SIMULATION OF RESILIENT NETWORKS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Computer Science and Engineering

By

MRUNMOY PANIGRAHI

ARAVINDA JENAMANI

B. MURLI KRISHNA

Department of Computer Science and Engineering

National Institute of Technology

Rourkela

2007

SIMULATION OF RESILIENT NETWORKS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Computer Science and Engineering

By

MRUNMOY PANIGRAHI

ARAVINDA JENAMANI

B. MURLI KRISHNA

Under the Guidance of

Dr. A.K. TURUK

Department of Computer Science and Engineering

National Institute of Technology

Rourkela

2007

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis entitled **"Simulation of Resilient Networks''** Submitted by **Mrunmoy Panigrahi, Roll No:10306035, Aravinda Jenamani**, **Roll No: 10406034D** and **B. Murli Krishna, Roll No. 10406033D** in the partial fulfillment of the requirement for the degree of **Bachelor of Technology** in **Computer Science Engineering**, National Institute of Technology, Rourkela , is being carried out under my supervision.

To the best of my knowledge the matter embodied in the thesis has not been submitted to any other university/institute for the award of any degree or diploma.

Dr. A.K. Turuk

Date:

Department of Computer
Science and Engineering
National Institute of Technology
Rourkela-769008

Acknowledgment

We avail this opportunity to extend our hearty indebtedness to our guide **Dr. A.K. Turuk**, Computer Science Engineering Department, for their valuable guidance, constant encouragement and kind help at different stages for the execution of this dissertation work.

We also express our sincere gratitude to **Dr. S.K.JENA**, Head of the Department, Computer Science Engineering, for providing valuable departmental facilities.

Submitted by:

Mrunmoy Panigrahi
Roll No: 10306035
Computer Science Engineering
National Institute of Technology
Rourkela

Aravinda Jenamani
Roll No: 10406034D
Computer Science Engineering
National Institute of Technology
Rourkela

B. Murli Krishna
Roll No: 10406033D
Computer Science Engineering
National Institute of Technology
Rourkela

CONTENTS

ABSTRACT

Resilient network is a network, which does not fail under any circumstances. It is the ability of the network to provide and maintain an acceptable level of service in the face of various challenges to normal operation. Disjoint routing strategies are used to calculate disjoint paths, which can be used by the network to route the packet in case of some failure to the existing primary path.

Packets arrive at a node in the network in a very random manner. The probability density function for describing the number of such arrivals during a specific period follows the Poisson distribution. The inter-arrival time and the service time at a node follow exponential distribution. As a packet travels from one node to the subsequent node along its path, the packet suffers from different types of delays at each node along the path. Thus the total node delay can be calculated by summing up all the different types of delays. With the increase in traffic and therefore the congestion, the average cost of transmission and retransmission (in the event of a failure) suffers. This varies according to the strategy used for resilience.

Routing protocols inherently provide a basic level of resiliency. The ability to “route” around the problem areas defines the efficiency of a routing protocol. However, the time to re-converge the network can vary greatly depending on the protocol being used. When a failure occurs, either the packets can be resent or a new path can be followed from the failure point.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

i)Resilience

ii)Need for resilience

iii)Types of failures

1.1 INTRODUCTION

1.2 Resilience :

The word “resilience” means the ability to adapt well to stress. It means that, overall you remain stable and maintain healthy levels of physical functioning in the face of disruption or chaos.

A resilient network is a network, which does not fail under any circumstances. Failure refers to a situation where the observed behaviour of a system differs from its specified behaviour. A failure occurs due to an error, caused by a fault. Faults can be hard or soft. For example a cable break is a hard failure whereas an intermittent noise in the network is a soft failure.

Resilience in the context of resilient network is the ability of the network, a device on the network, or a path on the network to respond to failure, resist failure, handle flux in demand and easily shift and configure – with little or no impact on service delivery. A resilient network is the agent that can help to diminish the loss of employee productivity in the event of a major disaster.

Now we discuss the importance of Resilience in industries.

1.3 Need for Resilient Network:

Businesses in all the industries are becoming dependent on Information Technology (IT) and the intra- and inter- organizational online communication and collaboration it enables. Digitization and workforce mobilization, automation and embedded computing have changed the way enterprises do business and interact with their customers, employees and business partners. The requirements for business infrastructure have also changed. Business infrastructure must provide a stable IT foundation for the internal rganization as well as allow integration with a virtual value chain of suppliers and customers. To effectively support the needs of today’s businesses, business infrastructure must, in effect, be

RESILIENT. Resilient implies flexible and adaptive yet at the same time fortified against all types of threats. Resilient network design is the key component of Resilience.

Resilient networks incorporate many of the elements of a highly available network. The resilient network architecture should include redundant (multiple) components that can take over the function of one another if one should fail. How the network, device or path reacts to failure should be determined before hand so that predictable network, device or paths are present after response to failure.

1.4 Types of Failures :

Single point failure: It indicates that a system or a network can be rendered inoperable, or significantly impaired in operation, by the failure of one single component. For example, a single hard disk failure could bring down a server; a single router failure could break all connectivity for a network.

Multiple points of failure: It indicates that a system or a network can be rendered inoperable through a chain or combination of failures. For example, failure of a single router plus failure of a backup modem link could mean that all the connectivity is lost for a network. In general it is much more expensive to cope with multiple points of failure and often financially impractical.

Disaster recovery is the process of identifying all potential failures, their impact on the network as a whole, and planning the means to recover from such failures.

In our project we have implemented two types of failures:

- *Link failure:* In case of link failure if one link between two nodes fails then only that link gets failed. It won't affect any other nodes in the network.
- *Node failure :* In case of node failure if any node fails, then all the links connected to it also fail

CHAPTER 2

LITERATURE SURVEY

i)Disjoint path strategy

ii)Implementation of the strategy

2. LITERATURE SURVEY

2.1 Disjoint Path Strategy

Networks are expected to meet a growing volume of requirements imposed by new applications such as multimedia streaming and video conferencing. Two essential requirements are support of Quality of Service (QoS) and resilience to failures. In order to satisfy these requirements, a common approach is to use two disjoint paths between the source and the destination nodes, the first serving as a primary path and the second as a restoration path. Such an approach, referred to as path restoration, has several advantages, the major one being the ability to switch promptly from one path to another in the event of a failure. The disjoint path strategy has many additional advantages. First, it allows using various protection schemes, such as 1+1 protection or 1:1 protection. With 1+1 protection, traffic is simultaneously transmitted on both paths, which allows instantaneous recovery from link failures. Alternatively, with 1:1 protection, traffic is transmitted along a primary path, and upon failure of one of its links, the traffic is switched to a restoration path. Second, the disjoint path strategy requires minimal network support, because failure detection and restoration can be implemented at the application level of the source. Finally, the disjoint path strategy provides a greater flexibility to application designers, as they can choose a protection scheme that is most adequate for a particular application. QoS constraints can be divided into bottleneck constraints, such as bandwidth, and additive constraints, such as delay or jitter.

There are 2 different types of disjoint path strategies:

- Node-disjoint path strategy
- Edge-disjoint path strategy

2.1.1 Node disjoint path strategy

Node disjoint path problems have attracted much attention in both mathematical terms and interconnection network studies due to its numerous applications in fault tolerant routing and so on. In what follows, we will use disjoint path for node disjoint path. Node-disjoint path strategy uses the network, which is left after the removal of the nodes along with all the associated links, present in the primary path between a source destination pair, to compute the restoration path. The minimum energy k node disjoint S-D paths problem can be stated as follows : Given an energy cost graph G = (V,E) with weights **W**ij and source destination pair S,D belonging to V, find a set of k-node disjoint S-D paths, P = p1,p2, … pk, such that sum Energy(P) is minimized. The set of paths P is shown in fig. 1.1 below with nodes colored gray being in both the paths.

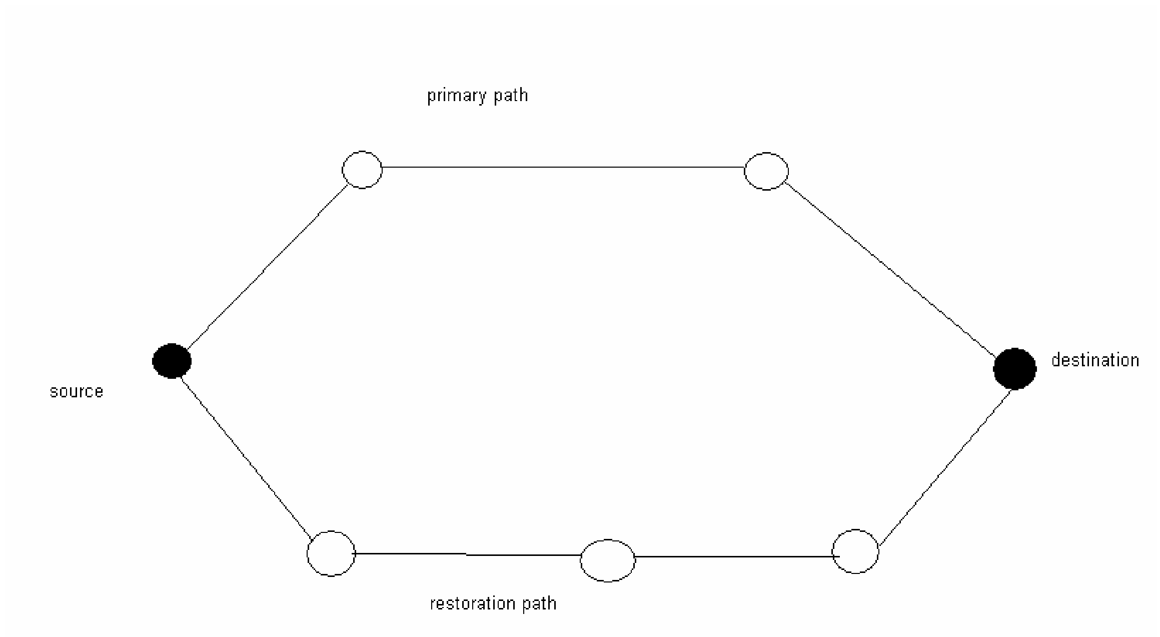

Figure 2.1 : Node disjoint paths

2.1.2 Edge disjoint path strategy

According to [C. Chekuri, A. Gupta, A. Kumar, J. Naor, D. Riaz] the model of edge-failures is adversial. They make the assumption that there is some fixed value k such that only k edge failures can happen in the network at any given instant of time. This should be contrasted with the probabilistic models, where edges are allowed to fail with some specified probabilities. This assumption is commonly used in practice and seems to work reasonably well. Another pragmatic reason for this assumption is that most networks are k-connected for some small k, and hence they cannot tolerate more than k adversial edge failures. Furthermore, it is interesting to note that the resulting optimization problems are already hard for the case of k=1. In the following discussion, we have restricted our attention to the single edge failure case of k=1; where appropriate, we will indicate how the ideas for k=1 extend to general k. Note that for k=2. both primary as well as backup edges are allowed to fail. Resilience against single edge failures can be built into the network by providing for each edge e , a backup path P(e), which is used when the edge e fails. However, since on ly one edge is guaranteed to fail, making the backup paths for 2 different edges intersect each other and share the same amount of bandwidth results in backup networks of lower cost. This multiplexing is one of the factors that make this problem especially difficult.

We consider the design of resilient networks that are fault tolerant against link failures. Resilient against link failures can be built into the network by providing backup paths, which are used in the eventuality of an edge failure occurring on a primary path in the network.

Edge disjoint path strategy uses the network, which is left after the removal of the edges present in the primary path between a source-destination pair, to compute the restoration path. The minimum energy k link disjoint S-D paths problem can be stated as follows : Given an Energy cost graph G = (V,E) with weights Wij and source destination pair S,D belonging to V, find a set of link disjoint S-D paths, P = p1,p2,...pk, such that sum Energy (P) is minimized. The set of paths P is shown in fig 2.2 below with nodes colored gray being common to both the paths

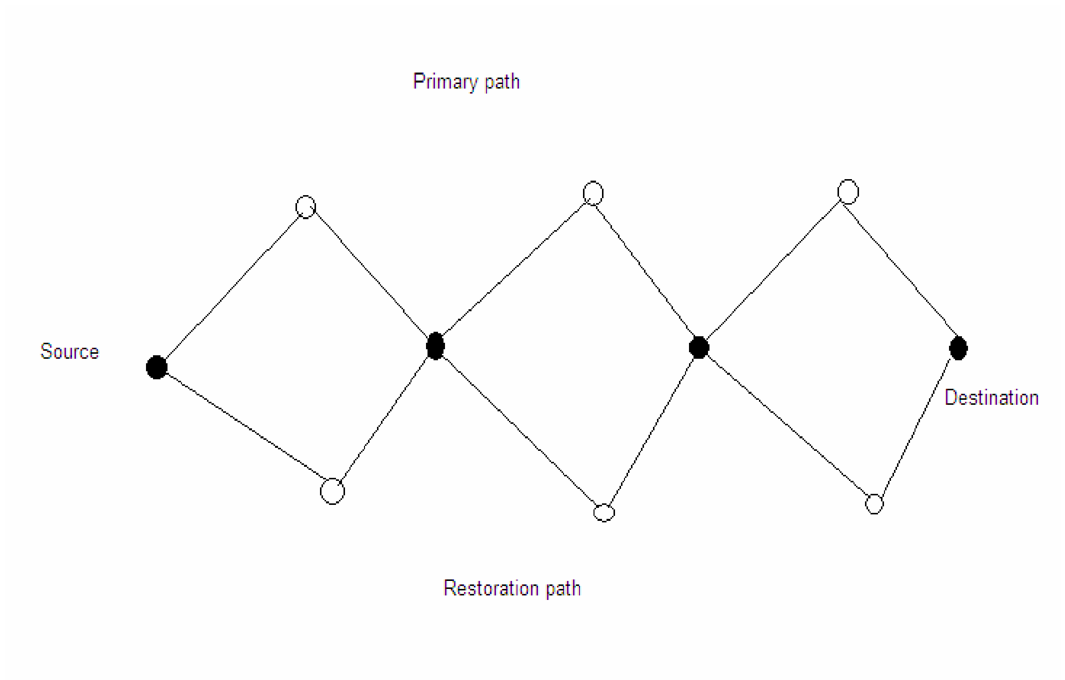

Figure2.2: Edge disjoint paths

2.2 Implementation of the Strategy

Dijkstra's algorithm solves the problem of finding the shortest path from a point in a graph (the source) to a destination. It turns out that one can find the shortest paths from a given source to all points in a graph in the same time; hence this problem is sometimes called the single-source shortest path problem. For a graph G = (V,E) where V is a set of vertices and E is a set of edges, Dijkstra's algorithm keeps 2 sets of vertices : S the set of vertices whose shortest paths from the source have already been determined and V-S the remaining vertices. While there are still vertices in V-S, sort the vertices in V-S according to the current best estimate of their distance form the source. Add u, the closest vertex in V-S to S. Relax all the vertices still in V-S connected to u relaxation. The relaxation process updates the costs of all the vertices v, connected to a vertex, u, if we could improve the best estimate of the shortest path to v by including (u,v) in the path to v.
Dijkstra's algorithm is given in detail below:

```
TableEntry = record
Header : list;
```

```
Known : Boolean;
Dist : DistType;
Path :Vertex;
end;

Table =array[Vertex] of TableEntry;

/***************************************************************
********************/



procedure InitiTable (Start :Vertx ; var G: Graph; var T: Table);
va i: integer;
begin
ReadGraph(g,t); {read graph somehow.}
fori:=1 to NumVertx do begin
T[i].Know := false;
T[i].Dist := MAXINT;
T[i].Path :=0;
end;
T[Start].Dist:=0;
end;

/***************************************************************
********************/
{ print shortest path to V after Dijkstra has run..}
{ Assume that the path exists}

procedure Printpath( V: Vertex; var T :Table);
begin
if T[V].path<> 0 then
begin
```

```
PrintPath( T[V].path ,T);
write (‘to’);
end;
write(V);
end;
/****************************************************************
********************/

procedure Dijkstra(var T: Table);
var
i :integer;
V,W :Vertex;
begin
for i:=1 to NumVertex do begin
V:= Smallest Unknown Distance Vertex;
T[V].known:=true;
for Each W Adjacent to V do
if not T[W].known then
begin
if T[V].Dist +C(V,W)< T[w].Dist then
begin { Update W.}
Decrease T[W].Dist tp
T[V].Dist +C(V,W);
T[W].path:=V;
end;
end;
end;
end;
```

After finding the shortest path, the next edge disjoint shortest path is computed by removing the links from the network that are present in the primary shortest path. Similarly for node disjoint shortest path we remove the nodes present in the shortest primary path. Removal of nodes results in the removal of the associated links. This implements the node and edge disjoint path strategy successfully.

CHAPTER 3

PROPOSED SCHEMES

i)Source retransmission strategy

ii)Intermediate node forwarding

3. PROPOSED SCHEMES

In this project we proposed 2 types of schemes to reduce packet loss due to faults in the network. They are

1. Source retransmission

2. Intermediate node forwarding

Faults were introduced in the network and recovery was initiated. Two recovery schemes were implemented namely:

3.1 Source retransmission strategy

In this method when fault is encountered at a node, a fault packet is sent back to the source node. The source node then finds a different optimal path to retransmit the packet. We illustrate this strategy with a 14 node NSFNET as shown in fig 3.1. Let node A be the source node and node J the destination node. Then the shortest path between the source and destination is A – D – E – G – H – J. And the second shortest path between source and destination is A – D – I – M - . Suppose there is a fault between node G and node H. In this strategy node G will generate a fault packet and send it to the source node A. The source on receiving the fault packet will select a new path. The retransmission path is A – D – I – M – J.

Since retransmission is done, this method requires a comparatively more recovery time because the fault packet has to be sent to the source from the intermediate node where the fault is detected. Moreover, during the recovery period, the source will transmit more number of packets to the destination node where some packets may be lost at the intermediate node due to buffer overflow.

This can be diagrammatically viewed as :

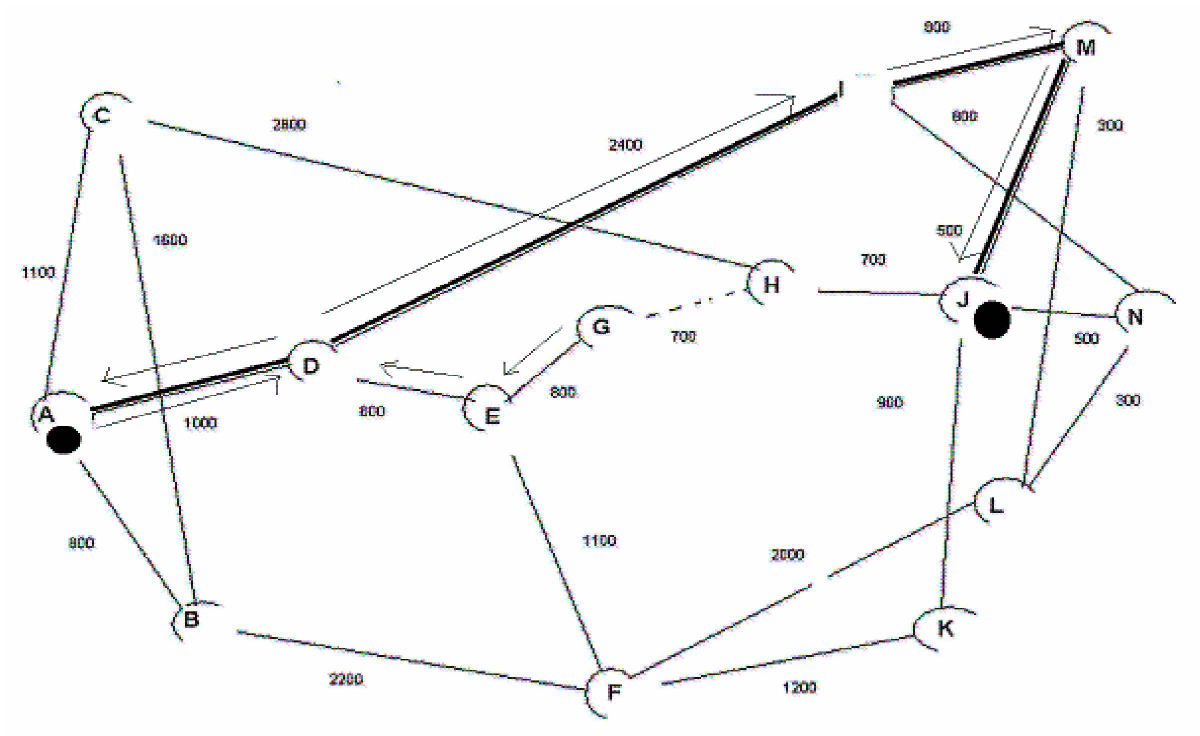

Fig 3.1 Source retransmission

Source Retransmission algo:-

1. Find shortest path from source to destination using Dijkstra's algorithm.

2. If failure occurs then

   a) Sent control signal to source node.

   b) Reconfigure the network

3. Go to step 1.

3.2 Intermediate node forwarding

In the source retransmission strategy, packets may be lost due to buffer overflow. Packets may experience an additional delay of the recovery time.

Next we propose a scheme, where the intermediate node that detects the fault will find a path to the destination. We consider the 14 node NSFNET as shown in fig. 3.2. Let node A be the source and node J the destination. Suppose there is a fault at G-H link. So according to this strategy node G becomes the new source and the destination remains the same. And from node G to node J, an optimal path is calculated and the packet is sent through that path. The optimal path from node G to node J is G – E – F – K – J. So the packet travels A – D – E – G – E – F – K – J. Recovery time is less in comparison to the former strategy because no fault packet/control signal needs to be sent back to the source.

This can be diagrammatically viewed as:

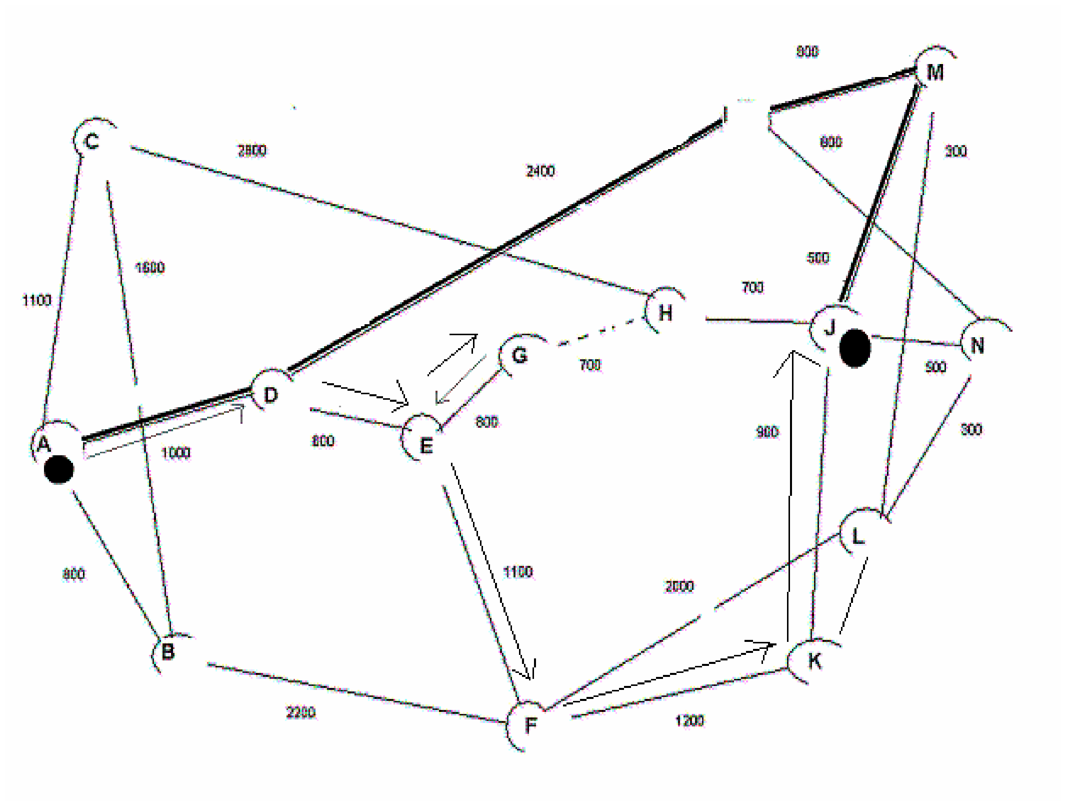

Fig 3.2 Intermediate node forwarding

Intermediate node forwarding algo:-

```
1. Find shortest path from source to destination using Dijkstra’s algorithm.
2. If failure occurs then
    a) Go back to the previous router.
    b) Reconfigure the network
    c) Set current node = new source
3. Go to step 1.
```

CHAPTER 4

SIMULATIONS AND RESULTS

i)Packet generation
ii)Random graph generation
iii)NSFNET
iv)Delay calculation
v)Results

4. Simulations and Results

This section is divided into 2 subsections A and B. In section A we give :

i) Packet generation

ii)Random graph and fault generation

iii)NSFNET

And in section B we give the results of the simulation and delay calculation such as :

i) Processing delay

ii) Queuing delay

iii) Transmission delay

iv)Propagation delay

Section A

4.1 Packet Generation

Packets arrive at a node in a network in a totally random fashion, that is, there is no way to predict when the packets will arrive next. The probability density function for describing the number of such arrivals during a specified period follows Poisson Distribution. Poisson process is an arrival process in which the interarrival time and the processing time at the node are exponential random variables. If the Poisson process has an arrival rate of r, then its corresponding inter-arrival time A1,A2 are exponential random variables with mean (1/r). Poisson Queuing model assumes that both arrival and departure rates are state-dependent, meaning that they depend on the number of packets at the nodes.

Exponential Distribution can be expressed by the formula -

$X = (-1/r)\ln(1-R)$

where R is the uniformly distributed random number between 0 and 1. Let y be the number of packet arrivals that take place during a specified time unit.

The Poisson probability density function is given by :

P(y=k) = exp(r,k) . exp(e,-r)/ k!

where k = 0,1,2 ...

Mean E(y) = r

Variance Var(y) = r

Congestion of packets results at a node because the packets cannot be serviced immediately on arrival and each new arrival has to wait for some time before it gets serviced. A reasonably long waiting queue may result in the loss of packets. Thus the size of the queue should be optimum to reduce the congestion as far as possible.

4.2 Random Graph Generation

A random graph is generated for a network with fixed number of vertices and fixed number of edges by randomly generating the edges connecting the vertices. Thus each time this graph is simulated we get a different network.

In our project initially, we have generated a random by generating the vertices randomly and evening them out on the computer screen. We did this using a linked-list data structure

However we obtained all our desired results from a standard NSFNET 14 node network.

4.3 NSFNET

In the beginning there was ARPA net, a wide area experimental network connecting hosts and terminal servers together. Procedures were set up to regulate the allocation of addresses and to create voluntary standards for the network. As local area networks became more pervasive, many hosts became gateways to local networks. A network layer to allow the interoperation of these networks was developed and called IP (Internet Protocol). Over time other groups created long haul IP based networks (NASA,NSF,states ...). These nets, too, interoperate because of IP. The collection of all of these interoperating networks is the Internet. Cornell University temporarily operates NSFNET (called the National Science Foundation network).NSFNET is a high speed network of networks which is hierarchical in nature. The network can be represented as :

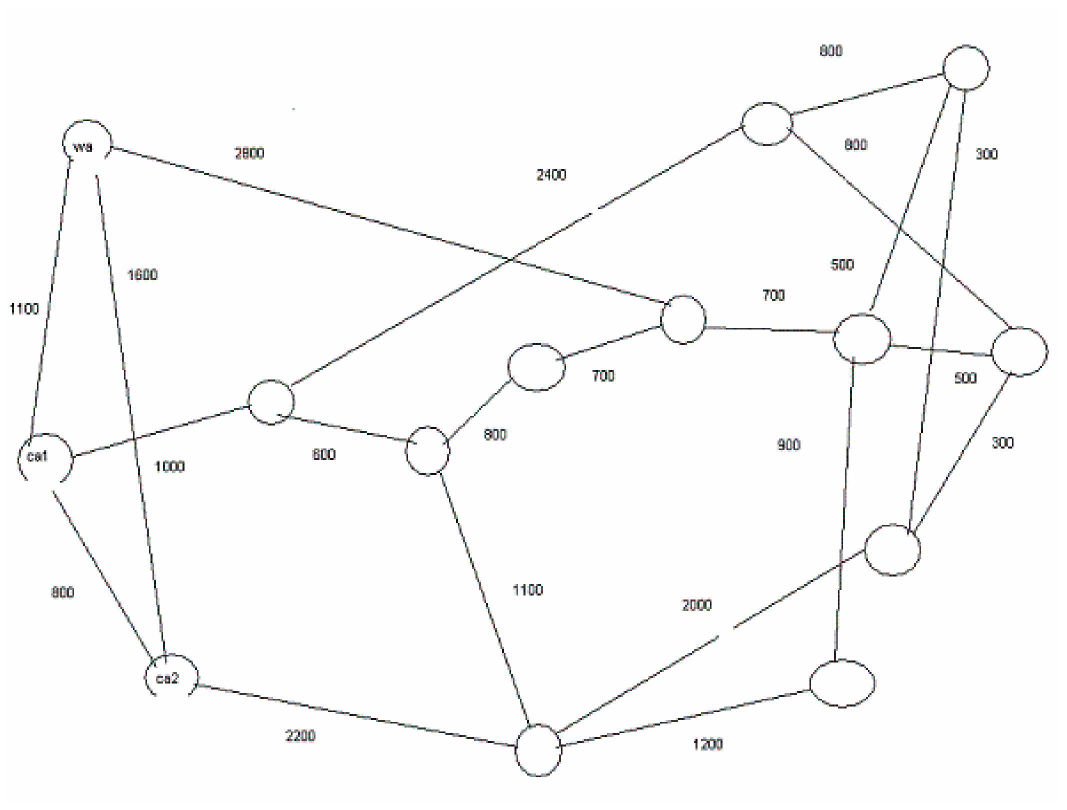

Fig 4.1 : NSFNET

In our project, we have taken NSFNET as the primary network and have done all the simulations on this network.

Section B

4.4 Delay Calculation

A packet starts in a host (the source), passes through a series of routers, and ends its journey at another host (the destination). As a packet travels from one node to the subsequent node along its path, the packet suffers from different types of delays at each node along the path. These delays include nodal processing delay, queuing delay, transmission delay and propagation delay; together these delays accumulate to give a total nodal delay.

4.4.1 Processing Delay

The time required to examine the packet's header and determine where to direct the packet is part of the processing delay. The processing delay can also include other factors, such as the time needed to check for bit-level errors in the packet that occurred in transmitting the packet's bits. After this nodal processing, the router directs the packet to the queue that precedes the link to the next router.

4.4.2 Queuing Delay

At the queue, the packet experiences a queuing delay as it waits to be transmitted onto the link. The queuing delay of a specific packet will depend on the number of earlier-arriving packets that are queued and waiting for transmission across the link. The delay of a given packet can vary significantly from packet to packet. If the queue is empty and no other packet is currently being transmitted, then our packet's queuing delay is zero. On the other hand, if the traffic is heavy and many other packets are also waiting to be transmitted, the queuing delay will be long.

4.4.3 Transmission Delay

As the packets are transmitted in the first come first served manner, a packet is transmitted only after all the packets that have arrived before it have been transmitted. Denote the length of the packets by L bits and denote the transmission rate of the link by R bits sec. The transmission delay (also called the store and forward delay) is L/R. This is the amount of time required to push (transmit) all of the packet's bits into the link. This delay is a function of the packet's length and the transmission rate of the link.

4.4.4 Propagation Delay

Once a bit is pushed onto the link, it needs to propagate to the router. The time required to propagate from the beginning of the link to the router is the propagation delay. The bit propagates at the propagation speed of the link. The propagation speed depends on the physical medium of the link and is of the degree of 2*10^8 to 3*10^8 metres/sec. The propagation delay is the distance between 2 routers divided by the propagation speed. This delay is a function of the distance between the 2 routers.

Thus the total nodal delay is given by,

d(nodal) = d(proc) + d(queue) + d(trans) + d(prop)

When a router receives a packet, it sends a short message to the source. Similarly, when the destination host receives the packet, it returns a message back to the source. The source records the time elapsed from when it sent a packet till it receives the corresponding message. In this manner, the source can determine the round trip delays to all the intervening routers.

4.5 Results

We have calculated the delays in three cases:

- In an NSFNET with no faults
- By source retransmission
- By intermediate node forwarding

4.5.1 No fault

If there is no fault in the network, the packet travels in the path A - D - E - G - H - J to the destination node J with cost 4000

4.5.2 Source retransmission

If there is a fault in the network and in case of source retransmission, cost to travel back to the source node is 2600. And the cost to travel to the destination in the second shortest path is 4700. So the total cost is 2600 + 4700 = 7300. This path is A - D - I - M - J.

4.5.3 Intermediate node forwarding

If there is a fault in the network and in case of intermediate node forwarding, the optimal path from the node G (where the fault is detected) is G - E - F - K - J. The cost to travel this is 4000. And the cost to reach G from A is 2600. So total cost =

2600 + 4000 = 6600.

4.5.4 Comparison

We plotted a graph for 2 cases: a network with unlimited buffer and one with buffer of 20 to analyze the performance of the 2 strategies. These are shown below. What we can infer from these is that for the case of unlimited buffer, intermediate node forwarding is very efficient. And for the case of limited buffer, intermediate node forwarding's efficiency starts to suffer for large no. of packets.

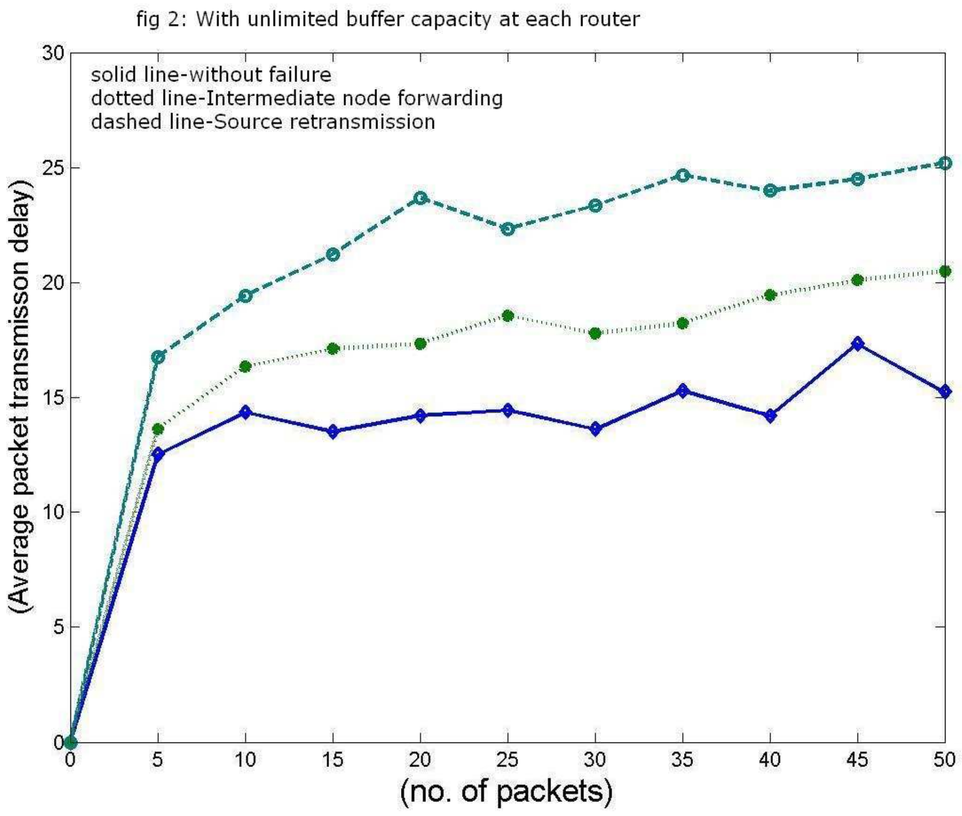

Fig 4.2 Performance analysis of a network with unlimited buffer

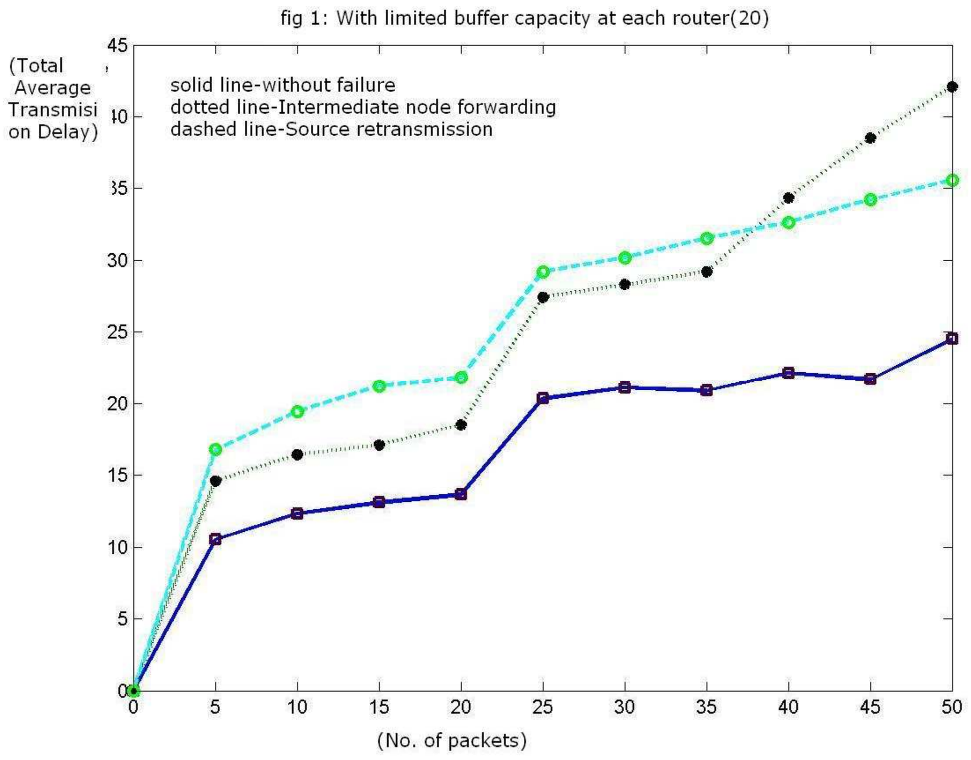

Fig 4.3 Performance analysis of a network with limited buffer

CHAPTER 5

CONCLUSION

5. CONCLUSION

In this project work we proposed 2 schemes for resilience in the network. Mainly
i) Source retransmission strategy
ii) Intermediate node forwarding.

In the source retransmission strategy fault is detected at the intermediate node and informed to the source. The source then finds an alternative path to the destination and routes the packet on the alternative path.

In the intermediate node forwarding strategy, an alternative path is found to the destination by the node where the fault arises. And packets are forwarded by this node. We have simulated both the strategies with a 14-node NSFNET standard network. Basically, we have calculated delays associated with each packet. Such as

1) Processing delay
2) Queuing delay
3) Transmission delay
4) Propagation delay

Packets are generated at each node in our simulation continuously and randomly and propagate to their destination. In case of failures, we have done the performance analysis of the above discussed 2 strategies and conclude that intermediate node forwarding behaves better with a network with unlimited buffer. But in a limited buffer network, as the no. of packets increase, source retransmission proved better.

CHAPTER 6

REFERENCES

References

[1] C. Chekuri, A. Gupta, A. Kumar, J. Naor, D.Riaz *Building edge-failure resilient network*

[2] Richard G. Ogier and Nachum Shacham, *A Distributed algorithm for finding shortest pairs of disjoint paths*

[3] www.rstn.net/solutions/**resilient**_ethernet.shtml

[4] C. Chekuri and S. Khanna, *Edge disjoint paths revisited*

[5] Mark Alan Weiss *Data Structures*

[6] S. Madhavapeddy, M. Dietzfelbinger and I.H. Sudborough – *Three disjoint path paradigms in nsf networks*