layout of the memory in a parallel computing environment. There are three types of memory architecture:

- Shared memory architecture: in this method all the processors share the same memory space. The disadvantage of this model is that, this method is not scalable and cost more money to build.
- Distributed memory architecture in this method a number of processing nodes are connected by a fast network. This method is scalable and cost effective.
- Hybrid Distributed-Shared Memory: in this method contains the two previous methods. And contain the advantages and disadvantages of both methods.

Figure (2.5) shows distributed memory architecture.

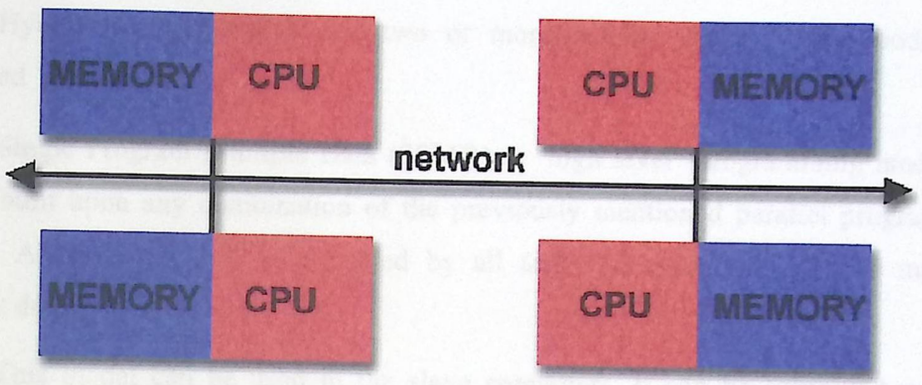


Figure (2.5) Distributed Memory Architecture [1]

2.2.2 Parallel Programming Models

There are several parallel programming models that can be used. These models is independent of the under laying hardware. The following models are the models that are commonly used:

- Shared Memory: in this model tasks share a common address space. One of the disadvantages of this model is that it is difficult to understand and manage data locality.
- Threads: in this model a single process can have multiple, concurrent execution paths.
- Message Passing: in this model tasks exchange data through communications by sending and receiving messages. A set of tasks use their own local memory during computation. Multiple tasks can reside on the same physical machine as well across an arbitrary number of machines. Data transfer usually requires cooperative operations to be performed by each process. For example, a send operation must have a matching receive operation. The most used implementation of this model is Message Passing Interface(MPI)
- Data parallel: A set of tasks work collectively on the same data structure, however, each task works on a different partition of the same data structure

- Hybrid: In this model, any two or more parallel programming models are combined

- Single Program Multiple Data (SPMD): a "high level" programming model that can be built upon any combination of the previously mentioned parallel programming models. A single program is executed by all tasks simultaneously. Tasks may use different data.

This model can be used in the slave computers. It can be combined with the message passing model. And can be implemented using massage passing interface (MPI).

- Multiple Program Multiple Data (MPMD): a "high level" programming model that can be built upon any combination of the previously mentioned parallel programming models. Each task can be executing the same or different program, and tasks may use different data

Chapter Three

Design Concepts

Chapter Three

Design Concepts

In this project we will start by making a simulation of parallel router. Implementation of the project will be dependant on the available time. By the end of this project we expect to achieve the following main objectives.

3.1 project Objectives

We are trying to solve the problem of overload on the router by making a simulation of system in which the routing process done by a number of personal computers which work in parallel. So the routing tasks are distributed between various PCs. In order to perform this simulation, we need to:

1. Design a parallel algorithm that can solve routing tasks, it inputs requests from a network and decide the best route for this requests.
2. The parallel algorithm can manage load balancing between different nodes of the parallel machine using some good dynamic load distribution technique.
3. Design and implement requests generator that can produce requests for the router. A random generator of requests should be implemented.
4. The router for which we will make a simulation is more scalable, because increasing the number of processing nodes will increase the routing potential.
5. The router should adapt new topological changes.

3.2 General Block Diagram

The block diagram shows the router which consists of four personal computers connected through a switch. Three local area networks connected to the router through the slave PCs. The packet generator will do the function of network-1, network-2 and network-3 by generating a random request to the router, as shown in figure (3.1).

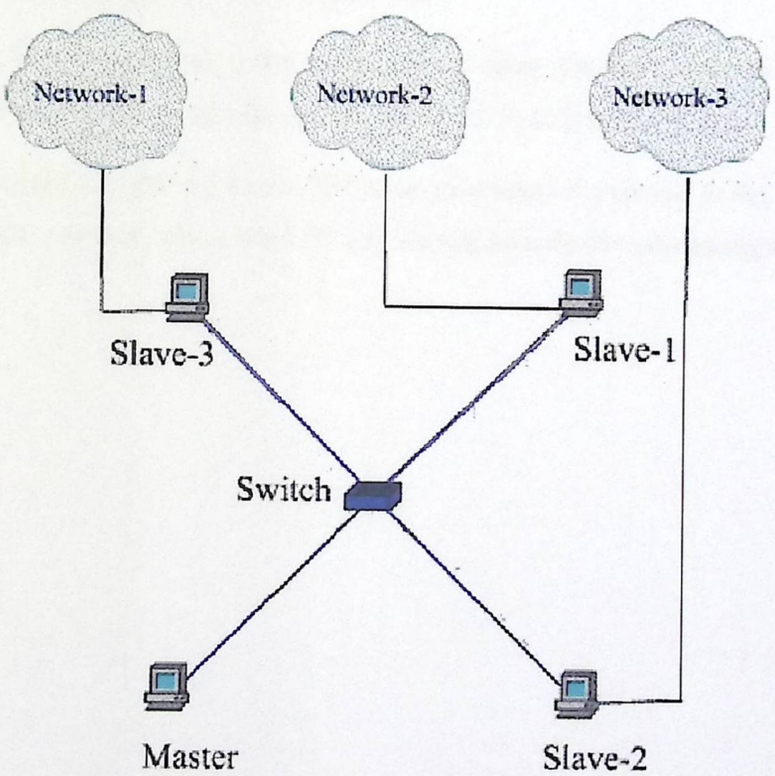


Figure (3.1) General Block Diagram

3.3 How System Works

The system will have the following features:

- The router will connect three LANs. The three LANs which the router will connect will be represented by one PC (packet generator). This pc will generate random requests to the system (Router). Two types of packets will be generated, normal data packets and topological changes.
- Our project is a Parallel System with distributed memory architecture with Single program, Multiple Data (SPMD) model.
- When a slave receives a data packet it will route it to the proper path. If the slave receives a topology changes packet it will forward it to the master.
- The master PC will have one NIC cards to connect the master to the other 3 slaves through a switch. The master PC will be responsible for calculating routing table.

Chapter Four

Hardware System Design

Chapter Four

Hardware System Design

In order to get the best performance, we should use very fast hardware resources. But because it is difficult to get these devices, we will try to use the best available components. The following hardware components will be used:

4.1 Computers

The computers that we will use are normal personal computers (off-the-shelf) such as Pentium II, III and 4. We need 5 personal computers classified to work as following:

- One computer will work as a master.
- Three Computers will work as slaves.
- One computer will work as request generator.

4.2 Network Devices:

4.2.1 Switch

Switch is a networking device that connects different network segments. It uses MAC address information of an incoming frame for making a decision about where to direct the frame. So it is traditionally considered as layer 2 device of OSI model. But new switch can work on layer 3 (routing switch) or layer 4 (application switch). Switch has internal processor, memory, I/O interfaces, and operating system.

Switches are used in segmenting the collision domains which yields to a better performance by decreasing the number of collisions, in addition it enhance security. There are two main methods of switching:

4.2.1.1 Cut-Through Mode

In this mode the switch reads the frame header and decides where to forward the frame before it receives the entire frame. So when it reads the destination field in the header of the incoming frame it forwards it to the proper port without making error checking. The advantage of this method is speed and it is suitable for small workgroups.

4.2.1.2 Store and Forward

In this mode the switch reads the entire frame into its memory and makes the needed error checking before transmitting the frame. It is consume more time than cut through method but it transmit data more accurately

The switch is an important network device in the project. It connects the three slave's personal computers to the master computer to form together a router. Because we should connect five personal computers we use 8 ports 10/100 mbps Ethernet switch which work on Cut-Trough Mod.

4.2.2 Network Interface Cards(NICs)

It is a connectivity device that attached to workstations and other nodes, it enables network nodes to transmit and receive data. NIC considered as layer 1 and layer 2 network device in the OSI model, because it transmit and receive signal using network media and it use physical address(MAC) to make sure that the frame is sent to the needed destination. In Ethernet networks NIC decide which node can transmit data (CSMA/CD). Figure (4.1) shows a PCI express NIC.

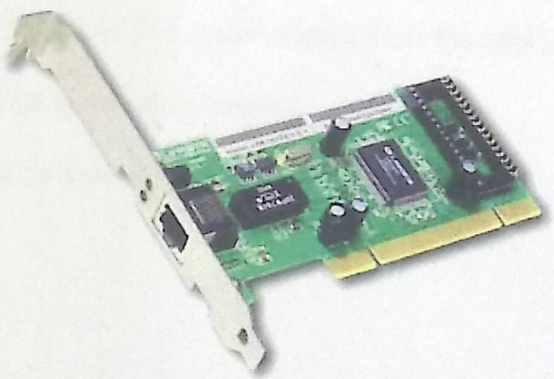


Figure (4.1) PCI Express NIC [3]

There are many types of network interface cards depending on its interfaces, speed, protocol, and other features. Some of NICs installed inside the node and some of it is attached externally such as laptops which use PCMCIA.

For NICs that installed internally, it is installed in the expansion slots on the system board. For workstations there are different types of expansion slots such as PCI, ISA, and EISA. PCI is introduced by Intel at the first time in 1992, it is 32 or 64 bit bus with 33 -66 MHz. clock speed. It is capable to transfer data at a rate of 264 MBps. So it is faster than ISA, and EISA expansion slots.

Intel introduces in 2002 a new version of PCI standard, which is PCI express. It is a 64 bit bus with a 133 MHz clock speed and transfer rate of 500 MBps in full duplex. PCI express network interface cards can be plugged in normal PCI slots, but it will work as a normal PCI NIC.

In our design it is more suitable to use PCI express network interface cards, but because we will use any available PCs, most of it will have a normal PCI slots. So we will use a normal PCI network interface cards.

We need 5 PCI (10/100) Mbps Ethernet network interface cards with RJ-45 connecters. It will be installed as following:

- Three interface cards for the three slave computers in the network.
- One interface card for the master computer.
- One interface cards for the request generator computer.

4.3 Cabling

In order to get high bandwidth and high performance fiber-optic cabling should be used to connect the master and the slaves to the switch, because it is the best media that can provide the required bandwidth and speed, but it is difficult for us to get fiber-optic cables, so we will use the best available cable which is category 5(cat 5) unshielded twisted-pair.

CAT 5 UTP can provide us with up to 100 Mbps throughput and a signal rate up to 100 MHz. CAT 5 contains four pairs of wires incased in a plastic sheath, each two pairs are twisted together to reduce crosstalk. The wires are twisted as following (orange and white orange), (brown and white brown), (blue and white blue), and (green and white green). Figure (4.2) shows a CAT 5 UTP cable example.

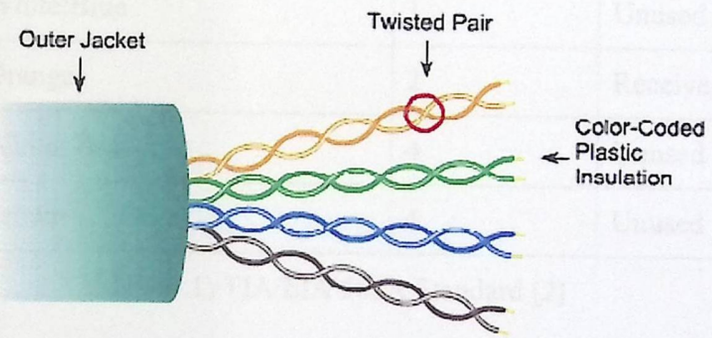


Figure (4.2) CAT 5 UTP [3]

CAT 5 UTP follows 100BaseT physical specification for Ethernet network so it is used in star physical topologies and uses the male RJ-45 connector type. It uses base band transmission with a maximum distance of 100 m. Female RJ-45 is used to plug the male RJ-45 in it.

4.3.1 Installing Cables

TIA/EIA has specified two standards for installing UTP cables: TIA/EIA 568A and TIA/EIA 568B. Any method can be used with no difference, but if one method is adapted, the same method must be used in all of the installation, as shown in table (4.1) and figure (4.3).

Pin Number	Color	Pair Number	Function
1	White/Green	3	Transmit
2	Green	3	Transmit
3	White/Orange	2	Receive
4	Blue	1	Unused
5	White/Blue	1	Unused
6	Orange	2	Receive
7	White/ Brown	4	Unused
8	Brown	4	Unused

Table (4.1) TIA/EIA 568A Standard [2]

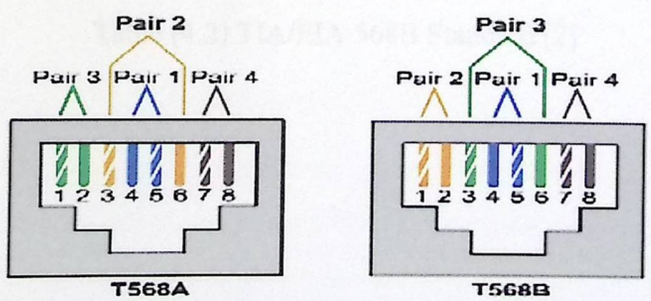


Figure (4.3) TIA/ EIA -568-B.1 [3]

A straight-through patch cable is used to connect the master, Packet generator and slaves PCs to the switch, both ends of this cable have the same order. When we connect the two PCs directly we use a crossover patch cable in which the termination locations of transmit and receive pins in one end are reversed, so one end of the cable follows TIA/EIA 568A and the other end follows TIA/EIA 568b standard, as shown in table (4.2)

Pin Number	Color	Pair Number	Function
1	White/ Orange	3	Transmit
2	Orange	3	Transmit
3	White/ Green	2	Receive
4	Blue	1	Unused
5	White/Blue	1	Unused
6	Green	2	Receive
7	White/ Brown	4	Unused
8	Brown	4	Unused

Table (4.2) TIA/EIA 568B Standard [2]

4.3 Block Diagram

Network-1, network-2 and network-3 are represented by packet generator. The packet generator will generate two types of packets, topology change packets, and normal data packets. Packets will be generated at random speeds and forwarded randomly to the three salves. Figure (4.4) shows a general block diagram.

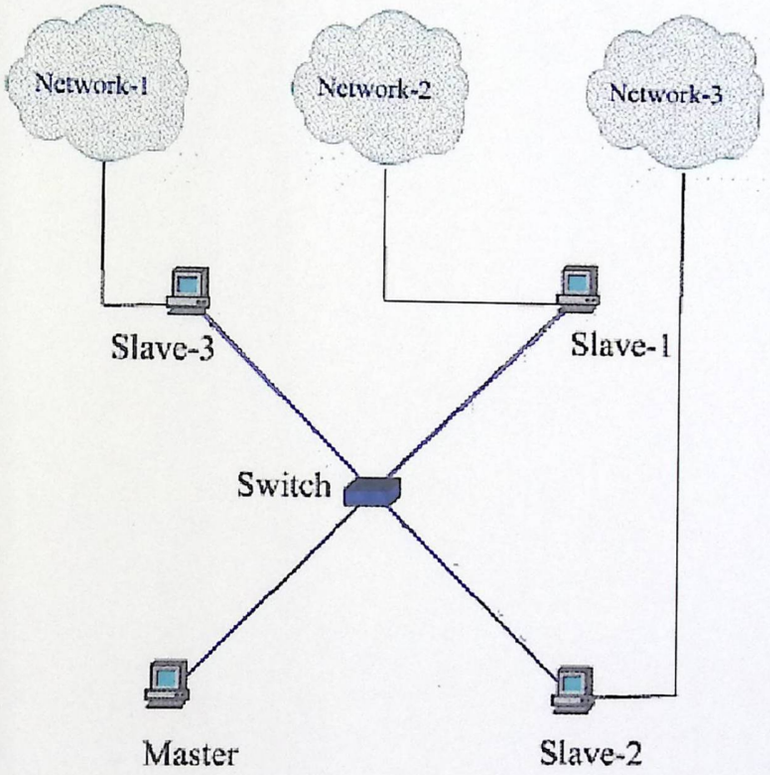


Figure (4.4) Block Diagram

Each slave has one interface cards to be connected to the switch. When the slave receives a packet, the slave chicks if the packet is a data or a topology change packet. If the packet is a data packet the slave forwards it to an appropriate path, else the slave will forward the packet to the master. The slaves send periodic messages to each other about their free memory status.

The master has one interface card to connect the master to the switch. The master receives the topology change packets from the slaves and uses it to update routing table.

The switch is used to connect the Packet generator, slaves and master together. It has 8 ports. The switch works in cut-through mode

Chapter Five

Software System Design

Chapter Five

Software System Design

To build this parallel router we have build three parallel algorithms. These algorithms require an operating system, programming language, and an appropriate packet format. The flowing is a discussion of these requirements:

5.1 Operating System

We will use Linux operating system in our simulation. We choose Linux because it has many features that support our project. The following are some of the features of Linux that support our project:

- Open source: Because Linux is open source, makes it easy for us to troubleshoot and solving any problem related to operating system that may face us in the project.
- Security: Linux provide us with a high level of security that is necessary for our project
- Free license: we don't have to pay much to use a copy of Linux operating system; this will reduce the cost of the project.
- High performance: Linux is a fast operating system. Our project require a high performance operating system, the speed is a key factor in routing process.
- Flexible: Linux supports libraries such as MPI. We will use some of these libraries in our project.

5.2 Programming Language

We will use C programming language. C is very fast programming language and supports procedural programming. It supports low level programming like assembly. This is an important feature; because in our project we have to deal with ports and other hardware components of computers. C supports a wide variety of libraries. Documentation of C under Linux is available; this makes it easy for us to use this programming language, in addition we have some experience of this programming language.

5.2 Protocol (TCP/IP)

We will use the TCP/IP standard, because it is the most important standard for internetwork communications and serves as the transport protocol for the Internet. It is widely used protocol in all over the world. The router uses it as a configuration tool. TCP/IP supports all standard physical and data link protocols and suitable for both LAN and WAN communication.

TCP/IP model consists of four conceptual layers that can be mapped to the OSI reference model as shown in the figure (5.1). It is consists o the following layers:

5.3.1 Application Layer

The application layer provide protocols that support email (SMTP) ,file transfer such as HTTP, TFTP, and FTP, network management (SNMP), name management (DNS), and remote login (Telnet). TFTP, DNS, SNMP, and Telnet are used by the router.

Figure (5.1) shows how TCP/IP model is mapped to OSI Model

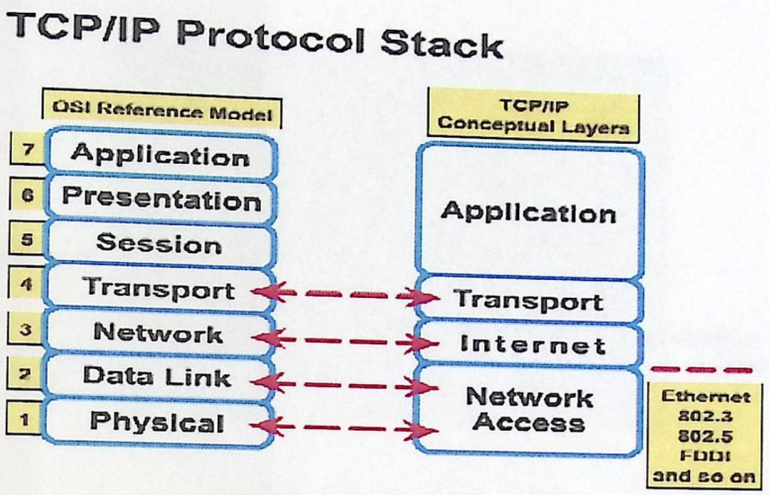


Figure (5.1) TCP/IP Mapped to OSI Model [3]

5.3.2 Transport Layer

The transport layer provides flow control, and reliable transfer of data. It includes two protocols TCP and UDP. TCP and UDP use port (or socket) numbers to pass information to the upper layers. Port numbers are used to manage requests from different applications that cross the network at the same time. Port numbers have the following ranges:

Numbers below 255 are for public applications.

Numbers 255-1023 are assigned to companies for marketable applications.

Numbers above 1023 are unregulated, and usually assigned by the source host.

Figure (5.2) shows popular will known ports numbers.

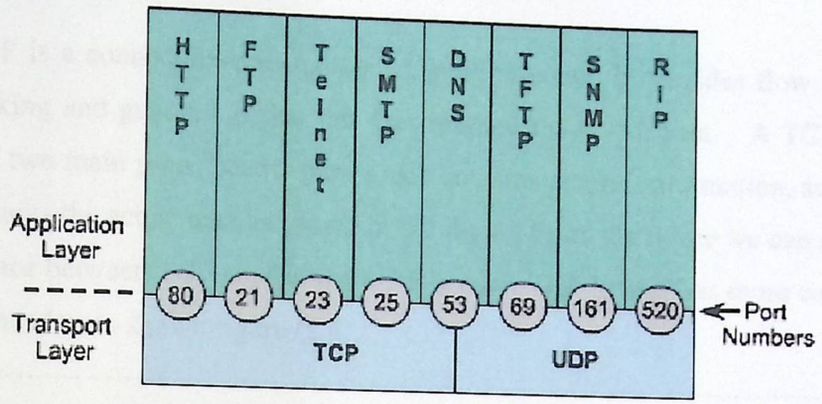


Figure (5.2) Ports Numbers [3]

5.3.2.1 UDP

UDP is a fast connectionless protocol. It does not provide software checking so it is unreliable. Because of its speed UDP is used in broadcasting. A UDP segment consists of two main parts, header part which contains control information, and data part which contains the actual data, as shown figure (5.3).

Bit #	0 ……16……31	
UDP Header	Source Port	Destination Port
	Length	Checksum
	Data	
	Data	

Figure (5.3) UDP Segment [2]

5.3.2.2 TCP

TCP is a connection-oriented and reliable protocol. It provides flow control and error checking and guarantees that the data reached the destination. . A TCP datagram consists of two main parts, header part which contains control information, and data part which contains the actual data as shown in the figure. From the figure we can easily notes the difference between TCP header and UDP header. TCP header has more control fields than UDP header, as shown figure (5.4).

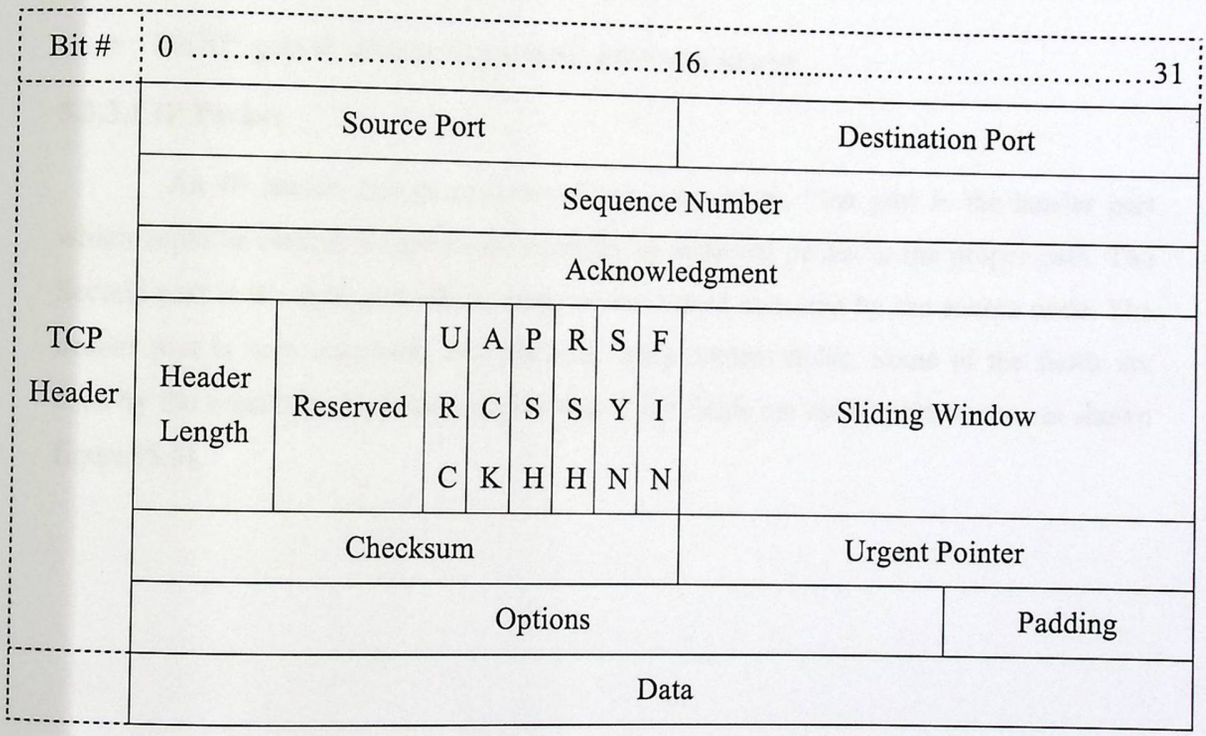


Figure (5.4) TCP Datagram [2]

5.3.3 Internet Layer

It is related to the network layer of the OSI reference model. It provides logical addressing. The router works on this layer. It provides four protocols:

- IP: a connectionless protocol, it uses the logical address to route the packets.
- ICMP :provides control and messaging capabilities
- ARP: gets MAC when IP address is known.
- RARP: gets IP address when MAC address is known.

5.3.3.1 IP Packet

An IP packet format consists of two main parts. First part is the header part which contains control information necessary to route the packet to the proper path. The Second part is the data part which contains the actual data sent by the source node. The header part is very important and contains many control fields. Some of the fields are used by the receiving workstation and some of the fields are used by the router, as shown figure (5.5).

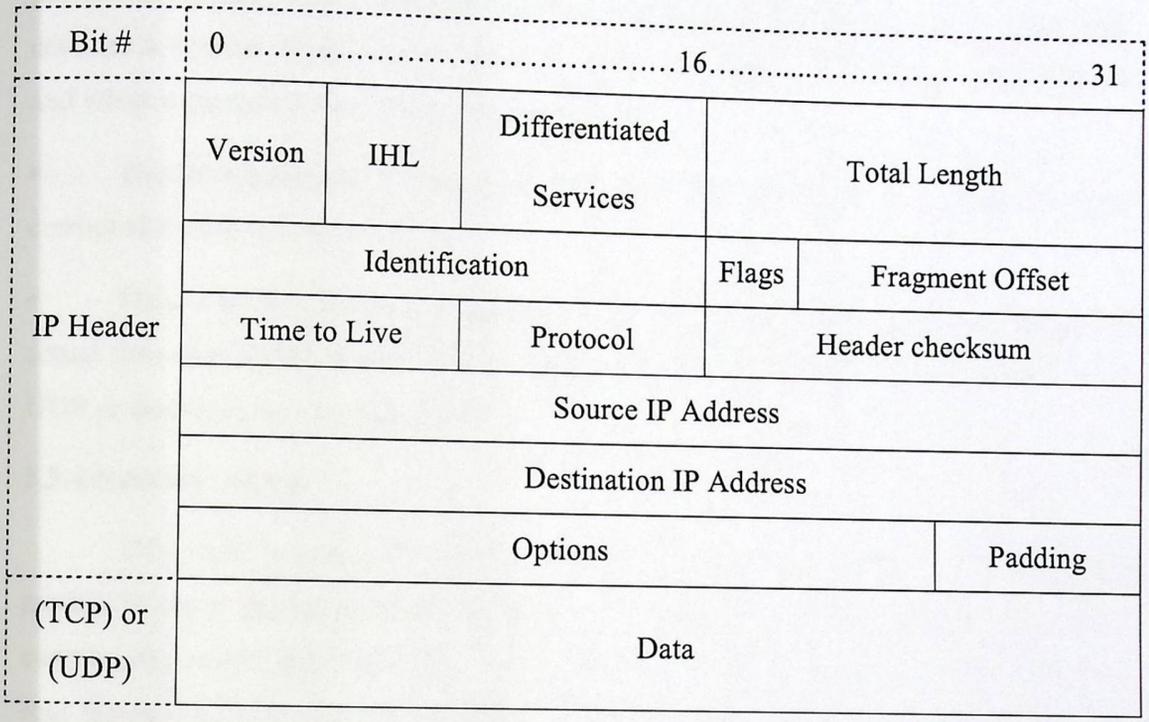


Figure (5.5) IP Datagram [2]

The following fields are the most important fields in our project:

- Source IP Address: A four bytes field which contains a logical address identifies the node which sent the packet.
- Destination IP Address: A four bytes field which contains a logical address identifies the node which will receive the packet.
- Header Length: A four bits field indicates to the receiving node where data will begin after the header end.
- Protocol: A one byte field indicates the type of packet (topology change packet or data packet).

- Time to Live: A one byte field which indicates the maximum time that the IP packet will remain in the network before it will be discarded by the router. This field contains a counter decreased by 1 by each router in the path. The counter starts with 64 and when it reaches 0 it is discarded by the router.

- Header Checksum: A two bytes field indicates whether the IP header has been corrupted during the transmission.

- Data: The size of this field depends on the data which may contain. It contains the actual data sent by the source node in addition to control information added by TCP or UDP protocols in the transport layer.

5.3.4 Network Access

This layer is mapped to the physical and data link layers of the OSI reference model. When a packet reaches this layer it is encapsulated into a frame. The frame contains the source and destination MAC addresses. The router uses the MAC address to rout the packet from one hop to another. When the frame reaches the router, the router remove the frame header and encapsulate the packet into new frame header with destination MAC address is changed to the next hop MAC address. Frame Format varies widely depending on the used technology. A generic frame format is shown in figure(5.6).

Ethernet					
8	6	6	2	64 to 1500	4
Preamble	Destination Address	Source Address	Type	Data	Frame Check Sequence

Figure (5.6) Generic Frame Format [2]

5.4 Algorithms Design

Figure (5.7) shows the functions of the three main algorithms.

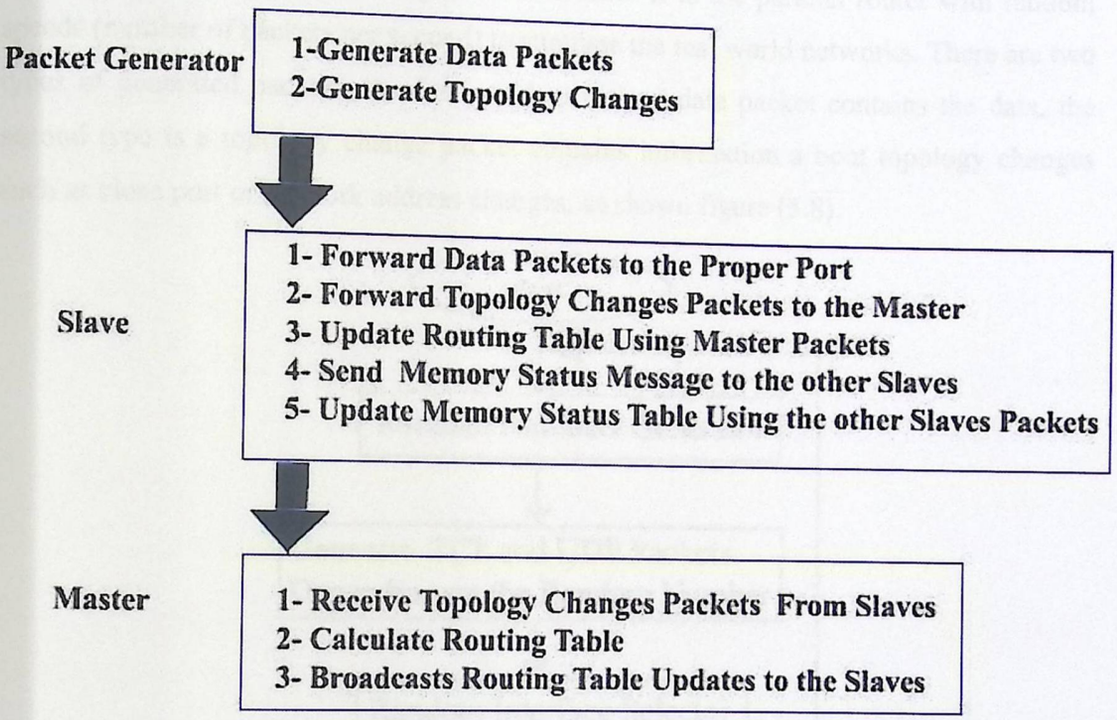


Figure (5.7) Algorithms Functions

5.4.1 Packet Generator

This algorithm generates packets and sends it to the parallel router with random speeds (number of packets per second) to simulate the real world networks. There are two types of generated packets, the first one is a normal data packet contains the data, the second type is a topology change packet contains information a bout topology changes such as close port or network address changes, as shown figure (5.8).

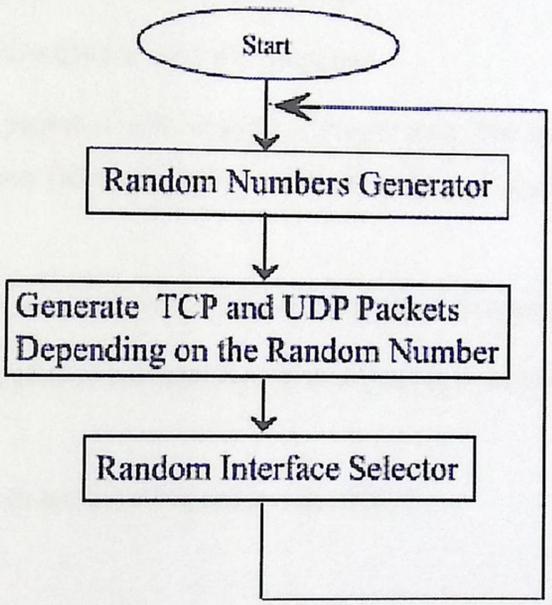


Figure (5.8) Packet Generator Flowchart

5.4.2 Slave Algorithm

This algorithm forwards the incoming packet proper path. The protocol field is used to determine whether the incoming packet is a Topology Change packet or a Normal Data packet or a Master packet. The forwarding process has the flowing scenarios:

- If the incoming packet is a data packet then:
  - If it's Time to live field invalid it is discarded..
  - If it is a valid packet it is forwarded to proper path. The path may be one of the ports on the same slave or may be one of the ports on another slave in the parallel router.
- If it is a Master packet, it would be used to update the routing table.
- If the incoming packet is a topology change packet, it would be sent to the master personal computer.

Figure (5.9) displays a flowchart of the slave algorithm.

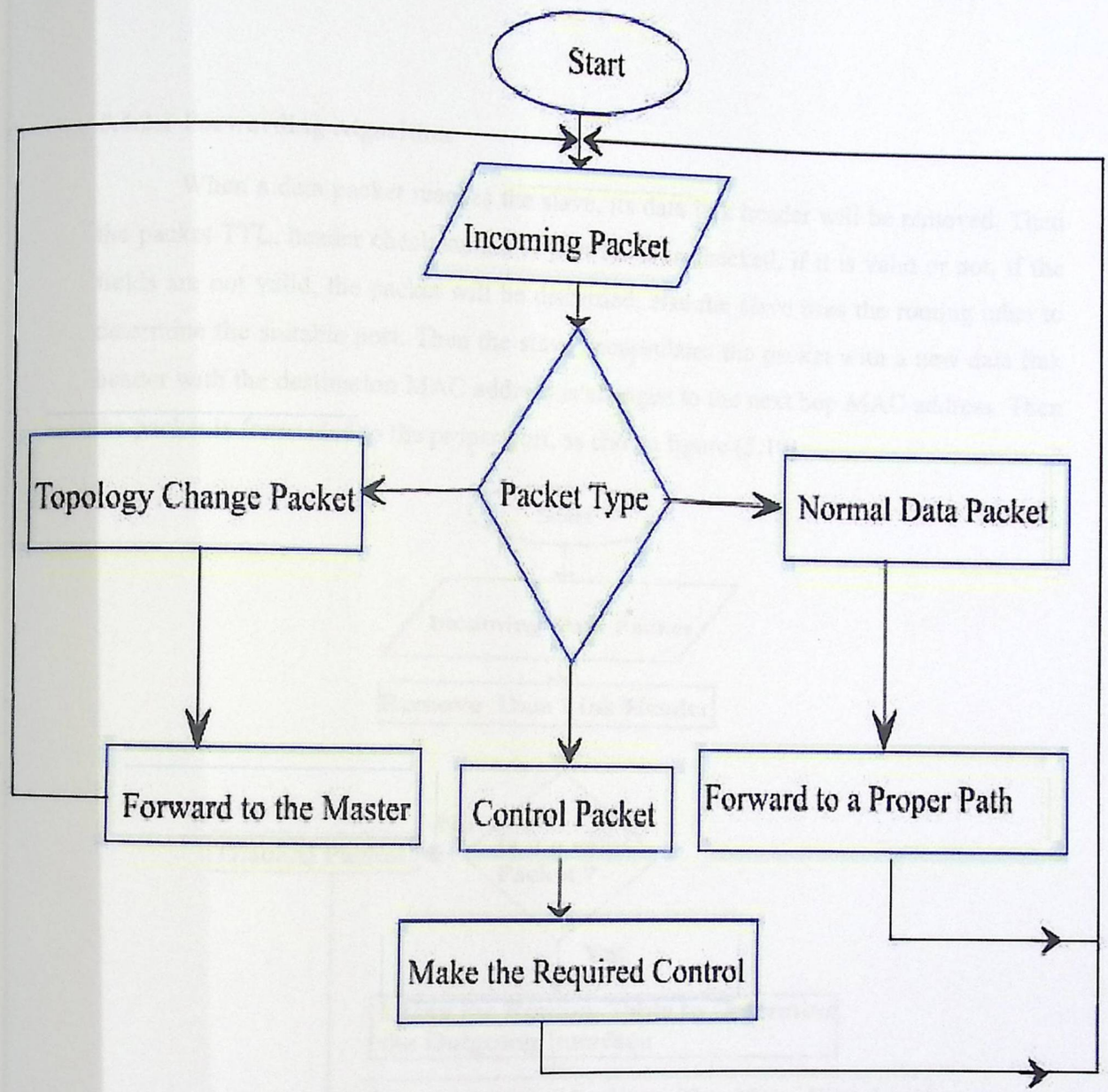


Figure (5.9) Slave Flowchart

5.4.2.1 Forwarding Algorithm

When a data packet reaches the slave, its data link header will be removed. Then the packet TTL, header check sum, and port fields is checked, if it is valid or not. If the fields are not valid, the packet will be discarded, else the slave uses the routing table to determine the suitable port. Then the slave encapsulates the packet with a new data link header with the destination MAC address is changed to the next hop MAC address. Then the packet is forwarded to the proper port, as shown figure (5.10)

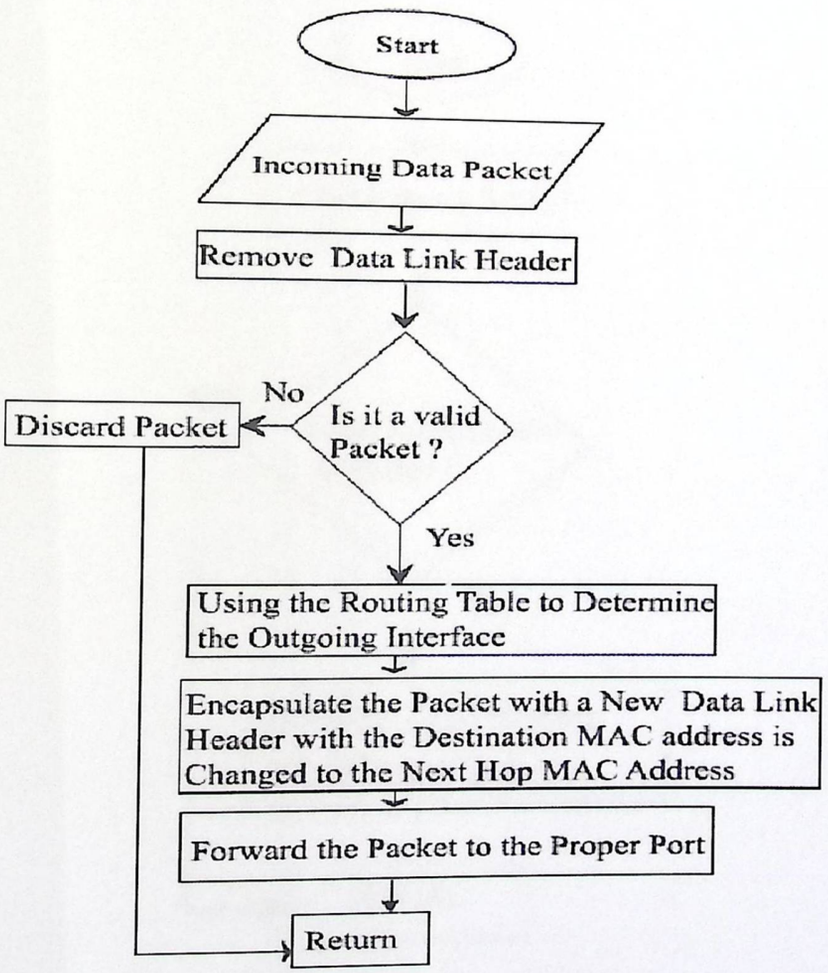


Figure (5.10) Forwarding Flowchart

5.4.2.2 Load Balancing:

Load balancing algorithm. Load balancing algorithm checks the status table if there are underloaded slaves, if all the slaves are overloaded, the algorithm will return without making load balancing. If one or more underloaded slaves available the overloaded slave will distribute the load on the underloaded slaves, as shown figure(5.11).

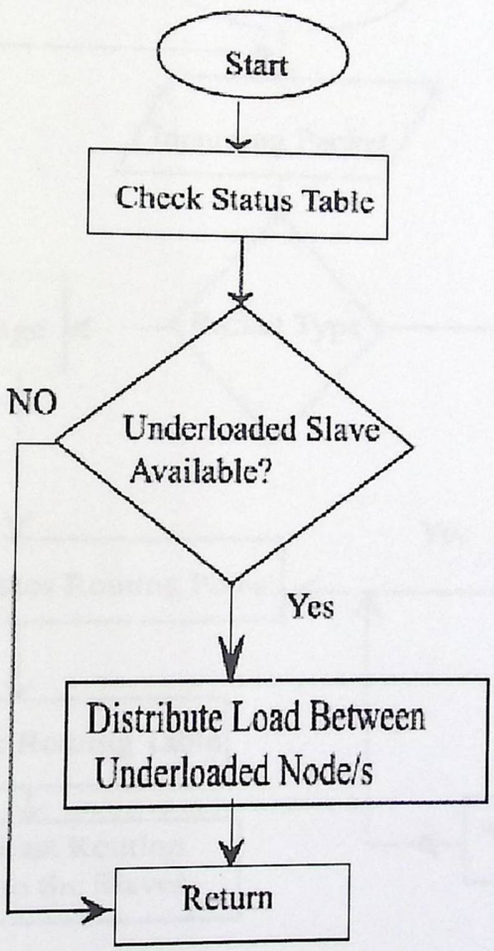


Figure (5.11) Load Balancing Flowchart

5.4.3 Master Algorithm

The master is responsible for updating the routing table and calculating the best path. This is a crucial task in the routing process. The master receives the topology changes packets and recalculates the best routing paths and makes the necessary updates to the routing table then broadcasts the routing table updates to the slaves, as shown figure (5.12).

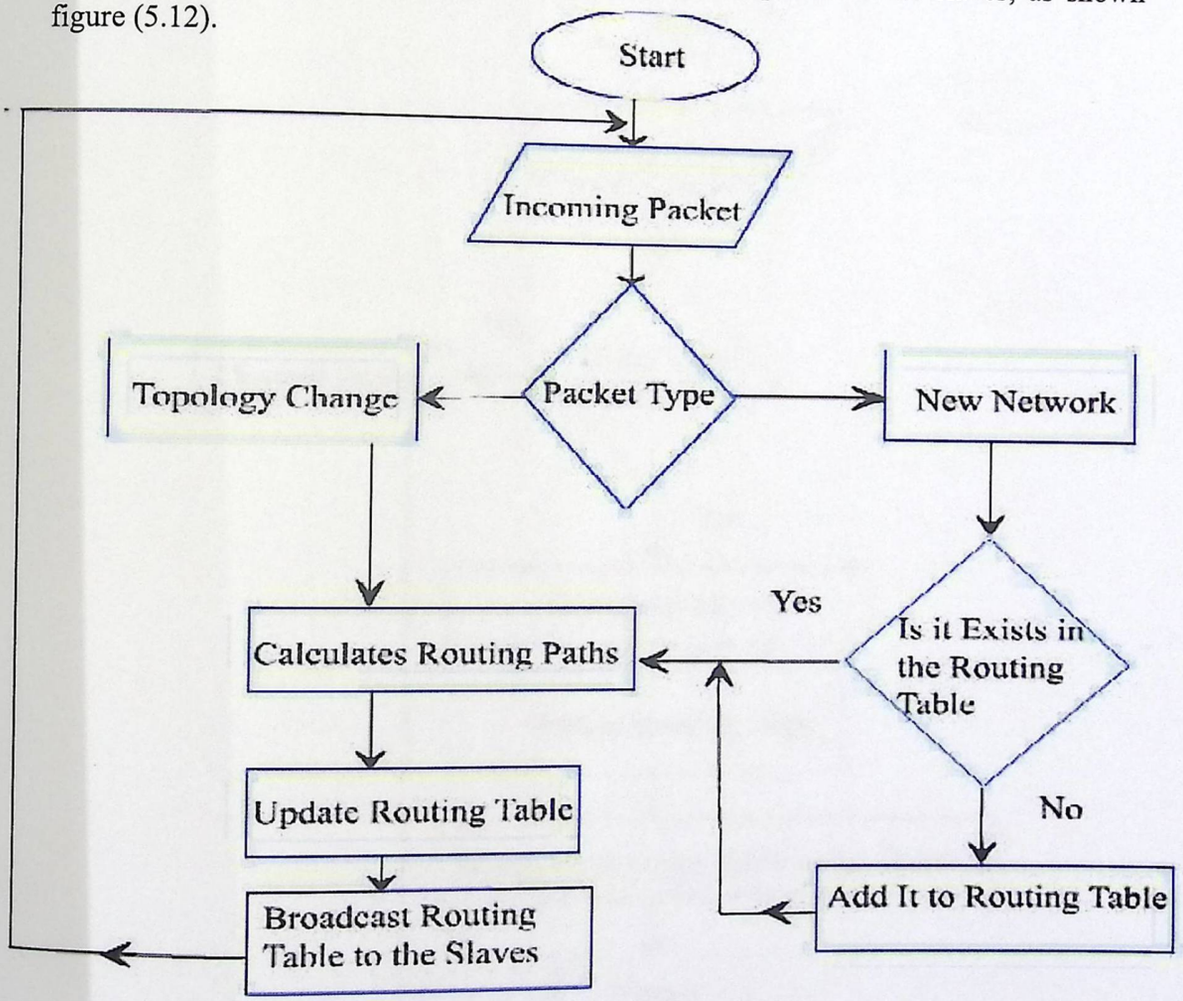


Figure (5.12) Master Flowchart

5.4.3.1 Detailed Path Determination

When a topology changes packet reaches the master its header examined if it is valid or not, if it is invalid it will be discarded, else if it is a valid packet it will be used to calculate best paths. Then the results will be used to update routing table. Then the master broadcasts the routing table to the slaves, as shown in figure (5.13).

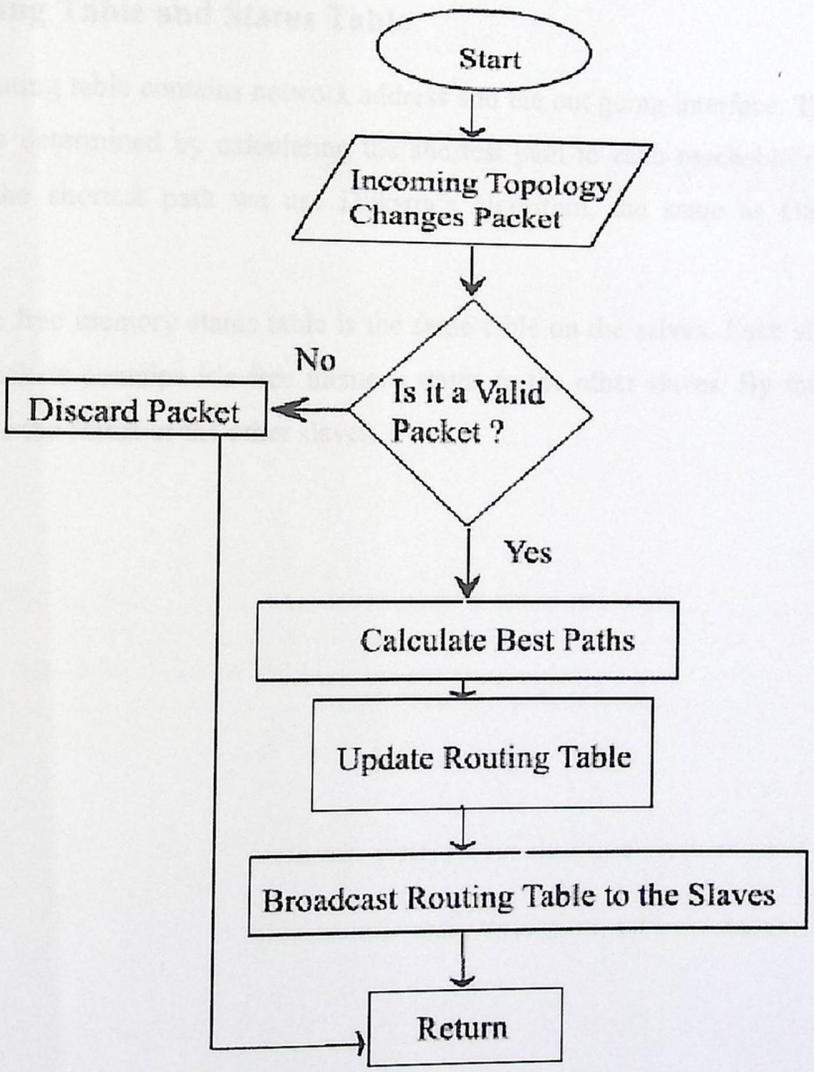


Figure (5.13) Path Determination Flowchart

5.5 Programming Model

We will use single program multiple data (SPMD) model and will be implemented using massage passing interface (MPI). We choose MPI because it is suitable for distributed memory architecture and has binding with C language, which is (MPI.h) which can be included in our project. Documentation of (MPI) is available.

5.6 Routing Table and Status Table

Routing table contains network address and the out going interface. The outgoing interface is determined by calculating the shortest path to each reachable network. To calculate the shortest path we use Dijkstra's algorithm, the same as OSPF routing protocol.

The free memory status table is the same table on the salves. Each slave sends a periodic packets contains it's free memory status to the other slaves. By this way each salve knows the status of the other slaves.

Chapter Six

Implementation and Testing

Chapter Six

Implementation and Testing

This chapter demonstrates the detailed steps of implementing the project in addition to the testing procedures.

6.1 Implementation

The implementation consists of two parts, hardware implementation and software implementation

6.1.1 Hardware Implementation

This phase is the base on which we implemented the simulation. We installed the network that connects the 5 PCs. Installing the network requires preparation and testing of the network devices .The following are more details a bout this phase

6.1.1.1 Computers

The university provided us with five Pentium 4 PCs which have the specifications shown in table(6.1).

Processor Type	Intel® Pentium® 4
Processor Speed	2.8 GHz
RAM	256 MB (DDR 400)
System Bus Speed	800 MHz
H.D	40 G.B
Ethernet Controller	On Board Intel® Pro Ethernet controller

Table (6.1) Computers Specifications

This is a general specification, the 5 PCs vary in their main memory size, and some of them have a memory size of 128 MB and some of the have 512 MB of main memory. This property enabled us to build a nonhomogeneous cluster of PCs.

6.1.1.2 Network Devices

We used a cut-through Ethernet Switch with 8 ports to connect the 5 PCs. To install the switch we connect one end of the network cable to the RJ-45 port on the rear panel, and the other end of the network cable to the Rj-45 port on the PC. We do the same thing with the remaining PCs.

Once the network cable is connected to both ends and the switch and the PC is powered on, the green LNK/ACT LED should be lit.

Table (6.2) shows a description of the LED's of the Switch

LED	Color	Status	Description
PWR(power)	Green	Lit	Power is Supplied
		Off	No Power
LNK/ACT Link/Activity)		Lit	A Valid Link is Established
		Flash	Data Packets Received
		Off	No Link is Established

Table (6.2) Switch LED Description

6.1.1.3 Cabling

We used a straight-through cat 5 UTP patch cables to connect the 5 PCs to the switch. At the two ends of the cable a male RJ-45 connector is installed. The five PCs are connected in physical star topology.

The UTP network cable must comply with EIA/TIA 568 specifications and category 5 standard for 100Mbps data transmission. Maximum length using UTP cable between the switch and the PCs is 100 meters (300ft).

Before installing any cable, it must be checked for their connectivity using the cable connectivity tester

6.1.2 Software Implementation

In this phase we install and configure the operating system and other software packages. Also we implement the simulation.

6.1.2.1 Operating System Installation

Fedora core 3 is a new version of Linux which is an upgrade version of Red Hat 9. Fedora installation comes with 4 CD's which can be downloaded freely from the internet. The following packages must be installed to implement the simulation of the router:

- GText Editor (gedit): we use it to write the code of the simulation and save it in a file with (.c) extension.
- GCC compiler: we use it to compile the C code.
- LAM package: this package is the base on which MPI works.
- MPI package: this package contains a header file which contains special functions to send and receive information packets between the PCs. In addition it contains other facility functions.

6.1.2.2 Operating System Configuration

After installing the operating system and the required packages, The following configurations of the operating system must be performed

6.1.2.2.1 TCP/IP Configuration

Each PC must be configured with a unique static IP address in addition a good host name should be assigned to each PC. We can configure the TCP/IP and other network related configuration by clicking on applications →system settings →network, or by simply run the following command on the shell:

```
[root@localhost ~]# system-config-network
```

After executing this command the screen figure (6.1) will appear:

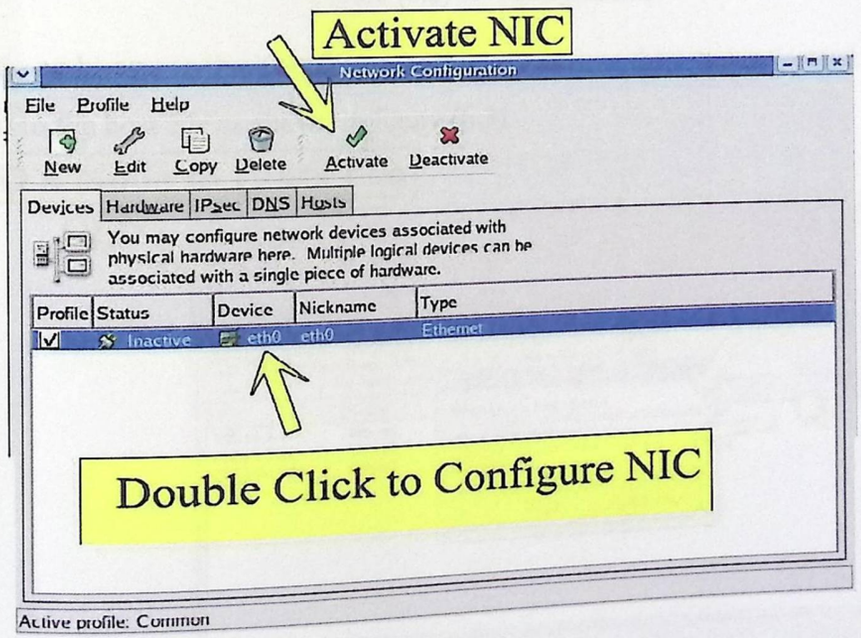


Figure (6.1) Network Configuration

By double clicking on the NIC a screen which enable us to set an IP address and a subnet mask to the host as shown in figure (6.2).

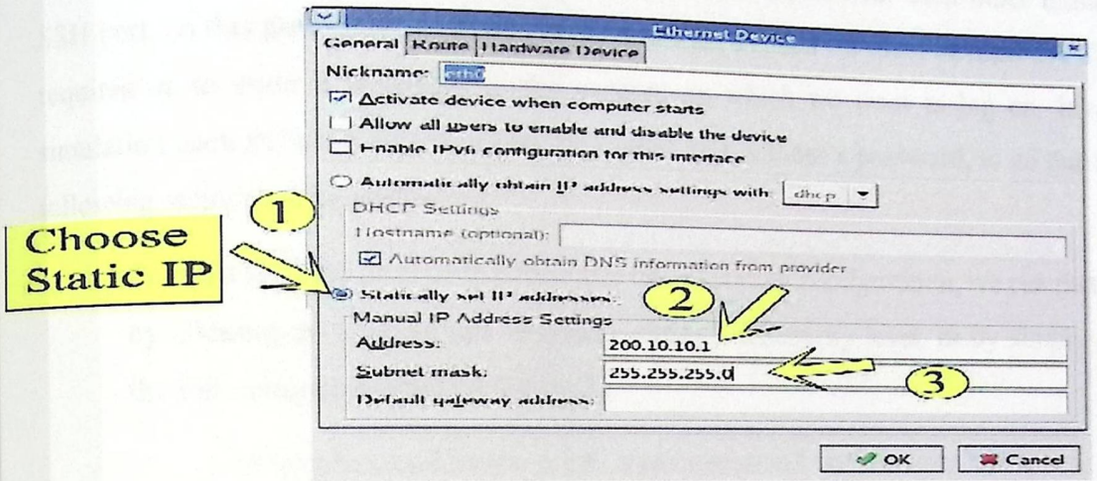


Figure (6.2) IP Configuration

To add hosts to the host file and deal with it using their names instead of their IP we add it to the host file as shown in figure (6.3).

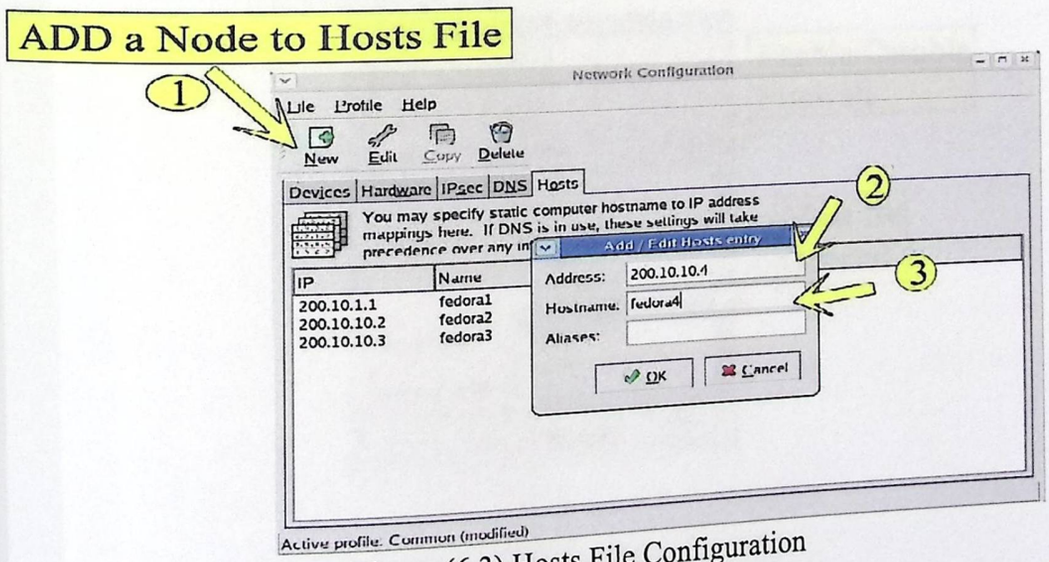


Figure (6.3) Hosts File Configuration

6.1.2.2.2 SSH configuration

To compile the code the five PCs must communicate with each other through SSH port, so this port must be opened or the fire wall on each PC must be disabled. SSH requires us to enter a password to the account on which we want to log on. In our simulation each PC must communicate with other PCs without a password, to do that the following steps must be performed:

1. To open ssh port or disable firewall or other security configuration, we can do that by clicking on applications →system settings →Security level, or by simply run the following command on the shell:

```
[root@localhost ~]# system-config-securitylevel
```

After executing this command the screen in figure (6.4) will appear.

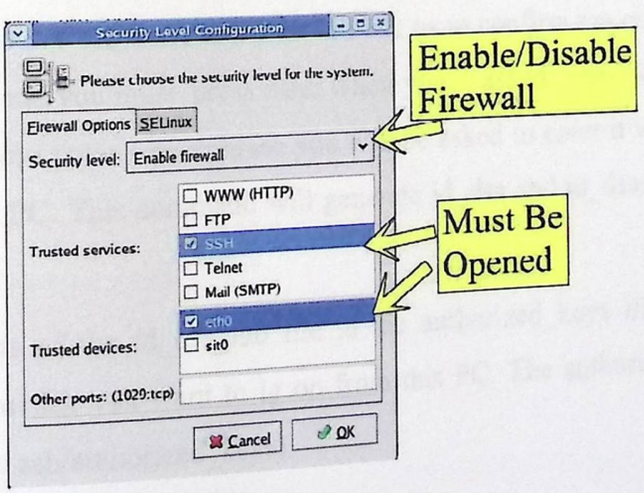


Figure (6.4) Firewall Configuration

2. Generate a public and private key for each PC. To do that, we log on to the account through which we make the simulation on each PC; on the terminal of each PC we write the command shown in figure (6.5).

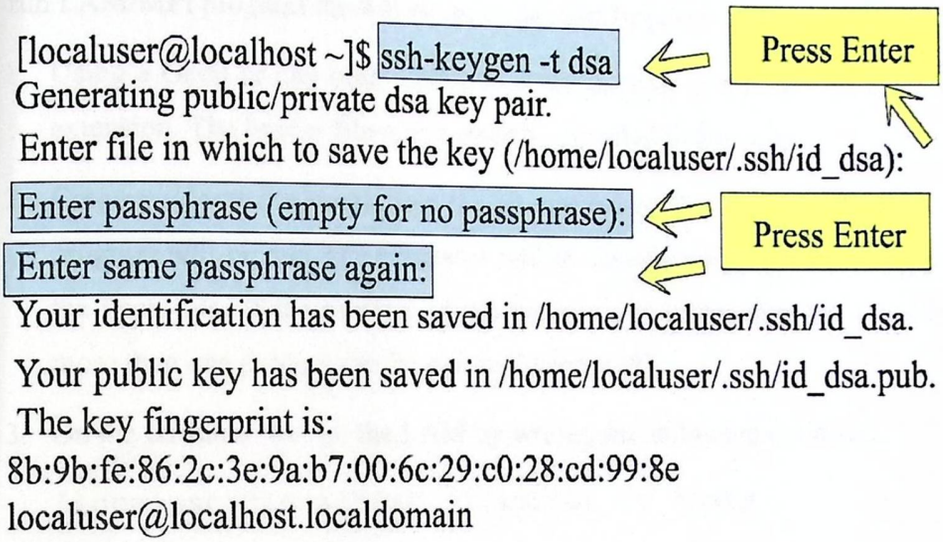


Figure (6.5) SSH Key Generation

We use dsa since we use ssh version 2 this will ask us to confirm the creation of private and public keys and you must press enter when you asked to enter a passphrase, because if you enter a passphrase you will be asked to enter it when you try to log on to another PC. This command will generate id_dsa and id_dsa.pub files on (/user/.ssh/).

3. Append the contents of the id_dsa.pub file to the authorized_keys file on the destination PCs to which you want to lg on from this PC. The authorized_keys file located on (user/.ssh/authorized_keys).

6.1.2.3 Programming Language

We used a GCC compiler to compile a LAM/MPI program. To write and compile and run LAM/MPI program the following steps must be performed:

1. Using a Gedit or any other editor to write the code and save it as a file with (.c) extension. The header file mpi.h must be included in the code.
2. Creating Hosts file containing the IP address of the PCs on which the parallel program will be run. The processes will be distributed on the PCs as specified on the hosts file. If the number of the processes is grater than the number of PCs more than one process can be run on the same PC.
3. On the terminal we run the LAM by writing the following command:

   ```
   [localuser@localhost ~]lamboot -v hosts
   ```

   Hosts is the name of the host file which contain the IP addresses of the 5 PCs on which LAM will be run. LAM can't be run on the root account because it interferes with system privileges, so it should be run on normal user account.
4. To compile the source code file we write the following command:

   ```
   [localuser@localhost ~]$ mpicc -o PROG CODE.C
   ```

   Where PROG is the name of the compiled out put file and CODE.C is the name of the source code file.
5. To run the compiled code we write the following command

   ```
   [localuser@localhost ~]$ mpirun -np n PROG
   ```

   Where n is the number of the processes and PROG is the compiled source code.
6. After each `mpirun` command It is preferable to run the command:

   ```
   [localuser@localhost ~]$ lamclean
   ```

7. To turnoff LAM we run the command:

```
[localuser@localhost ~]$ wipe -v hosts
```

6.1.2.4 Data structures

This section demonstrates the used data structures in the project algorithms implementation

6.1.2.4.1 Routing table

We use a struct to represent the routing table. To represent the IP address we use an array of four elements inside the struct, each element represents an octet of the IP address. To represent the out going interface we use another short integer.

6.1.2.4.2 Sending and Receiving Buffers

We defined buffers as arrays of pointers of TCP and UDP types. We use it to sending and receiving of packets. First we initialize these buffers by a NULL value and when we need to use one element of the buffer we call a function to allocate the needed memory space and making the pointer of the buffer pointing to that memory space.

6.1.2.4.3 Adjacency Matrix

To represent the topology of the network we use a two dimensional array as adjacency matrix. the columns and rows of the matrix represents routers and networks, the intersection between the columns and the rows represent the link between the two nodes, if there is no link between the two nodes (-1) will be assigned, if there is a link the cost of the of the link will be assigned as shown in the figure(6.6):

.....From.....

TO		Net0	Net1	Net2	Net3	Net4	Net5	Net6	Net7
	Net0	0	-1	-1	-1	40	-1	-1	-1
	Net1	-1	0	23	-1	-1	-1	12	90
	Net2	-1	23	0	-1	-1	36	-1	-1
	Net3	-1	-1	-1	0	-1	51	-1	-1
	Net4	40	-1	-1	-1	0	-1	-1	-1
	Net5	-1	36	-1	51	-1	0	-1	-1
	Net6	-1	12	-1	-1	-1	-1	0	-1
	Net7	-1	90	-1	-1	-1	-1	-1	0

Figure (6.6) Adjacency Matrix

6.1.2.4.4 TCP and UDP Packets

A data frame encapsulates an IP packet. An IP packet encapsulates a TCP or UDP datagram. We build two types of frames one for TCP datagram and another for UDP datagram. We use a different struct to represent each one of the frames. The struct fields simulate the actual frame, IP and (TCP or UDP) header fields. The routing process depends mostly on the IP address and don't care about the data contents, so we initialize the frames with random data.

6.1.2.4.5 Contiguous Datatypes

MPI_Send and MPI_Recv subroutines deal only with contiguous datatypes in send and receive operations. We use struct to represent the packets which are noncontiguous datatype, so we have to convert it to contiguous datatypes, to do that we use MPI_Type_struct subroutine to get anew contiguous datatype from the old datatype.

6.1.2.5 Used MPI Subroutines

Subroutine	Description
MPI_Send	Blocking standard mode send operation
MPI_Recv	Blocking receive operation
MPI_Iprobe	Checks if a message matching source Tag and comm. has arrived
MPI_Type_struct	Returns a new datatype that represents count blocks each with a distinct format and offset
MPI_Type_extent	Returns the extent of any defined datatype
MPI_Type_commit	Makes a datatype ready for use in communications
MPI_Wtime	Management Returns the current value of time as a floating point value
MPI_Type_free	Marks a derived datatype for deallocation and sets its handle to MPI_DATATYPE_NULL
MPI_Init	Initializes MPI
MPI_Finalize	Terminates all MPI processing
MPI_Comm_rank	Returns the rank of the local task in the group associated with a communicator

Table (6.3) MPI Subroutines Reference

6.1.2.6 Algorithms Implementation

6.1.2.6.1 Main Function

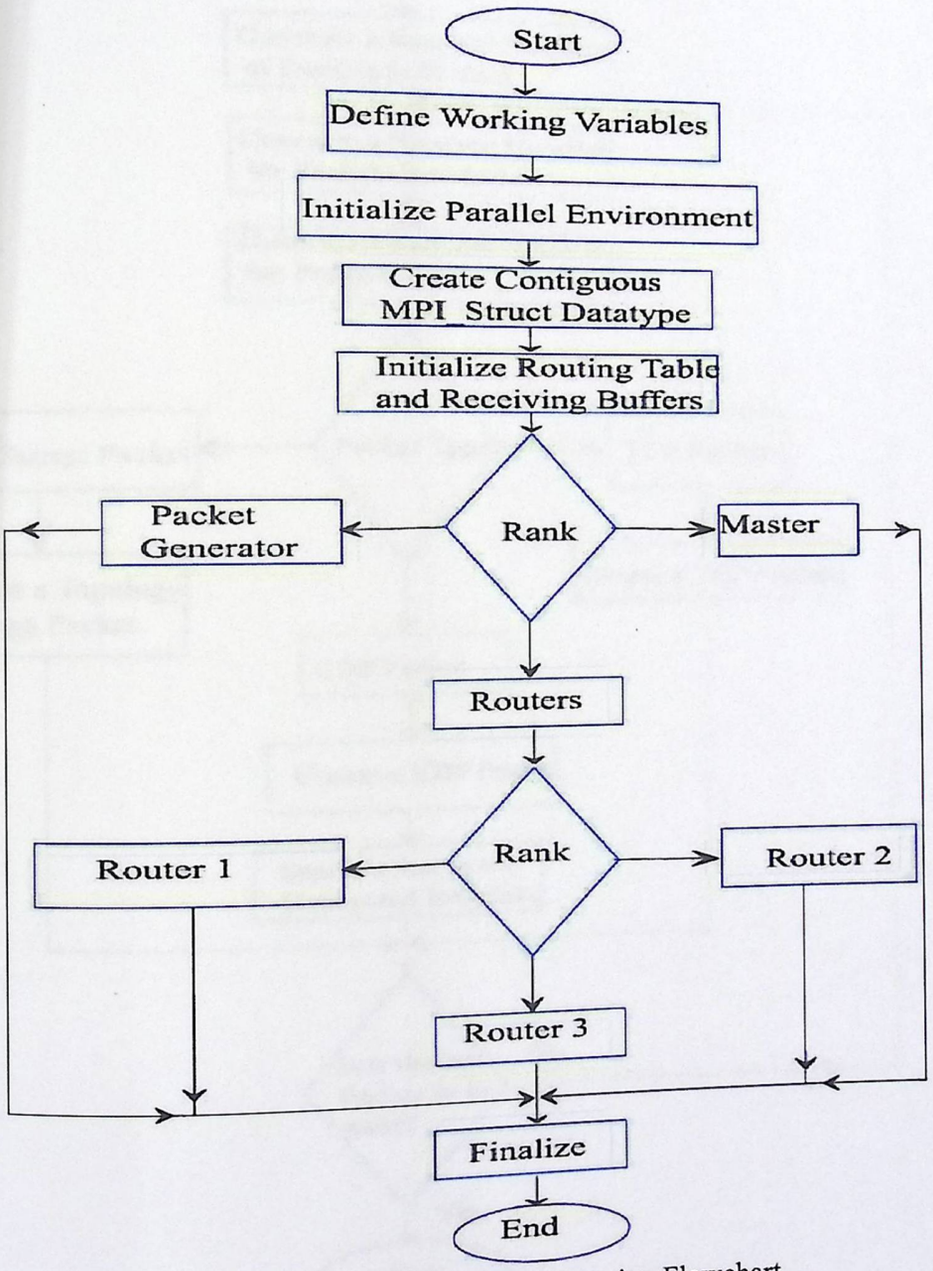


Figure (6.7) Main Function Flowchart

6.1.2.6.2 Packet Generator

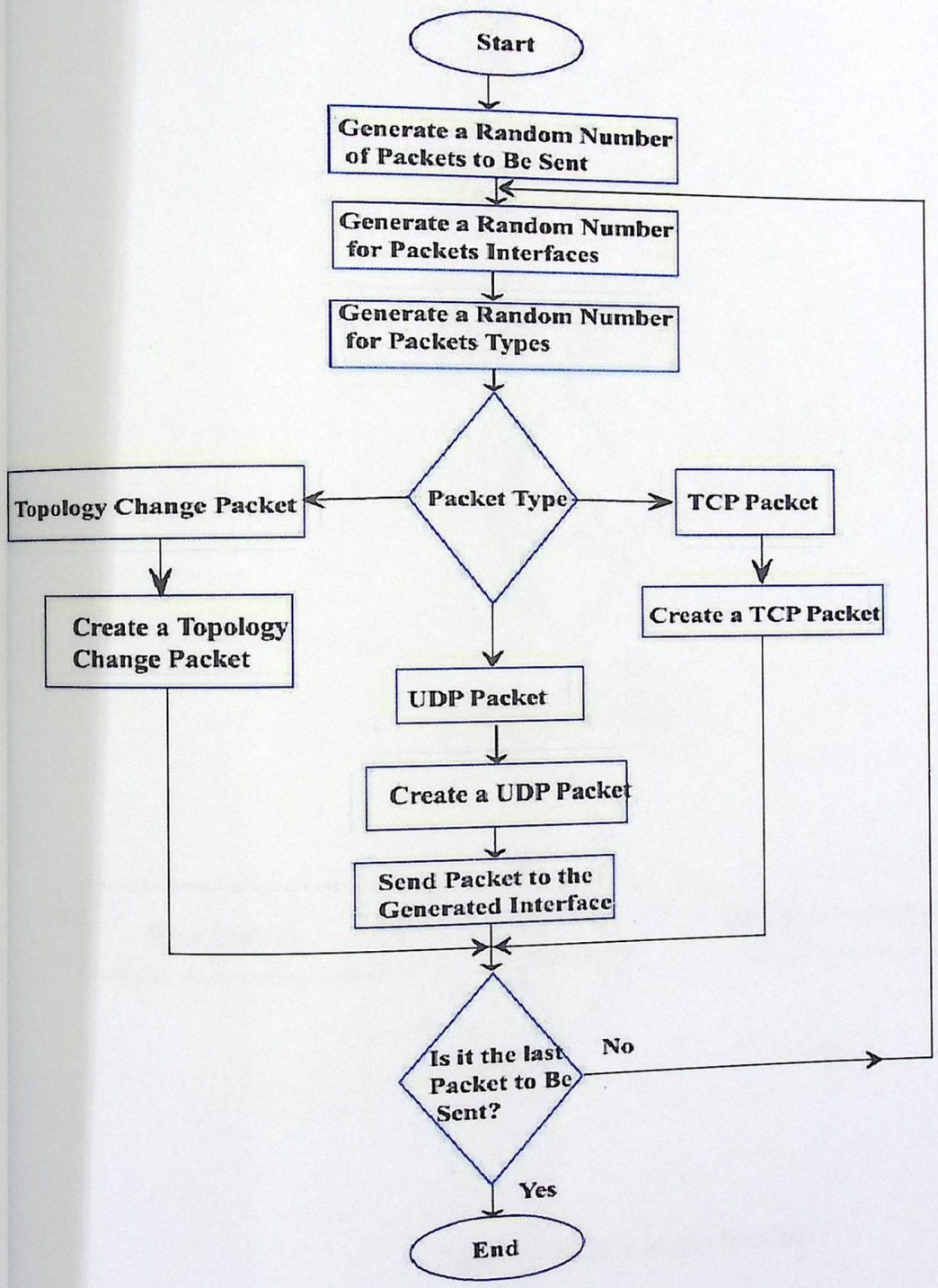


Figure (6.8) Packet Generator Flowchart

6.1.2.6.3 Slave Work

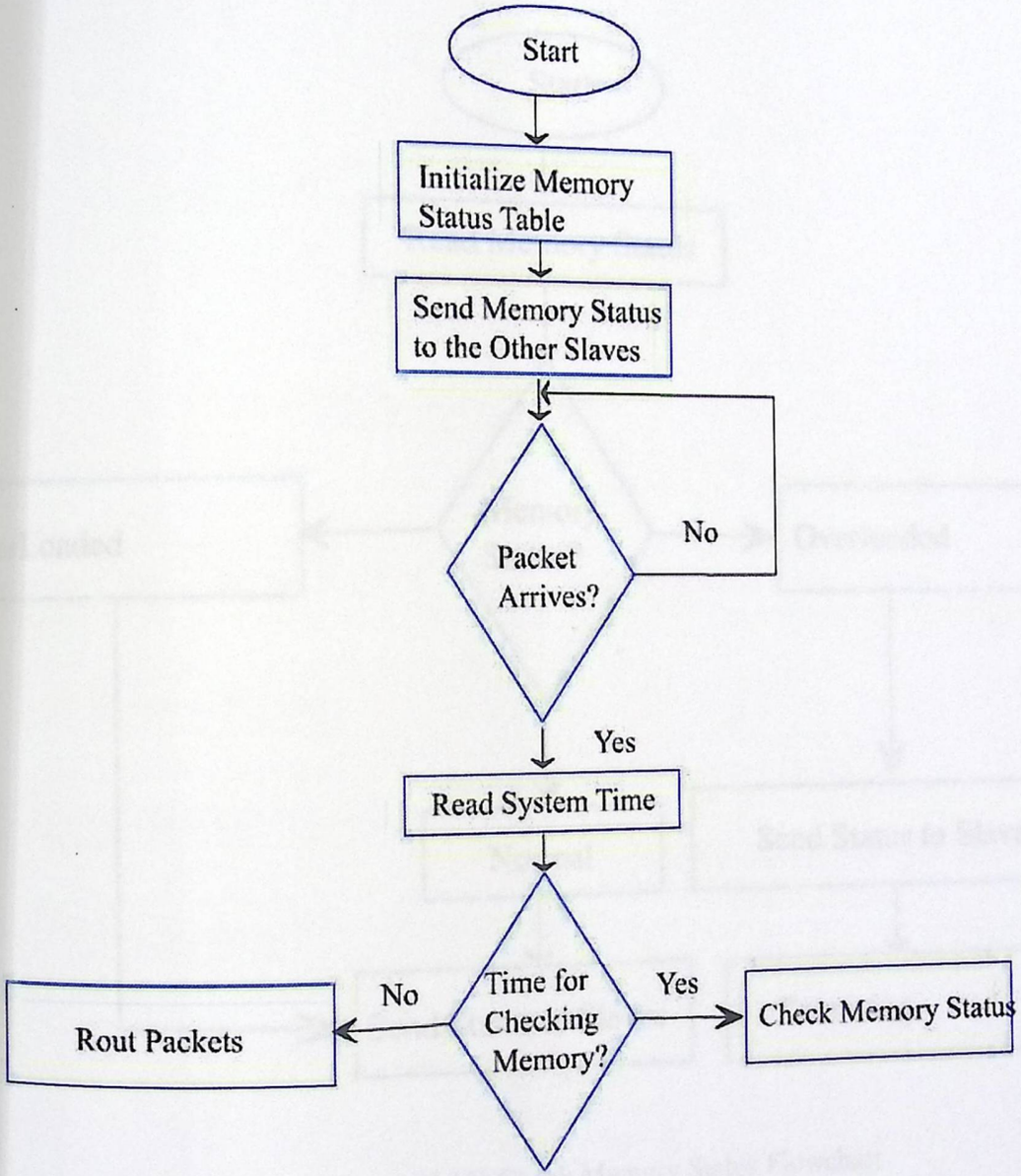


Figure (6.9) Slave Work Flowchart

6.1.2.6.4 Check Memory Status

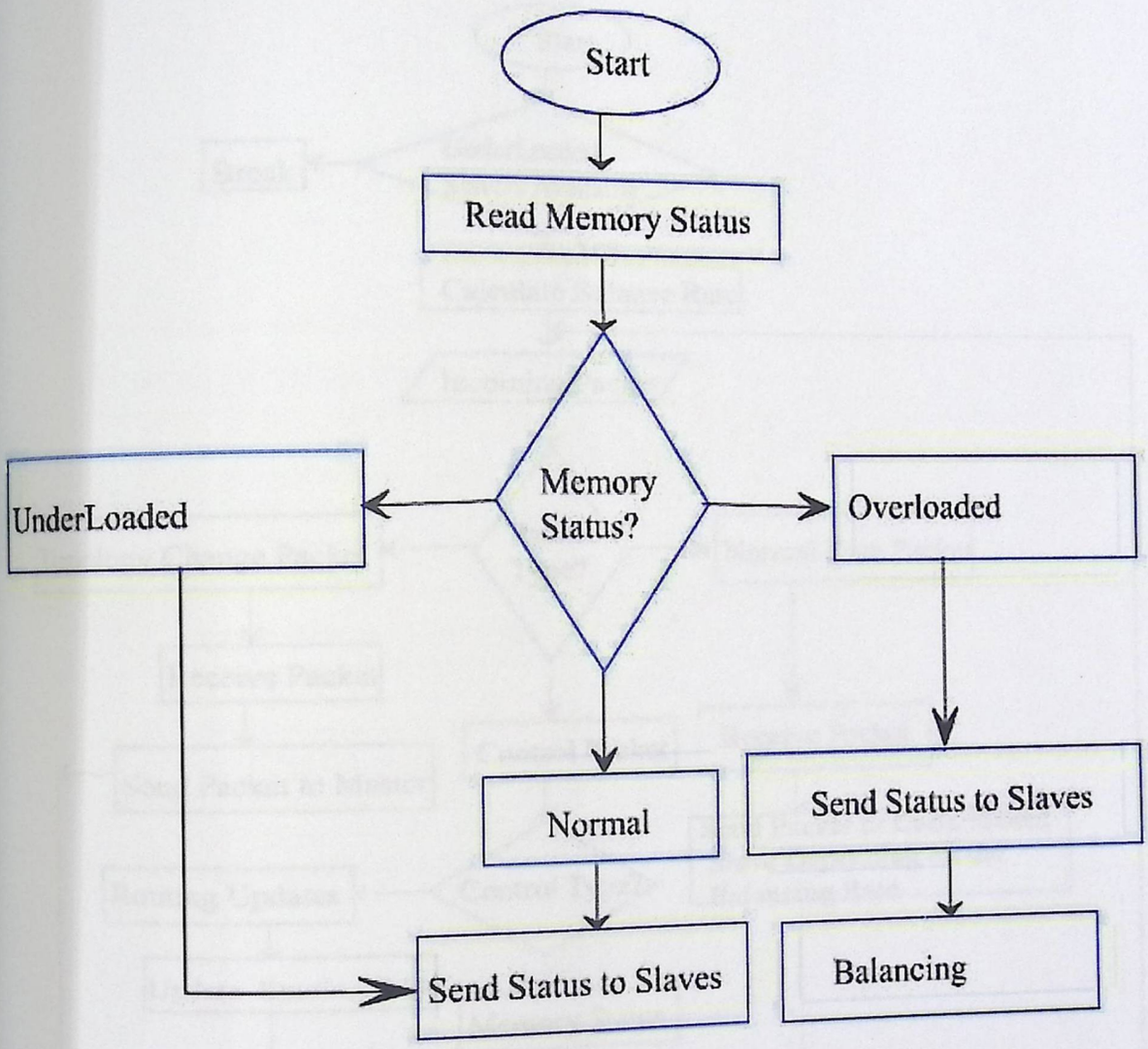


Figure (6.10) Check Memory Status Flowchart

6.1.2.6.5 Load Balancing

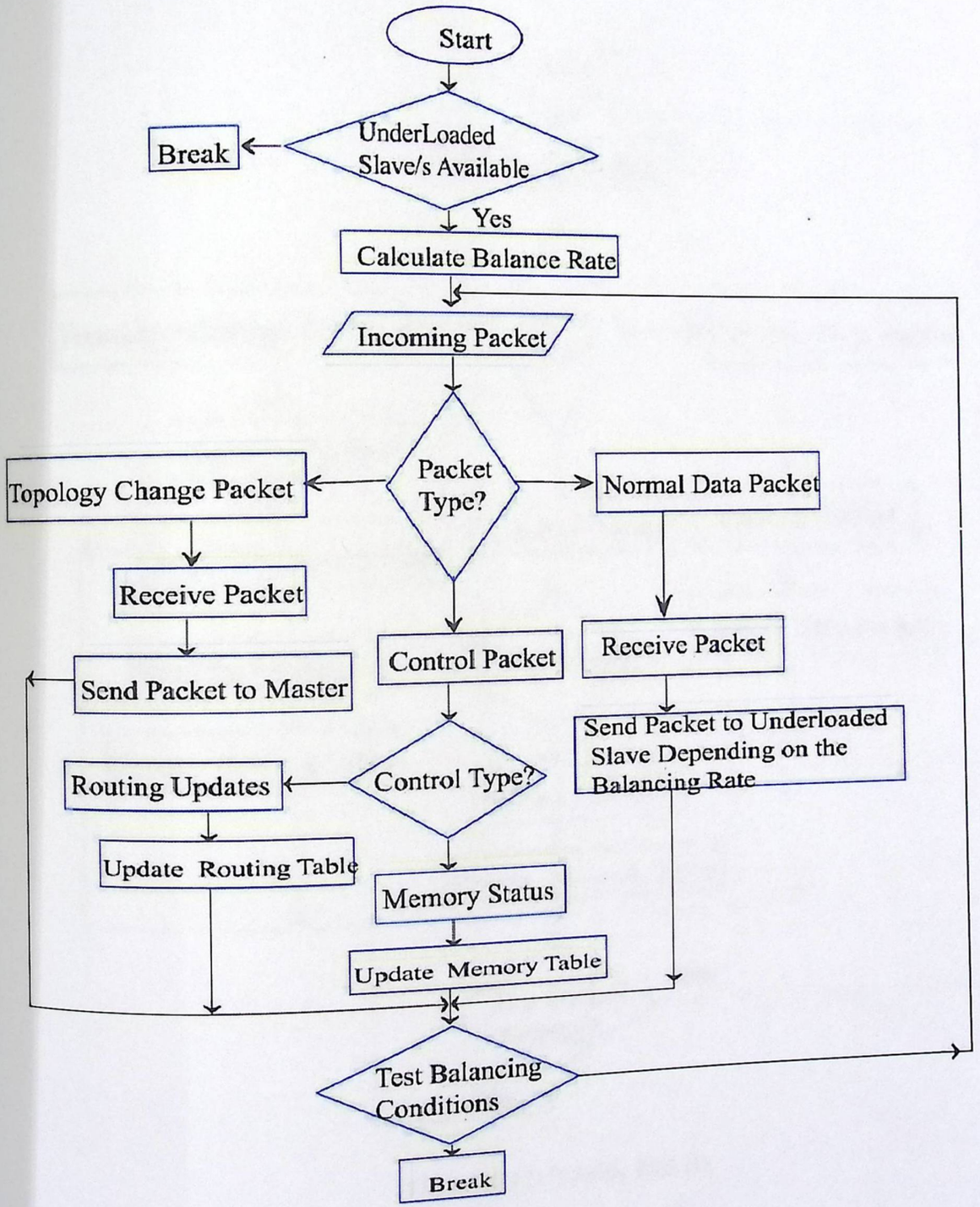


Figure (6.11) Load Balancing

6.1.2.6.6 Routing Packets

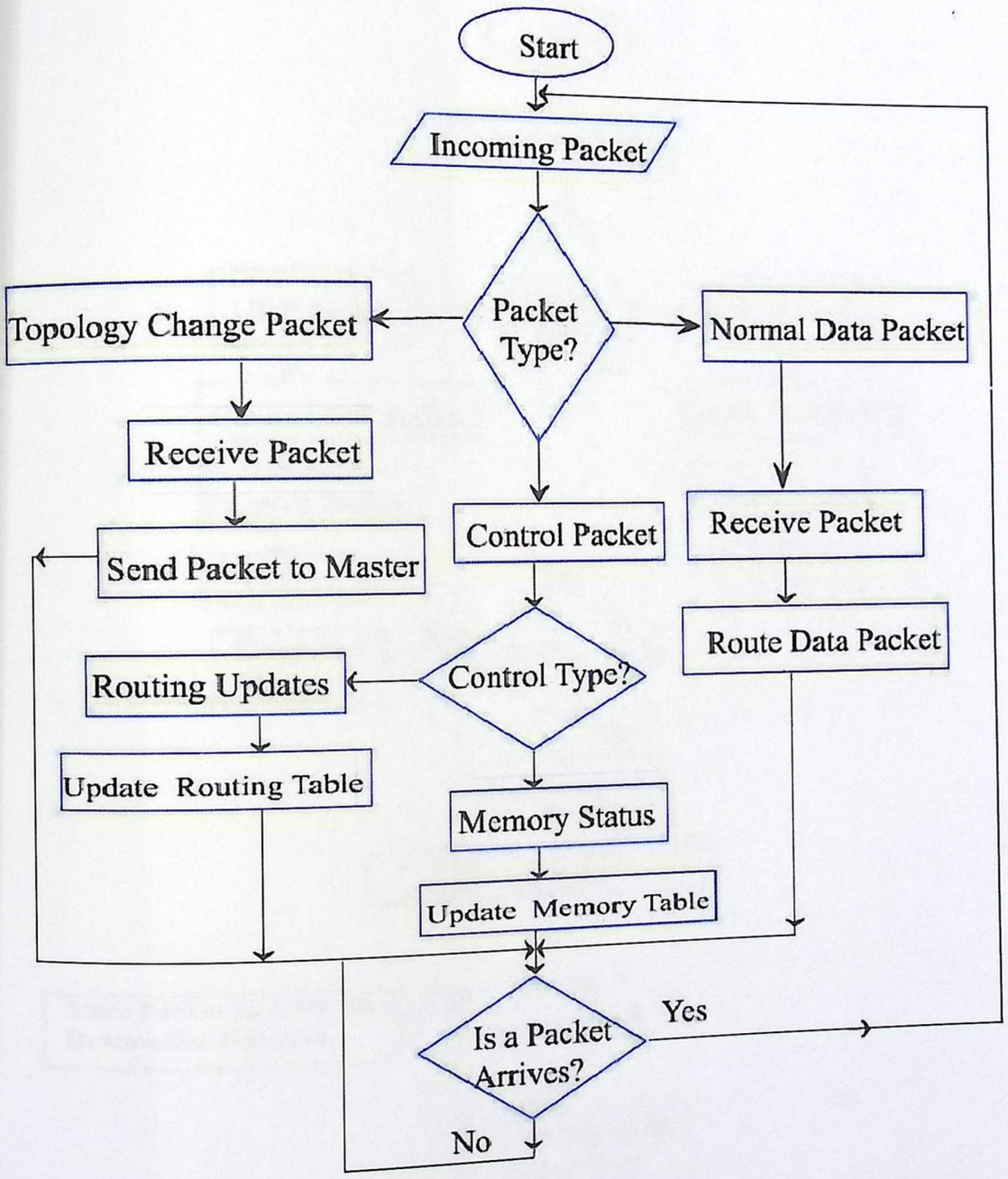


Figure (6.12) Routing Packets

6.1.2.6.7 Routing TCP and UDP Packets

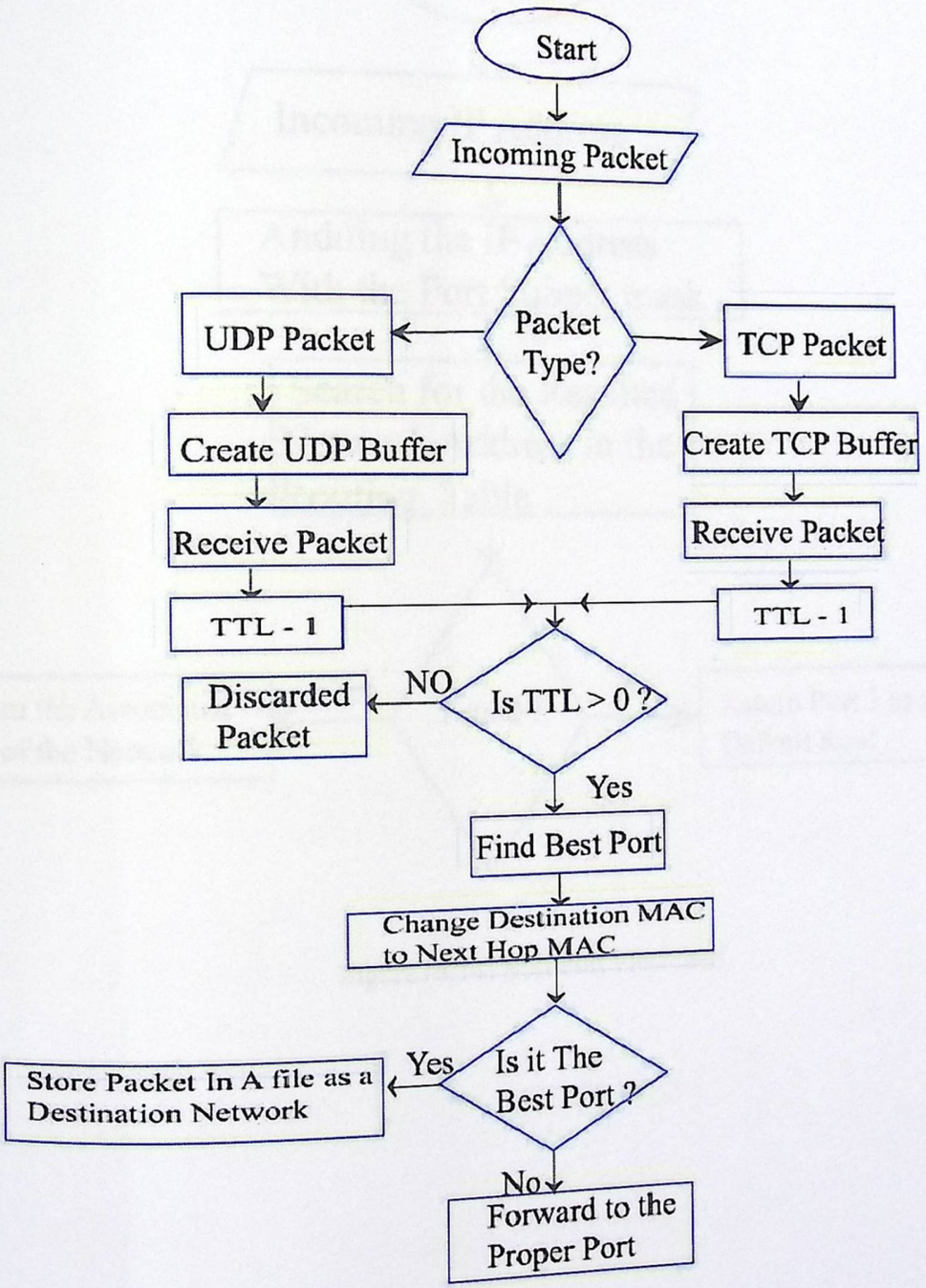


Figure (6.13) Routing UDP and TCP Packets Flowchart

6.1.2.6.8 Find Best Port

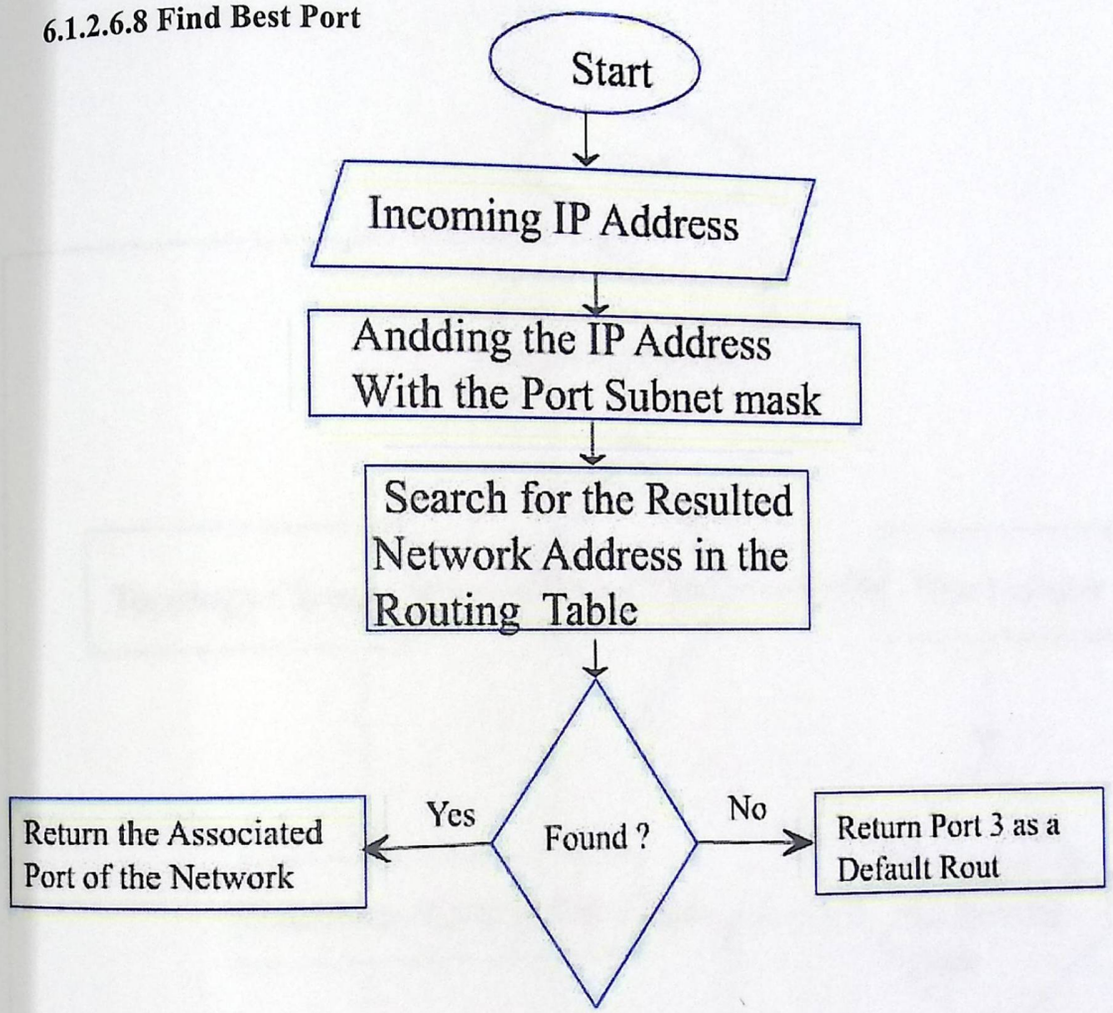


Figure (6.14) Best Port Flowchart

6.1.2.6.9 Master Works

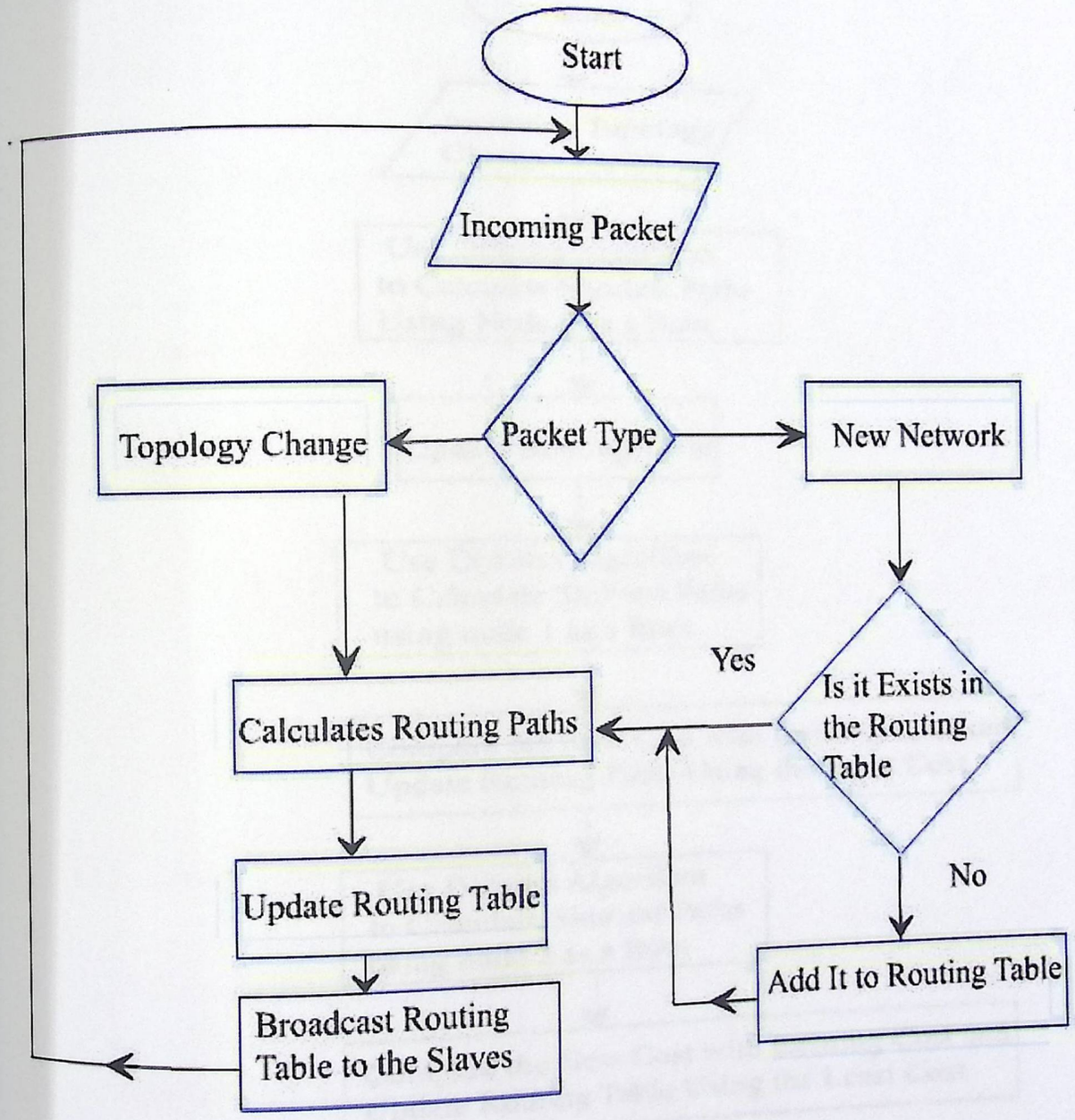


Figure (6.15) Master Work Flowchart

6.1.2.6.10 Calculate Routing Paths

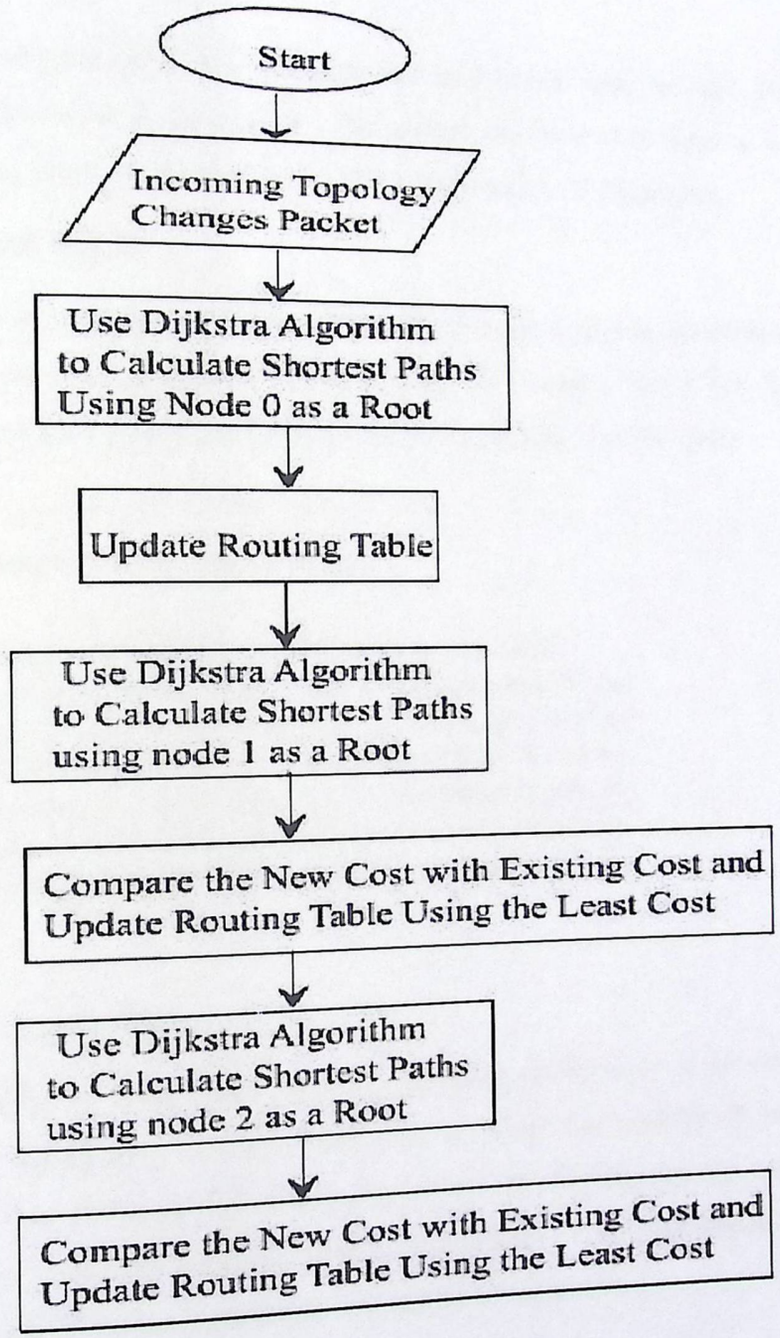


Figure (6.16) Calculating Routing Paths Flowchart

6.2 Testing

This Section demonstrates the methods and procedures used to test the functionality and effectiveness of the project. The project has more than issue to be tested. Our tests focus on the tests that are related to the functionality of the project.

6.2.1 Testing NIC's and TCP/IP

The network Interface card is an important part of our project, so it is important to test the functionality and configuration of TCP/IP of each NIC on all of the 5 PCs. To perform the test we open the terminal and write the following command on the shell:

```
[localuser@localhost ~]$ ping 127.0.0.1
```

The following result indicates the success of the test:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.070 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.067 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.068 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.068 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.068 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.069 ms
.
.
```

6.2.2 Testing Cables Connectivity

To test the connectivity of the cables we use a cable connectivity tester. Each cable must be tested before installing it. To test the cables we install the two male RJ-45 ends of the patch cable into the female RJ-45 of the cable tester, if all the LED's of the tester are lighted the cable has a good connectivity, if some of the LED's of the cable taster did not lighted then the cable has a bad connectivity and should be replaced.

6.2.3 Testing the Switch

After testing the cables and installing it between the PCs and the switch we connect the switch to the power, The power LED on the front panel of the switch should be lit, if it is not lit, then there are problems on the power or the switch, if the switch power LED lit, then the switch is working, so we check the Link/Activity LEDs, if the LED of each connected PC is lit or Flashing then the PC is well connected. If the LED is off then there is a problem of the link.

6.2.4 Testing Nodes Reachability

It is important for each node to reach every other node and to be reachable from all of the other nodes. To test nodes Reachability we write a ping command from each node to every node. In order to ping a node with IP (200 . 10. 10. 1) We should write the following command on the shell:

```
[localuser@localhost ~]$ ping 200. 10. 10 .1
```

The following reply indicates that the node with IP address 200. 10. 10 .1 is reachable:

```
PING 200. 10. 10 .1  (200. 10. 10 .1) 56(84) bytes of data.
64 bytes from 200. 10. 10 .1: icmp_seq=0 ttl=64 time=0.070 ms
64 bytes from 200. 10. 10 .1: icmp_seq=1 ttl=64 time=0.070 ms
64 bytes from 200. 10. 10 .1: icmp_seq=2 ttl=64 time=0.070 ms
64 bytes from 200. 10. 10 .1: icmp_seq=3 ttl=64 time=0.070 ms
64 bytes from 200. 10. 10 .1: icmp_seq=4 ttl=64 time=0.070 ms
.
.
```

6.2.5 Testing SSH Port

To run a parallel program using LAM/MPI SSH port must be opened To test if SSH port is open or not we write the following command on the shell:

```
[localuser@localhost ~]$ ssh localuser@fedora1
```

The following reply indicates that SSH port on host fedora1 using local user account is opened

```
Last login: Sun Jun 12 20:23:53 2005

[localuser@fedora1 ~]$
```

6.2.6 Testing LAM

To run the parallel program we must run the LAM first. so LAM must be installed and running. The following command test if LAM is installed and will configured:

```
[localuser@localhost ~]$ lamboot -v hosts
```

The following reply indicates that LAM is installed and running.

```
AM 7.0.6/MPI 2 C++/ROMIO - Indiana University

n-1<3716> ssi:boot:base:linear: booting n0 (localhost)
n-1<3716> ssi:boot:base:linear: finished

[localuser@localhost ~]$
```

6.2.7 Testing Routing

- **Routing**

  We test the function of the router by making a file to store the routed packets. After analyzing the files we conclude that the router is doing its work efficiently.

- **Load Balancing**

  We test the router by overloading one of the routing nodes and by tracing the routing process we note that the router perform load balancing effectively.

- **Test Scalability**

  We make a test using one PC, 2 PCs, 3 PCs, 4 PCs and 5 PCs, we note that the router work normally in all cases so our router is scalable.

- **Adapting Topology Changes**

  We test the router by making the packet generator sends a periodic topological changes and we notice that the router adapt the routing table depending on the topological changes

- **Routing Speed**

We run the simulation using one PC, 2 PCs, 3 PCs, 4 PCs and 5 PCs and notice that there is noticeable increase on the speed of routing when using more routing nodes. Figure (6.17), figure (6.18) and figure (6.19) show the results of three tests using different number of Packets:

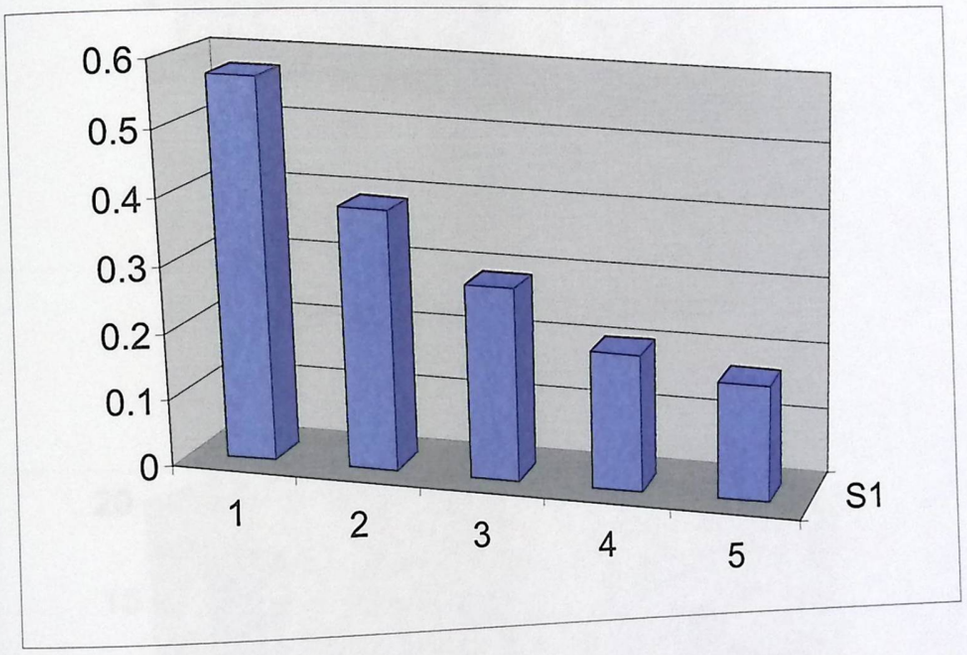


Figure (6.17) Speed using 300 Packets

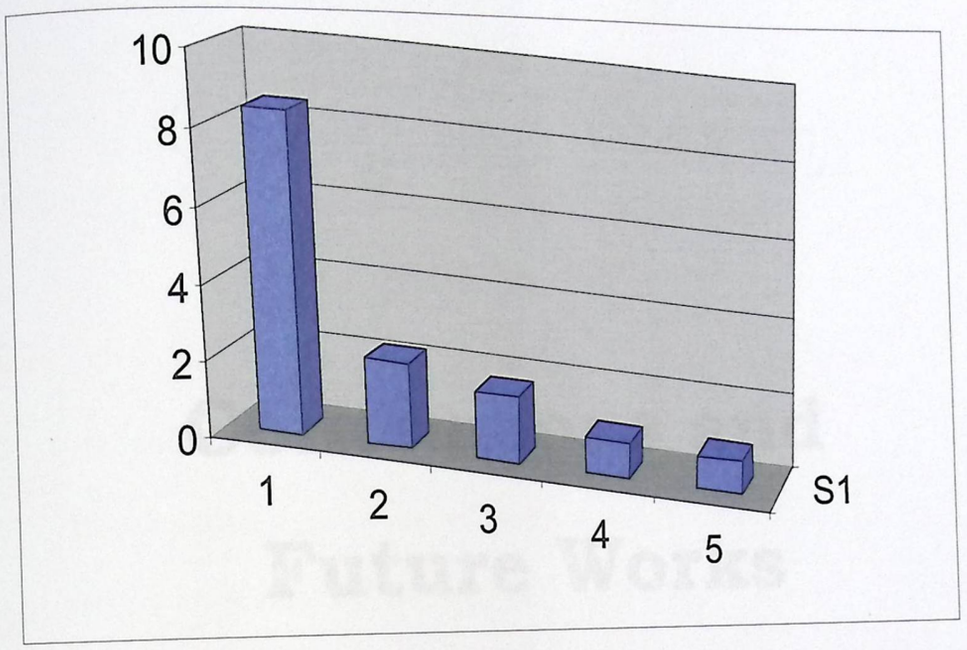


Figure (6.18) Speed using 3000 Packets

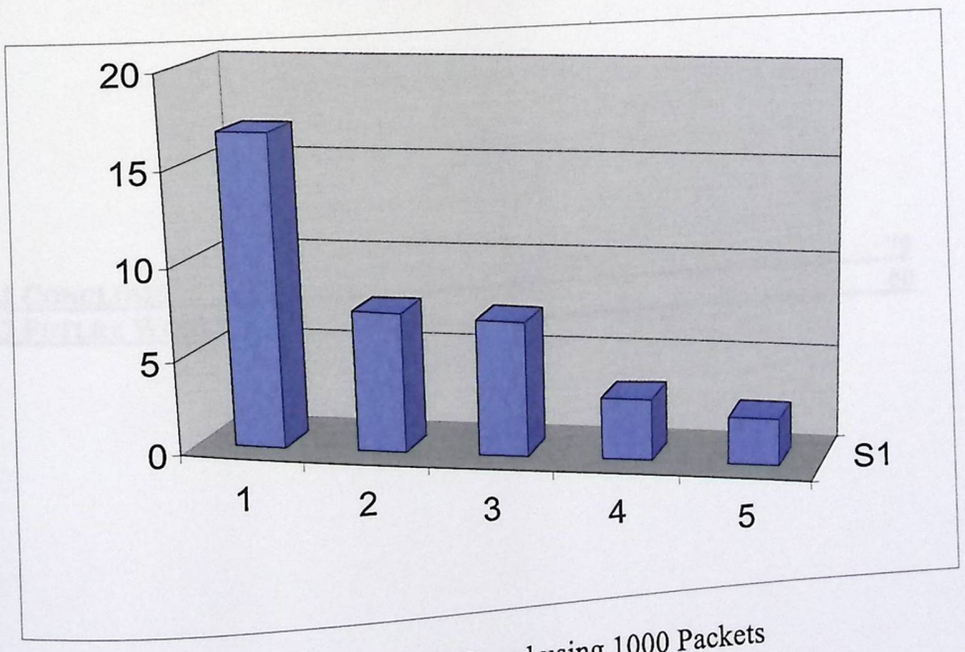


Figure (6.19) Speed using 1000 Packets

Chapter Seven

Conclusions and Future Works

Chapter Seven

Conclusions and Future Works

7.1 Conclusions

After completing this project we conclude the following:

- Parallel computing can be used to solve the routing problem.
- In this project the parallel router is scalable, since we can easily add new routing nodes to do routing.
- Clusters benefits depend on the availability of computer resources.
- Using clusters is cost effective
- Parallel computing can be used to solve some of the problem and not all of the problems because not all computing problems can be solved in parallel.
- The speed of using parallelism in routing is directly proportional the number of routing nodes.

7.2 Future Work

- This project is a simulation of parallel router in which we create the packets, a more real router can be build using a real packets.
- If we have the chance to continue our high studies we will try to implement this simulation to build a real parallel router that can be installed instead of the normal router.

References

References

1. http://www.llnl.gov/computing/tutorials/parallel_comp
2. Tamara Dean_ "Network+ Guide to Networks"_ Third Edition_ Course Technology_ Canada_2004
3. http://www.cisco.com
4. Sandra Loosemore _ "The GNU C Reference Manual"_ 0.09 Edition DRAFT_ Free Software Foundation_ Boston_1999
5. http://www.pluris.com
6. http://www.cse.msu.edu/~xiaoxipe/papers
7. www-unix.mcs.anl.gov/mpi
8. www.lam-mpi.org
9. www.mpi-forum.org
10. www.netlib.org/utk/papers/mpi-book/mpi-book.html
11. www.lam-mpi.org/download/files/7.0.6-user.pdf
12. www.lam-mpi.org/download/files/7.1.1-install.pdf
13. www.csit.fsu.edu/~burkardt/pdf/ibm/aix_mpi_ref.pdf
14. http://eies.njit.edu/~ziavras/VLSIdesign.pdf#search='parallel%20router
15. www.faqs.org/rfcs/rfc2328.html
16. www.ieee.org/
17. www.ietf.org/rfc.html
18. http://www.fdsnet.com/Routing/pdf/routing_guide.pdf
19. www.yolinux.com/TUTORIALS/LinuxTutorialNetworking.html

Appendices

Appendix A: MPI Subroutine Reference

Appendix B: Glossary

Appendix A

MPI SUBROUTINES REFERENCE

********** MPI_Send **********

NAME
MPI_Send - Performs a basic send

SYNOPSIS
#include <mpi.h>
int MPI_Send(void *buf, int count, MPI_Datatype dtype, int dest,
int tag, MPI_Comm comm)

INPUT PARAMETERS
buf - initial address of send buffer (choice)
count - number of elements in send buffer (nonnegative integer)
dtyp - datatype of each send buffer element (handle)
dest - rank of destination (integer)
tag - message tag (integer)
comm - communicator (handle)

NOTES
This function may block until the message is received. Whether or not MPI_Send blocks depends on factors such as how large the message is, how many messages are pending to the specific destination, whether LAMD or C2C communication is being used, etc.

********** MPI_ Recv **********

NAME
MPI_Recv - Basic receive

SYNOPSIS
#include <mpi.h>
int MPI_Recv(void *buf, int count, MPI_Datatype dtype,
int src, int tag, MPI_Comm comm, MPI_Status *stat)

INPUT PARAMETERS
count - maximum number of elements in receive buffer (integer)
dtype - datatype of each receive buffer element (handle)
src - rank of source (integer)
tag - message tag (integer)
comm - communicator (handle)

OUTPUT PARAMETERS
buf - initial address of receive buffer (choice)
stat - status object (Status), which can be the MPI constant MPI_STATUS_IGNORE if the return status is not desired

NOTES
The count argument indicates the maximum length of a message; the actual number can be determined with MPI_Get_count .

********** MPI_ Iprobe **********

NAME

MPI_Iprobe - Nonblocking test for a message

SYNOPSIS

```
#include <mpi.h>
int MPI_Iprobe(int src, int tag, MPI_Comm comm,
        int *flag, MPI_Status *stat)
```

INPUT PARAMETERS

src - source rank, or MPI_ANY_SOURCE (integer)

tag - tag value or MPI_ANY_TAG (integer)

comm - communicator (handle)

OUTPUT PARAMETER

flag - 1 if the message is ready to be received, 0 if it is not (logical)

stat - status object (Status), which may be the MPI constant MPI_STATUS_IGNORE

NOTES

This function does not actually receive a message; it only indicates that a message matching the signature specified is ready to be received.

********** MPI_ Init **********

NAME
MPI_Init - Initialize the MPI execution environment

SYNOPSIS
#include <mpi.h>
int MPI_Init(int *pargc, char ***pargv)

INPUT PARAMETERS
pargc - Pointer to the number of arguments
pargv - Pointer to the argument vector

NOTES
MPI specifies no command-line arguments but does allow an MPI implementation to make use of them. LAM/MPI neither uses nor adds any values to the argc and argv parameters. As such, it is legal to pass NULL for both argc and argv in LAM/MPI.

******************************** MPI_ Finalize ********************************

NAME

MPI_Finalize - Terminates MPI execution environment

SYNOPSIS

```
#include <mpi.h>
int
MPI_Finalize(void)
```

NOTES

All processes must call this routine before exiting. The number of processes running after this routine is called is undefined; it is best not to perform much more than a return rc after calling MPI_Finalize .

MPI mandates that the same thread invoke MPI_Init (or MPI_Init_thread) and MPI_Finalize ; if a different thread invokes MPI_Finalize , MPI_ERR_OTHER will be returned, and MPI will not be finalized.

********** MPI_Wtime **********

NAME

MPI_Wtime - Returns an elapsed time on the calling processor

SYNOPSIS

```
#include <mpi.h>
double MPI_Wtime(void)
```

RETURN VALUE

Time in seconds since an arbitrary time in the past.

NOTES

This is intended to be a high-resolution, elapsed (or wall) clock. See MPI_Wtick to determine the resolution of MPI_Wime . If the attribute MPI_WTIME_IS_GLOBAL is defined and true, then the value is synchronized across all processes in MPI_COMM_WORLD .

********** MPI_Comm_rank **********

NAME

MPI_Comm_rank - Determines the rank of the calling process in the communicator

SYNOPSIS

```
#include <mpi.h>
int MPI_Comm_rank(MPI_Comm comm, int *rank)
```

INPUT PARAMETERS

comm - communicator (handle)

OUTPUT PARAMETER

rank - rank of the calling process in group of comm (integer)

****************************** MPI_ Type_extent ******************************

NAME
MPI_Type_extent - Returns the extent of a datatype

SYNOPSIS
#include <mpi.h>
int MPI_Type_extent(MPI_Datatype dtype, MPI_Aint *pextent)

INPUT PARAMETERS
dtype - datatype (handle)

OUTPUT PARAMETER
pextent
- datatype extent (integer)

NOTES
This function is deprecated. It has been replaced with an MPI-2 function (see the "See also" section, below), which provides the same functionality. This function is (or effectively is) a wrapper to the replacement function, anyway. User programs should use the MPI-2 replacement function instead of this function.

****************************** MPI_Type_struct ******************************

NAME

MPI_Type_struct - Creates a struct datatype

SYNOPSIS

```
#include <mpi.h>
int MPI_Type_struct(int count, int *lengths,
            MPI_Aint *disps, MPI_Datatype *oldtypes,
            MPI_Datatype *newtype)
```

INPUT PARAMETERS

count - number of blocks (integer) -- also number of entries in arrays array_of_types , array_of_displacements and array_of_block-lengths

blocklens - number of elements in each block (array)

indices - byte displacement of each block (array)

old_types - type of elements in each block (array of handles to datatype objects)

OUTPUT PARAMETER

newtype - new datatype (handle)

NOTES

This function is deprecated. It has been replaced with an MPI-2 function (see the "See also" section, below), which provides the same functionality. This function is (or effectively is) a wrapper to the replacement function, anyway. User programs should use the MPI-2 replacement function instead of this function.

****************************** MPI_Type_commit ******************************

NAME
MPI_Type_commit - Commits the datatype

SYNOPSIS
#include <mpi.h>
int MPI_Type_commit(MPI_Datatype *dtype)

INPUT PARAMETER
dtype - datatype (handle)

Appendix B

GLOSSRY

Glossary

ARP	Address resolution Protocol
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
IP	Internet Protocol
ISO	International Organization for Standardization
LAN	Local Aria Network
LSA	Link State Advertisement
MAC	Media Accesses Control
MPI	Message Passing Interface
MPR	Massively Parallel Router
NIC	Network Interface Card
RIP	Routing Information Protocol
OSI	Open System Interconnection
OSPF	Open Shortest Path First
PCMCIA	Personal Computer Memory Card International Association
RARP	Reverse Address Resolution Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WAN	Wide Aria Network