An Efficient Token Management Algorithm for Message Dependent Deadlocks Recovery Architecture

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Submitted By
Nidhi Garg
(800931013)

Under the supervision of:
Dr. Rinkle Rani
Assistant Professor,CSED
Thapar University,Patiala

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2011

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*An Efficient Token Management Algorithm For Message Dependent Deadlocks Recovery Architecture*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr.(Mrs.)Rinkle Rani* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Nidhi

(Nidhi Garg)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Rinkle

(Dr. Rinkle Rani)
Assistant Professor,
Computer Science and Engineering Department.

Countersigned by:

(Dr. Maninder Singh)
Head,
Computer Science and Engineering Department,
Thapar University,
Patiala.

(Dr. S. K. Mohapatra)
Dean (Academic Affairs),
Thapar University,
Patiala.

Acknowledgement

I would like to express my deepest appreciation to Dr. (Mrs.) Rinkle Rani, my mentor and thesis supervisor for her constant support and motivation. She had been instrumental in guiding me throughout the thesis with her valuable insights, constructive criticisms and interminable encouragement.

I am also thankful to Dr. Maninder Singh, Head, Computer Science and Engineering Department and Mr. Sumit Miglani, P.G. Coordinator for their constant support and encouragement.

I express my thanks to my family for their support and affection and for believing in me always.

In the end, I would like to thank all the faculty members and staff of the department and my friends who directly or indirectly helped me in the completion of this thesis.

Nidhi

Nidhi Garg

(800931013)

Abstract

Interconnection networks are the backbone for communication in multicomputer environment based on point-to-point switches which are benefitted from the parallelism offered by non-blocking switches capable of forwarding packets at full link speed concurrently between input and output ports. These networks being lossless networks are more prone to deadlocks where probability of occurrence of deadlock is rare; there deadlock recovery strategies are used and are discussed in this thesis. Two kind of deadlock occurs Routing dependent and Message dependent where as routing dependent deadlocks occur when router allocates network resources to the messages in such a way that the complete set of resource dependencies form knotted cycles along escape resources. Techniques for handling routing deadlock are not sufficient for dealing with message-dependent deadlocks. There are various approaches to avoid message dependent deadlock but these techniques are not very scalable .So recovery technique is used called m-disha where network is recovered from message dependent deadlocks by giving an alternate path on the network which pre-empts the controller .Again there is a limitation in this is that no reliable token management system.

In this thesis, a new algorithm has been proposed to manage token in the network. The proposed algorithm uses pre-defined arrival time and hop time to manage token. The results have been shown on how the proposed algorithm manages the token which was not shown in previous work.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Interconnection networks are reliable communication being used in many different applications ranging network among the processing from (VLSI) to WAN. These applications include [8].

- Telephone Switches
- Interconnects for multicomputer
- Cluster of Work Stations
- Local Area Network

Two main parallel computer architectures exist:

a. Physically shared-memory parallel computer where N processors access M memory modules over an interconnection networks [12].
b. Physically distributed –memory parallel computer where a processor and a memory module form a processor–memory pair that is called processing element (PE). All N PEs are interconnected via an interconnection network as shown in Figure 1 [12].
c. In a message-passing system, PEs communicate by sending and receiving single messages while in a distributed-shared-memory system, the distributed PE memory modules act as a single shared address space in which a processor can access any memory cell. This cell will either be in the memory module local to the processor, or be in a different PE that has to be accessed over the interconnection networks.

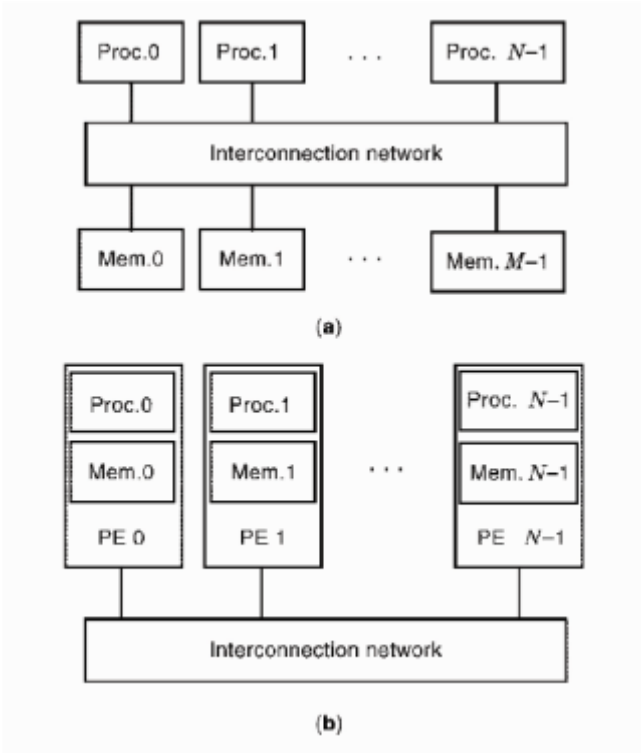

Figure 1. a) Physically Shared memory b) Distribued memory Parallel Computer Architecture [12]

1.1 Classification of Interconnection Networks

Interconnection Networks can be classified according to different categories based on network topology shown in Figure 2 [8]:

- Shared-medium network
- Direct Networks
- Indirect Networks
- Hybrid Networks

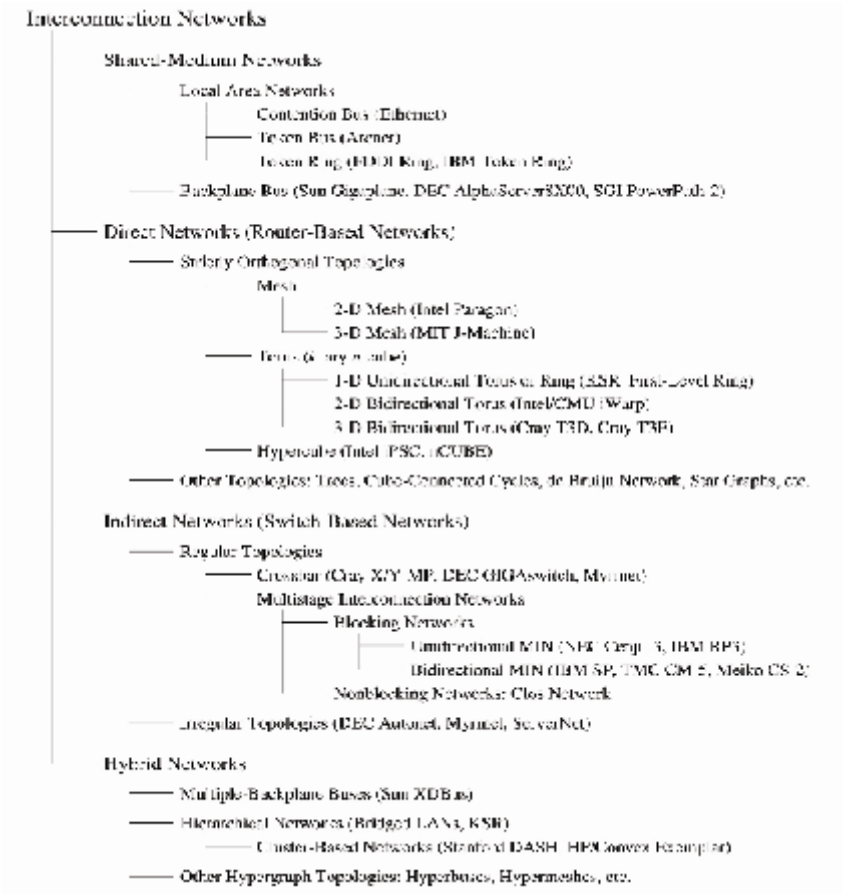

Figure 2. Classification of Interconnection Networks [8]

1.2 Direct networks

Direct networks have been traditionally modeled by a graph G(N,C) where the vertices of the graph N represent the set of processing nodes and the edges of the graph C represent the set of communication channels as shown in Figure 3 [2,8].

Some basic network properties:

- ***Node degree****:* Number of channels connecting that node to its neighbors.
- ***Diameter****:* The maximum distance between two nodes in the network.
- ***Regularity:*** A network is *regular* when all the nodes have the same degree.
- ***Symmetry****:* A network is *symmetric* when it looks alike from every node.

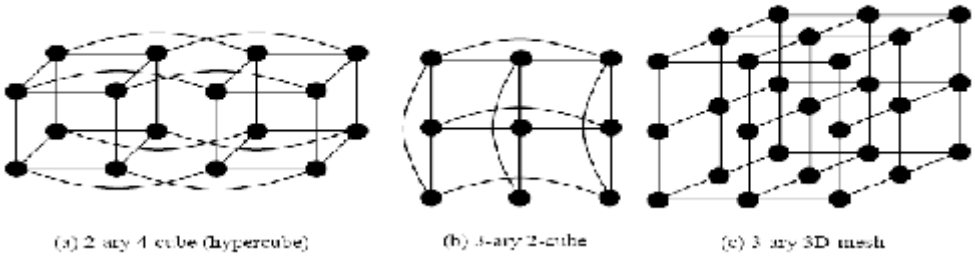

Figure 3. Direct networks

1.3 Characteristics of Interconnection Networks

1.3.1 Network Topology

The topology of a network defines how the nodes are interconnected and modeled as graphs in which vertices represent the nodes and the edges denotes the channels. A network topology is orthogonal if and only if nodes can be arranged in an orthogonal n-dimensional space, and every link can be arranged in such a way that it produces a displacement in single dimension [8].

Strictly Orthogonal Topology Classification:

a) Meshes

The mesh is an asymmetrical topology in which the node degree depends on its location. Performance depends on the location of source and destination. Channels near the centre of the mesh experience have higher traffic density than those on the periphery.

b) k-ary n-cube:

k-ary n-cube is defined as an interconnection structure of n dimensions having k nodes in each dimension. Each node in the k-ary n-cube is identified by an n-coordinate vector $(x_0, x_1, \ldots, x_{n-1})$, where $0 \leq xi \leq k - 1$. Two nodes $(x_0, x_1, \ldots, x_{n-1})$ and $(y_0, y_1, \ldots, y_{n-1})$ are connected if and only if there exists an i such that $xi = (yi \pm 1) \bmod k$, and xj =yj for all $j \neq i$. There are wraparound channels in the k-ary n-cubes which are not present

in *n*-dimensional meshes. If $k = 2$, then every node has n neighbours, one in each dimension. If k >2, then every node has 2n *neighbours*, two in each dimension.

c) Hypercube:

Hyper cubes are special cases of an *n*-dimensional mesh in which $ki = 2$, for all i, $0 \leq i \leq n$ -1; they can be termed 2-ary *n*-cubes.

d) Torus:

A *k*-ary *n*-cube is called a torus when $n = 2$. A torus can be constructed by connecting the corresponding end nodes of the 2-D mesh with wraparound connections. The channel bandwidth of the three topologies can be expressed as: mesh >torus > hypercube.

1.3.2 Routing Techniques

It determines the path selected by a packet to reach its destination.

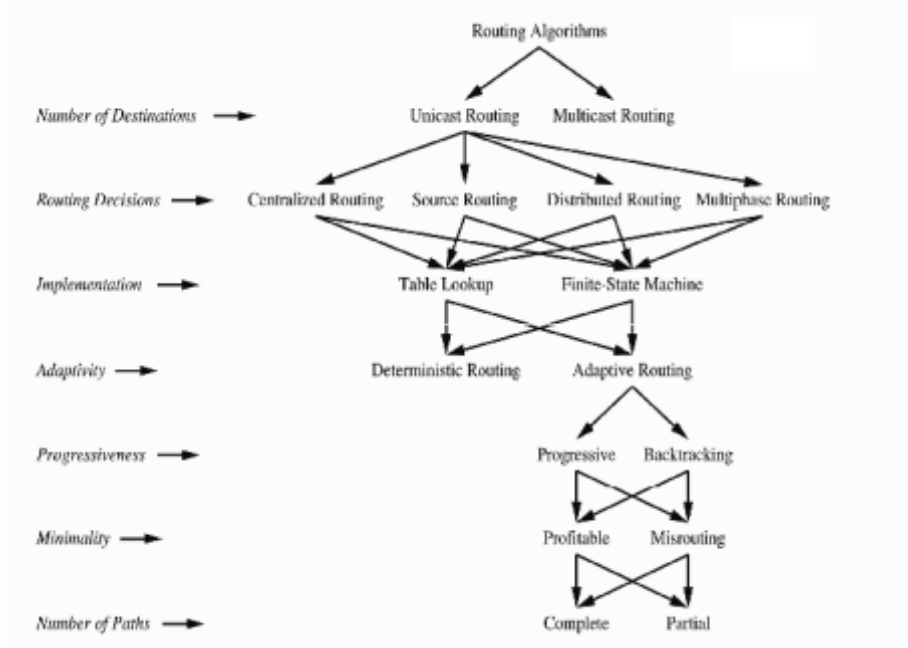

Figure 4. Routing Techniques [8]

1.3.2.1 Routing Algorithms Classification:

A routing algorithm determines a path for a packet to reach its destination. It must be decided within each intermediate router which output channels must be selected for the incoming messages. There are various types of routing algorithms depending on various network parameter as shown in Figure 4:

a) *Deterministic Routing Algorithm:*

In this the path is completely determined by the source and destination addresses also referred to as Oblivious Routing**.** These algorithms [1] always generate the same single routing path for a given pair of source and destination, usually choosing the shortest one. Only the addresses of current and destination nodes are used to compute the path. As messages with the same source and destination always use the same network path, they cannot use alternative paths to avoid blocked channels. This algorithm should be progressive and profitable, which means that the header should move forward reserving a new channel at each routing operation, under condition that the supplied channel always brings the packet closer to the destination. Thus deterministic routing algorithms use greedy algorithms, always choosing the shortest path.

b) *Non Deterministic or Adaptive Routing:*

In this the path taken by a particular packet depends on dynamic network conditions such as presence of faulty or congested channels. These algorithms do not restrict a message to a single path when traveling from the source to the destination. While making a decision the current network conditions are considered. This makes the routing more flexible and reduces unnecessary waiting time delays, providing better fault tolerance. Adaptive routing algorithms contain two functions: Routing and Selection.

Routing function gives a set of output channels based on the current node and destination Node.

Selection function selects the most appropriate output channel from that set.

1.3.3 Flow Control:

It deals with the allocation of channels and buffers to a packet as it travels along the path through the network. Resource collision occurs when a packet cannot proceed because some resource that it requires is held by another packet. A good flow control policy should avoid channel congestion while reducing the network latency [8]. Allocation of channels and their associated buffers to packets can be viewed from 2 perspectives:

- Output Selection Policy

  It is requested by packets arriving on many different input channels.
- Input Selection Policy

  It determines which packet may use output channels.

1.3.4 Switching

Nodes in a direct network communicate by passing messages from one node to another. The mechanism to transfer message through the network is called switching. Message may be divided into one or more equal or variable-size packets. Packet is the smallest unit of information that contains routing and sequencing information.

1.3.4.1 Various switching Techniques

Switching techniques determine when and how internal switches are set to connect router inputs to outputs and the time at which message components may be transferred along those paths. These techniques are coupled with flow control mechanisms for the synchronized transfer of units of information between routers and through routers in forwarding messages through the network [8, 20].

a) *Circuit Switching:*

Messages that are sent through network during connection establishment which means a dedicated path or a physical circuit is

established between source and destination before data transfer initiates as shown in Figure 5(a) [4].

Advantages: No buffer is needed in this technique and latency should be minimal once the path is established.

Drawbacks: Insufficient use of network resources i.e. no link required to transfer the message [6].

b) *Packet Switching*:

This technique is also called Store and Forward Technique [2].In this message is divided into one or more data packets and routing information is added to each packet. These packets are sent through the network without the establishment of an overall connection between the source and destination. Network resources are reserved only when needed by a packet and it will be forwarded to the next node only if it was completely received by the current node as shown in Figure 5(b).

Advantages: physical links are used only when needed and if messages are short and frequent.

Drawbacks: Require buffer on each and every node and latency should be high which is proportional to the number of links travelled.

c) *Virtual-cut-through Switching*:

Combination of Packet and Circuit switching A message is divided into fixed-size packets**.** All packets of the message travel through the same route-so only the first packet need to carry routing information. The first packet is responsible for finding paths from source to destination-no need to set up the path before-hand. If the next required link is available, a packet is immediately passed forward without being stored in the intermediate node's buffer. If the complete path were available, the whole message would pass through the destination with minimal latency. If path is blocked then buffering at intermediate node is still needed.

Drawbacks: If path is blocked, the whole message needs to be buffered at an intermediate node.

d) Wormhole Switching:

In this packet is divided into number of small flits in order to decrease the buffer size at routers and to achieve much faster routers and are transmitted in a pipelined fashion[15]. The header flit(s) of a message contains all the necessary routing information, all the other flits contain the data elements and builds a path in the network, which the other flits follow as shown in Figure 5(c) [8, 20].

Blocking of packets is the main problem so it can be solved using Wormhole by simply stops every flit in its current position. This blocks the resources in case of stalled pipelines and makes the wormhole technique deadlock-prone.

Advantage: Avoids the large buffer problems of the original cut-through approach. The main advantage of wormhole switching derives from the pipelined message flow, the message moves flit by flit across the network so that each node needs to store only one flit.

Drawbacks**:** Header flit cannot advance in the network due to resource contention, all the trailing flits are also blocked along the path and these blocked messages can block other messages because header flit has routing information. This chained blocking can also lead to deadlock where messages wait for each other in cycle and hence no message can advance further.

1.4 Deadlocks in Interconnection Networks

In Interconnection direct networks where packet usually travels across several intermediate nodes before reaching to the destination. They require high bandwidth shortened time delay and high delivery over short distances. A decade ago interconnection networks were bus-based systems but as the time passed pressure has

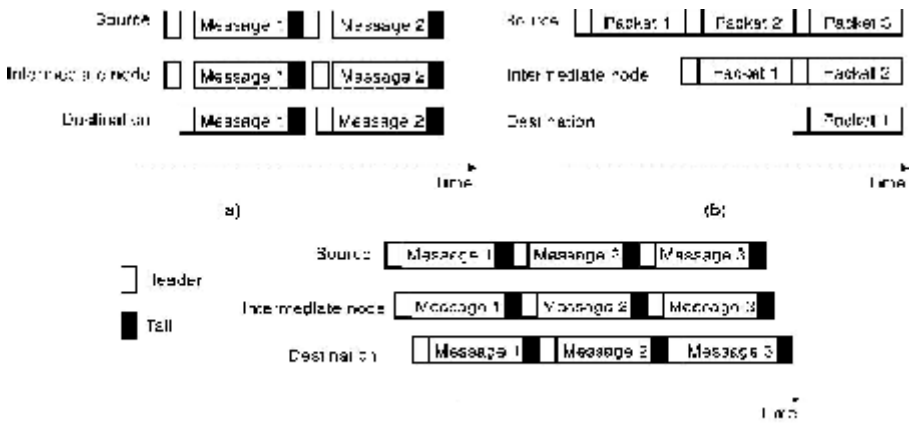

Figure 5. Data Transport through an intermediate node in a) Circuit Switching Network b) Packet Switching Network c) Wormhole Routing Network [12]

Table 1. presents the summary of the various switching techniques

Table 1. Summary of switching techniques [5]

Switching technique	Communication entity	Path reservation	Buffer size	Resource utilization
Circuit switching	Flit	Yes	Small	Poor
SAF switching	Packet	No	Large	Good
VCT switching	Packet	No	Large	Good
Wormhole switching	Flit	Yes	Small	Moderate

increased on data centers so interconnection networks have migrated from point-to-point switch based networks where non-blocking switches are capable of forwarding packets at full link speed concurrently between input and output ports. They mainly use techniques like virtual cut-through or wormhole switching. In both of these techniques, when a packet header arrives at a switch, then switch immediately routes this packet onwards which reduces packet latency in large interconnection networks. When comparing interconnection networks to traditional computer networks then no packet loss occurs in interconnection networks. Such lossless networks are susceptible to deadlocks shown in Figure 6, because buffer resources of one channel will have direct and indirect dependencies on buffer resources of other channels. In that case the system must choose routing strategies to avoid deadlocks. If all the switches forward their packets counter clock-wise, deadlock occurs when all switches simultaneously try to forward streams of packets to the switch located diagonally opposite. The packet will need to make two hops to reach their destination but are blocked because they need access to the buffer in the next switch which is already occupied. Thus the routing function causes a cyclic dependency between the channel's buffer resources.

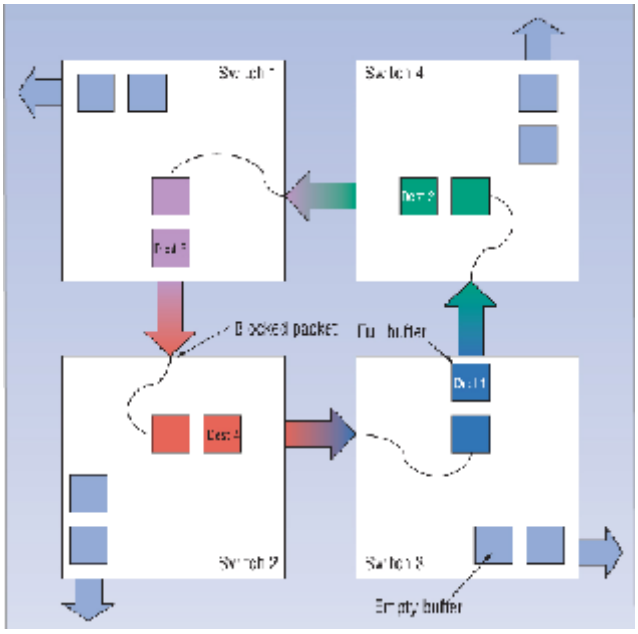

Figure 6. Deadlock configuration example. The packets in switches 1, 2, 3, and 4 are destined for switches 3, 4, 1, and 2, respectively. A lack of buffer space in the next switch blocks each packet in this set and, since each packet awaits another packet in the set to advance, all packets remain blocked [18]

In direct networks, packets usually travel across several intermediate nodes before reaching to the destination. In switch based networks, packets usually traverse several switches before reaching the destination. So it may happen that some packets are not able to reach their destination those are called undelivered packets as shown in Figure 7.

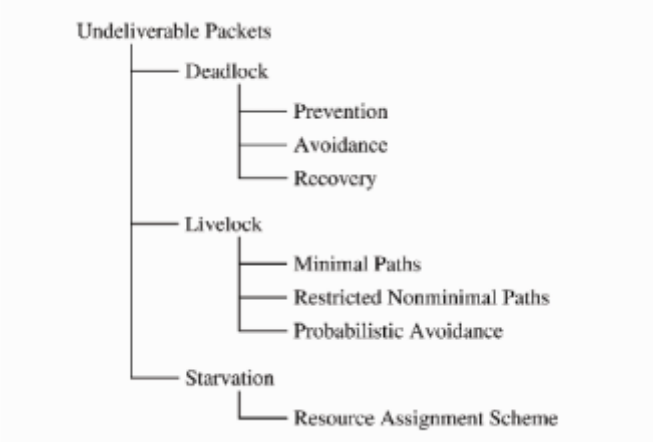

Figure 7. Classification of the situations that may prevent packet delivery[8]

Deadlock:

It is one of the situations that can postpone packet delivery indefinitely. It happens when a packet is requesting a resource that is held by another packet while holding the resource that is requested by other packet. So there is a cyclic dependency between channels occurs and the packet may be blocked forever. It is one of the most difficult problems to solve. There are three different strategies that can cope up with deadlock: deadlock prevention, deadlock avoidance and deadlock recovery. When cyclic buffer dependencies develop from the topology and routing algorithm the deadlock arises in the network. As buffers fill, packets begin waiting. If a cycle of waiting packets develops, then no progress can be made and the packet waits forever.

a) *Deadlock Prevention:* In prevention resources are granted to a packet in such a way that a request never leads to a deadlock. All the required resources are reserved before starting packet transmission.

b) *Deadlock Avoidance*: In avoidance resources are requested as the packet advances through the network that are granted to a packet only if the resulting global state is safe. This strategy avoids sending additional packets to update the global state because these packets consume network bandwidth and they may contribute to produce deadlock.

c) *Deadlock Recovery*: When resources are granted to a packet without any check then chances of occurrence of deadlock is possible, so some detection mechanism must be provided to check the presence of deadlock. If deadlock is detected, some resources are de-allocated and granted to other packets. In order to de-allocate resources, packets holding those resources are usually aborted.

Livelock:

When a packet is running forever in circular motion around its destination, because the channels that are required to reach the destination are occupied by other packets called livelock. And in order to remove livelock several techniques have been proposed such as minimal path, restricted non-minimal path, and probabilistic avoidance[8].

Starvation:

When a resource that was requested by a packet is always granted to other packets and the packet stops permanently in never getting the resource it needs called starvation. It can be avoided by using correct resource assignment scheme.

1.4.1 Classification of deadlocks based on knot cycle density:

1.4.1.1 Single Cycle deadlocks:

Deadlocks having knot cycle density of one cycle are called single-cycle deadlocks as shown in Figure 8. It forms under minimal adaptive routing only when one routing option is available to all messages comprising the deadlock set[11,21]. In single-cycle deadlock having value of fan-out=1.

1.4.1.2 Multi cycle deadlocks:

Deadlocks having knot cycle density greater than one cycle called multi-cycle deadlocks[21,30]. Multiple resource dependency cycles occur. Network uses minimal adaptive routing and two virtual channels per physical channel per direction as shown in Figure 9. For deadlock to occur all messages in the deadlock set must be blocked waiting to acquire any virtual channel which are owned by other members of the deadlock set. Outgoing arc per blocked message is greater than one i.e. fan-out >1.

1.4.1.3 Cyclic Non-deadlocks:

When multiple cycles exist which does not result in deadlock it is referred to as cyclic non-deadlock[21,30]. It can lead to a situation where messages block cyclically faster than they can be released, therefore remained blocked for extended period of time causing increased latency and reduced throughput as shown in Figure 10.

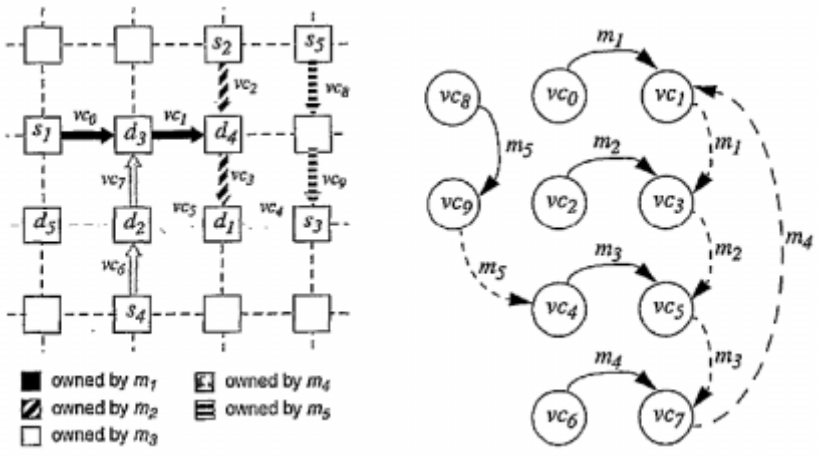

Figure 8. (a) A "Single-cycle deadlock"(consisting of messages {m1, m2, m3, m4} formed under minimal adaptive routing in a network with 1 virtual channel per physical channel in each direction . (b) The set of vertices {vc1, vc3, vc5, vc7} forms a knot in the channel wait-for graph [30]

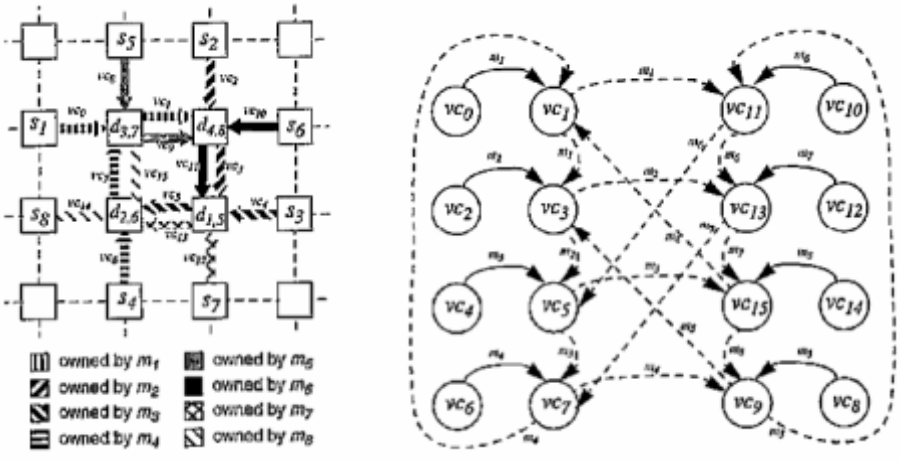

Figure 9. (a) A "Multi-cycle deadlock"(consisting of messages {m1, m2, m3, m4, m5, m6, m7, m8} formed under minimal adaptive routing in a network with 2 virtual channel per physical channel in each direction . (b) The set of vertices {vc1, vc3, vc5, vc7, vc9, vc11, vc13, vc15} forms a knot in the channel wait-for graph [21]

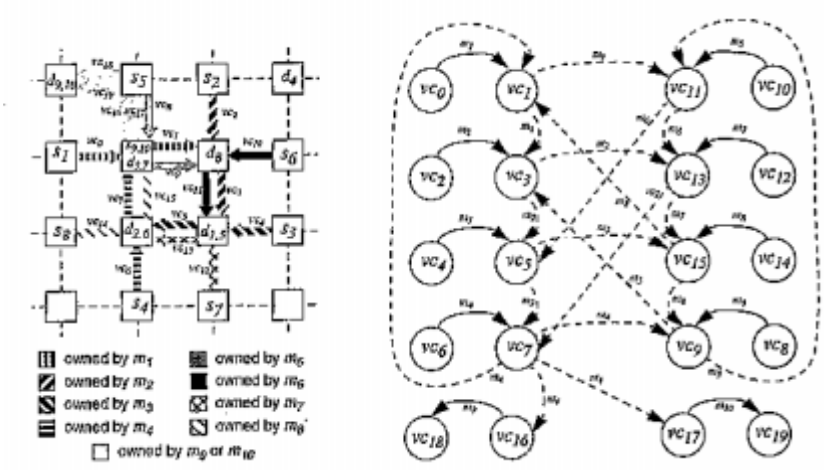

Figure 10. (a) A "Cyclic-non deadlock" network state under minimal adaptive routing with 2 virtual channels per physical channel in each direction. (b) There are multiple cycles but no knots in the channel wait-for graph [21]

1.4.2 Factors influencing deadlock

Number of interrelated factors influences the probability of deadlock formation as shown in Table 2:-

Routing Freedom:

Number of routing options available to a packet being routed at a given node within the network is called routing freedom and can be increased by adding physical and virtual channels.

Number of blocked packets/messages:

Blocked messages are those messages in the network which cannot acquire any of the alternative channels required to make progress at any given point in time. The capacity for blocked messages increases as the network's capacity to hold messages increases.

Number of resource dependency cycles:

As the number of network resources i.e. physical/virtual channels and the routing options allowed on them by the routing freedom increases, the number of packets that tend to block significantly decreases, also the number of resource dependency cycles decreases and the resulting probability of deadlock also decreases.

Table 2. (a) A How key network factors directly affect routing freedom and the capacity for blocked messages, and. (b) How they indirectly influence the probability of deadlock [21]

Network Factors	Routing Freedom	Capacity for Blocked Messages
Increased Routing Adaptivity	▲	
Increased Physical Channels	▲	▲
Increased Virtual Channels	▲	▲
Increased Buffer Depth		▲
Increased Message Length		▼

Effects of Network Factors	Number of Blocked Messages	Potential for Cycles	Probability of Deadlock
Increased Routing Freedom	▼	▲	▼
Increased Num. of Blocked Msgs.		▲	▲
Increased Number of Cycles	▲		▲

▲ Increases ▼ Decreases ▲ Exponentially Increases ▼ Exponentially Decreases

The resources of an interconnected network are shared among attached end nodes whereas limitation of the network is that resources prevent packet transmission in the network. While unbalanced use of network resource causes traffic burst. Such contention causes packet to be delayed may lead to deadlock. Virtual channels were introduced to avoid deadlock in network with cyclic links each virtual channel is implemented by a separate pair of message buffers, allow messages to make forward progress rather than to remain blocked behind a preceding message deadlock can occur as a result of circular dependencies on network resources by messages or packets. Once deadlock occur that reduces communication, efficiency and reliability which consequently degrading network performance and system performance. Messages in interconnection networks may subject to 2 kinds of deadlocks: routing-dependent deadlocks and message dependent deadlocks. Routing deadlocks can occur when the network delivers connection-less datagram messages and the routing function in router allocates network.

Avoidance-based deadlock handling techniques [16, 29] prevent network-wide cyclic dependencies from occurring by forcing packets to use some or all of the resources. These techniques trade-off routing flexibility for deadlock freedom. By contrast, recovery-based techniques [16, 26, and 29] allow deadlocks to form, and recover from them when detected. Considering that deadlock rarely occurs for typical traffic loads and network parameters, recovery techniques allow for better utilization of network resources.

Techniques for handling routing dependent deadlocks is not sufficient for handling message dependent deadlocks because they assume that the messages in the network always sink upon arrival at their destinations i.e. assumed that the delivery of messages is not coupled in any way to the generation and consumption of any other message in the network or at network end points. This assumption is valid for network with homogenous message types but not valid when dependencies between different message type exists thus eliminating routing dependent deadlocks by designing deadlock free routing algorithms. However,

they are susceptible to deadlocks arising from the interactions and dependencies created at network end points between different message types. This type of deadlock called message dependent deadlocks.

Message-dependent deadlocks, once they occur, block related resources at network endpoints as well as inside the network, although routing algorithms themselves avoid routing dependent deadlocks inside the network.

The advantages of techniques based on these approaches depend upon how frequently deadlocks occur and how efficient messages can be routed while guarding against deadlocks. The main objective of deadlock handling technique is to prevent network resources from being blocked forever without making progress in packet delivery.

1.5 Organization of thesis

The remainder of this study is structured as follows Chapter 2 describes review techniques for handling deadlocks. Chapter 3 defines the problem statement. Chapter 4 presents the previous technique for handling message dependent deadlocks and proposes an efficient algorithm of managing token in message dependent deadlock architecture. Results and findings are presented in Chapter 5 and finally Chapter 6 concludes the thesis work and gives future directions.

Chapter2 Literature Review

2.1 Deadlock Recovery

When deadlock is detected, one or more packets are compelled to release the buffer resources they are keeping, allowing other packets to use them and breaking the deadlock called Deadlock Recovery [8]. It is an alternative to deadlock avoidance or prevention. A detection mechanism identifies potential deadlock configurations; once deadlock is detected, a recovery scheme breaks the deadlocked cycle.

Detection of Potential Deadlocks

A packet is deadlocked when it is simply waiting for a channel occupied by a long packet. If the header flit of any packet is blocked for a certain amount of time period than the corresponding packet is potentially deadlocked. Deadlock detection mechanisms using heuristic time out mechanism can be implemented either at the source node or at intermediate nodes that contain a packet header.

There are various techniques discussed in this review depending on routing dependent deadlocks which all are based on heuristic time-out mechanism. Whereas Kim and Chien gave Compression- less routing technique where additional dummy flits are added to normal messages if message length is small then packet can be aborted and resent later.

2.1.1 Deadlock Recovery Techniques

Progressive Technique:

It de-allocates resources from other normal packets and reassigns them to other deadlocked packets.

Regressive techniques:

It de-allocates resources from deadlocked packets; usually killing them and the set of possible actions taken depends on where deadlocks are detected or not.

If deadlocks are detected at the source node, regressive deadlock recovery is used as in Compression-less Routing..

2.1.1.1 Compression-less Routing:

This recovery technique is based on heuristic time –out mechanism where a packet can be killed by sending a control signal that releases buffer and propagates along the path reserved by the header called Compression-less Routing. Dummy flits are added to normal messages called message padding, and these messages cannot leave the source node until the header flit reaches its destination. Buffers are released from the current node, reversing back along the path reserved by the message after a certain time-out period [15]. When a message finds all the usable virtual channels busy at a given physical channel, it can wait up to a fixed time-out period for one of the virtual channel to become free again. When message experiences a transmission failure due to time out at an intermediate router, the source retransmits the message after a random time gap. The packet is re-injected again into the network after some time period which requires a packet buffer associated with each injection port. Thus sender determines if the message is not long enough, then sender pads it to ensure that the header reaches the destination before the last flit is injected by the source as shown in Figure. 11. During routing, if a message is blocked at a node for an interval larger than a preset time-out, it is aborted by the source and resent later. The performance of this routing depends on the time-out interval.

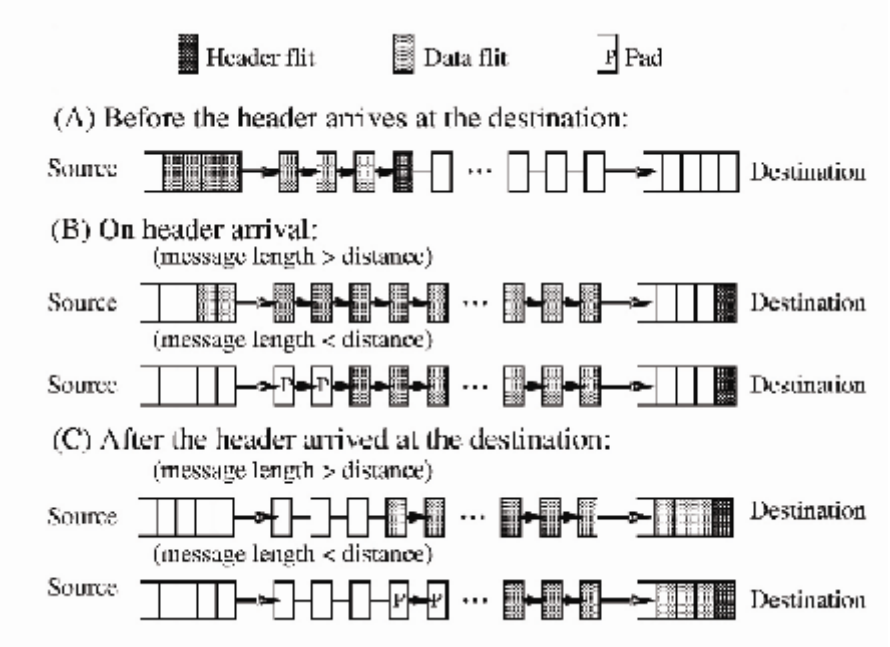

Figure 11. Message routing and padding in CR networks

If deadlocks are detected at an intermediate node containing the header, then both regressive and progressive deadlock recovery are possible.

2.1.1.2 Disha

As Chien and Kim gave an routing technique in which blocked message is aborted and resent later where as Anjan and Timothy gave another technique named Disha. This is a progressive deadlock recovery technique in which instead of killing a deadlocked packet, resources are temporarily de-allocated from normal packets and assigned to deadlocked packet so that it can reach to its destination. And later on, resources are re-allocated to the pre-empt packets which is more efficient than regressive deadlock recovery technique. Disha supports deadlock-free fully adaptive wormhole routing. By giving required hardware towards effective deadlock recovery routing becomes flexible. This is applicable to any interconnection network topology where no virtual channels are required for prevention of deadlocks which is based on true fully adaptive wormhole routing as it permits unrestricted routing on all existing virtual channels. An additional buffer called deadlock buffer is added which is central

to the router form deadlock free lane which can be accessed by all adjacent nodes. When all required virtual channels are busy then message is blocked at a specific intermediate router. Also if the blocking time exceeds a certain time-out threshold value than the message is presumed to be deadlocked. Now that message is switched to the deadlock-free lane where it is progressively routed using the central deadlock buffer until it reaches to the destination. Deadlock free lane can be implemented by temporarily de-allocating network bandwidth from normal messages and assigning it to deadlocked messages. As the deadlocked message passes, network bandwidth is re-allocated to preempted messages. In disha a cost effective single deadlock buffer which is shared between adjacent nodes is added [4]. Routers designed are not only simple and faster, but results in raised communication efficiency and improved network performance. This scheme is extremely efficient even when recovery is sequential i.e. disha sequential.

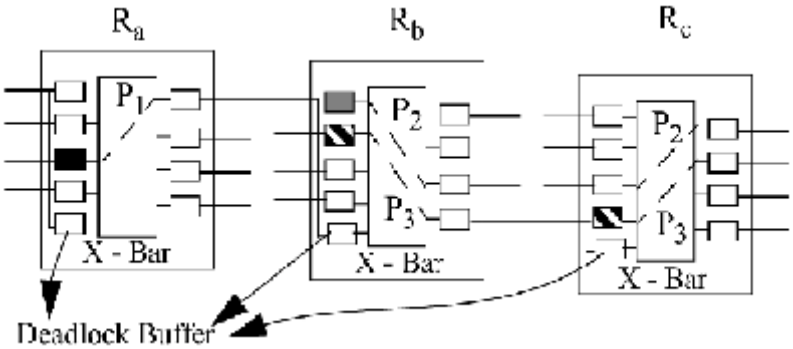

Figure 12. (a) Shows a possible deadlock scenario where P1 would otherwise wait on P2 indefinitely.

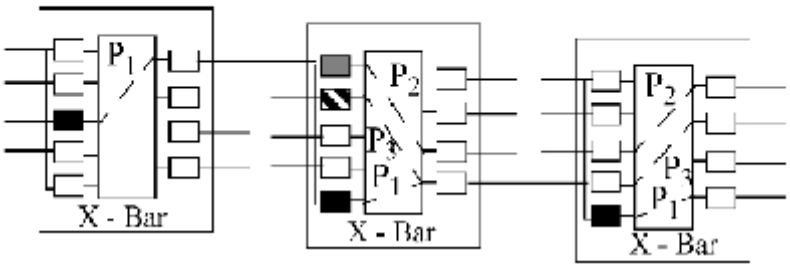

Figure 12. (b) Shows how deadlocks are resolved P1 is routed around P2 on the Deadlock Buffer to break the cycle.

Deadlock Recovery with Disha

In disha the intial state of 3 Routers R_a, R_b and R_c are given in Figure 12(a). In this packet P_1 is blocked by P_2 and is unable to proceed. Router R_a is unable to send out the header of P_1 for certain time-out period and assumed packet is being in deadlocked state. Now the Router R_a sends out the header of P_1 on the deadlock buffer, bypassing the normal buffer currently occupied by P_1 followed the deadlock buffer path through subsequent router to reach its destination as shown in Figure 12(b). This does not have any single point of failure, and simultaneous recovery from multiple deadlock cycles is not possible with disha sequential as it operates on flit-by-flit crossbar switches.

Disha Concurrent Recovery [3][24]

This technique disha concurrent is free from the requirement of mutually exclusive access to the deadlock-free lane by structuring the routing on the central deadlock buffer such that cyclic dependencies do not occur. Also no supplementary virtual channel is used. For this firstly construct a Hamiltonian path on the network using deadlock buffer then label the nodes in sequence along the Hamiltonian path. The deadlock buffer at a node can be used only by a deadlocked packet at a neighboring node for which the packet's destination has a higher label than the node to which the buffer belongs. Once a packet is placed on a deadlock buffer, which is restricted to using only deadlock buffers at consequent nodes until it is delivered. Consecutive deadlock buffers can be used only in increasing label order that are not necessarily in sequential and although we define a Hamiltonian path on the deadlock buffer. The Hamiltonian path as constructed is equivalent to the high-channel network. This structured path allows a deadlocked packet to reach at any node with a higher label. Hamiltonian path is acyclic and routing on the deadlock buffers is deadlock free even when short-cut paths are allowed[13]. Deadlocked packets at nodes higher than the destination label nodes use normal edge buffers to reach at an intermediate node lower in label than the destination, which they use the acyclic deadlock buffer path to arrive at the destination. The two approaches sequential and

concurrent both are based on the same progressive recovery approach, whereas the limitation of disha sequential is that it will not handle multiple deadlocked messages that are overcome in disha concurrent by constructing Hamiltonian paths. The improved performance in disha concurrent is more reliable and at the less implementation cost.

2.1.1.3 Software-Based Progressive Recovery Technique[19]

This technique require no buffer to handle deadlock when detected it only absorbs the deadlocked message at the current node and later re-injects it at the current node for continued routing towards its destination called Software-Based Deadlock Recovery Technique. The deadlock detection mechanism only assumes that a message is deadlocked if its header is being routed and at least one of the messages from all that involved in deadlock will have its header at the head of an input buffer, waiting for an output channel. When router detects a deadlocked message, in order to recover from this it remove that message from the network by ejecting it at the current node. The router selects the internal memory channel at the current node for this message, as if this node were its destination. A control bit is required to differentiate between normal and deadlocked messages. Finally, removed messages must be re-injected into the network at a later time [2]. If the software messaging layer receives any message whose destination is not the current node, it must inject the message again into the network. The true message destination can be found in the message header. The mechanism is a low-cost progressive deadlock recovery technique. Instead of killing deadlocked messages, it absorbs them at the current node, allowing them to make advance at later time. The main advantage of this technique is its simplicity. This recovery mechanism does not even require dedicated buffers in the router to recover from deadlock. Therefore, this mechanism provides a simple router design which is an improvement over disha but on the other hand it assumes that a processor is associated with every node, and this is not true for all networks so in that case, disha should be preferred. On the other hand, this mechanism is a software solution which is always slower than a hardware one. By taking this into account that deadlocks are not frequent, so this mechanism will not be used frequently. This

recovery mechanism provides more routing flexibility and also eliminates the use of buffers for deadlock recovery but on the other hand this does not increase performance over disha.

2.1.1.4 ZOMA: Deadlock-Recovery Mechanism [1]

ZOMA is Pre-emptive deadlock-recovery mechanism. It allocates a central buffer of the same capacity as the edge buffer. This central buffer will be used to break a deadlock cycle at the node where the header of the blocked packet occupies through the break/reconnect line. When a deadlock situation is detected at a node, the edge buffer that contains the header of the blocked packet is pre-empted and stored in the central buffer, and then cleared to receive other flits. Flow control in the opposite direction, using the added line, signals the previous node where the packet came from to break and free the connection through its crossbar, to enable other packets to pass through it. This operation is referred to as a break mode operation as shown in Figure 13. Each node containing part of the blocked packet circulates break mode signal to the previous node until the source node of the packet has been reached. Break mode operation performs the following steps firstly it moves the flits of the deadlocked packet from the edge buffer to the central buffer. Then the input and output port numbers allocated by the packet through the crossbar of that node are saved after that clears the connection made by this packet through the crossbar, so that other packets can utilize it then toggles the break/ reconnect bit. After that, the remaining packets that are involved in the deadlock can now progress and will finally freed up their occupied resources. Once the node with the pre-empted header in its central queue has a free edge buffer, the header is re-routed again. If the routing is successful, then re-connect signal is sent back to the previous node containing the remainder of the packet in order to re-establish the connection through its crossbar, and restart the pre-empted routing operation of the packet. This signal called as a connect mode operation circulates back along the same path taken by the break mode signal and reverses what the latter has done by using the previously saved trails. The hardware required by this approach is moderate and neither has it increased the complexity of

the crossbar switch, nor does it not lie on the critical path of the routing decision for normal packets.

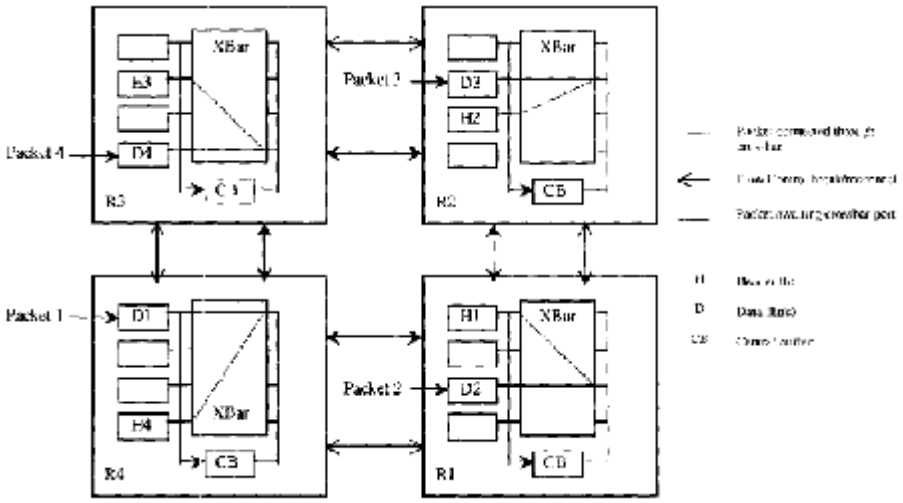

Figure 13. Single Cycle deadlock in proposed router design ZOMA

A central buffer is required per node and the capacity of this central buffer should be the same as that of an edge buffer. The central buffer is connected to all the input ports of the switch and can output to any output port. Two registers per node are needed to store the input and output switch ports. A break/reconnect signal wire is also needed in order to pre-empt and restart routing of a packet. Its main function is to signal the previous node to perform either a break mode or a connect mode operation. Instead of having use of two wires for each mode, use one wire with a bit in each node. When a break signal is received by a node, it toggles this bit, so that when next time it receives, will actually be interpreted as a connect signal. In order to ensure sequential deadlock recovery, a synchronization token shared amongst all the nodes in the network is required, such as the one used in disha. This hardware is sufficient to perform sequential deadlock recovery i.e. one deadlocked packet is handled at a time.

2.1.1.5 Suspensive Deadlock Recovery [28]

In this recovery technique called an escape-restoration routing, a small amount of exclusive buffer called escape-buffer is used to handle one of the deadlocked

packets. This transmission of the packet is suspended by temporarily escaping it to the escape-buffer. After the other deadlocked packets were sent, the suspended transmission restarts by restoring the escaped packet. Because of escaping and restoring the deadlocked packet, this technique does not need to take the channels of other normal packets. When deadlocked packet is detected then all flits of the deadlocked packet are temporarily saved to escape buffer, and the entire switch and channel connection are released. This procedure is called as escape as shown in Figure 14(b1) and Figure 14(b2) respectively. For the escaped packet at each router, the temporarily released channel is restored to its previous form based on the previous state of the saved switch-connection and the released channel-connection as a result, the escaped packet is restored, and selecting a channel of the routing thus restarts. This procedure is called as restoration as shown in Figure 14(b3).

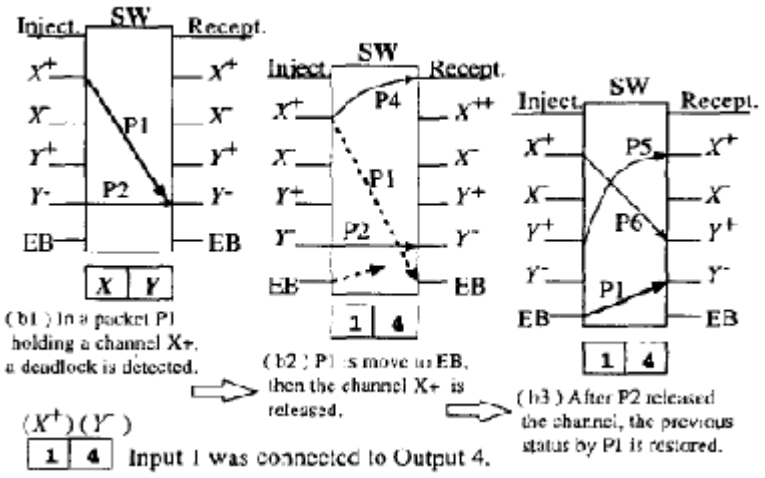

Figure 14. Switch connection state

Chapter 3 Problem Statement

Interconnection networks being lossless networks are more prone to deadlocks while probability of occurrence of deadlock is rare; there deadlock recovery strategies are used where some resources are de-allocated and granted to some other packets. A lot of work has already been done in designing more efficient routing strategies to recover from routing dependent deadlocks. Like in disha sequential technique extra deadlock buffer is added due to which less blocking delays occurred at intermediate nodes. In compression-less routing strategy, recovery is based on heuristic time–out mechanism without the use of virtual channels while on the other hand deadlocked message are always removed from the network in case of software based routing. ZOMA is efficient in terms of low hardware resource requirements as compared to other techniques. The other suspensive deadlock recovery technique requires a small amount of exclusive escape buffer at each router for handling one of deadlocked packets where the transmission of the packet is suspended by temporarily escaping from the escape-buffer but deadlock also arises due to the interactions and dependencies created at network end points between different message type called message dependent deadlocks. Whereas m-Disha architecture provides a solution of recovering from such type of deadlocks but not giving any efficient token management system.

Objectives of the study

1) Study the various deadlock avoidance and recovery techniques for routing dependent and message dependent deadlocks.

2) Propose an algorithm for token management for message dependent deadlock recovery architecture.

3) Implement the mDisha architecture and proposed algorithm in Java.

4) Analyze the performance of existing and proposed Architecture.

Chapter 4 Token Management in Message Dependent Deadlocks

Message dependent deadlock is another form of deadlock that can occur in interconnection networks due to the circular wait for situation .Unlike routing dependent deadlock, message dependent deadlock is due to the interaction and dependency among different message types at network end nodes. This chapter describes the token management in message dependent deadlock recovery technique by describing what message dependent deadlock is , how deadlock can be recovered and how the token can be managed.

4.1 Message Dependency

The exchange of various types of messages i.e. request and reply is pervasive in computer systems, including multiprocessor/multicomputer systems, client-server systems, and even uni-processor systems—I/O operations occur between processor and memory. At any given end node there can be coupling between the two messsage types i.e. the reply generated by the destination is directly coupled to the request received by the destination [25]. In shared-memory systems, request and reply message types used to exchange information between communicating entities. As the coupling between message types is transferred to network resources due to the limitness of resources along the message path inside each node additional dependencies on network resources are created, called as message dependencies. Message dependencies occurring at network endpoints may prevent messages from dropping at their destinations. When message dependencies are added to the complete set of resource dependencies, knotted cycles may form along escape resources, resulting in possible deadlock. The deadlocks arise from these phenomena called as message dependent deadlocks. Partial order relation ($\prec$) between two message types m1 and m2 is possible if and only if m2 can be generated by a node receiving m1.

Message type m2 is said to be subordinate to m1, and all message types subordinate to m2 are also subordinate to m1. The number of message types within a message dependency chain called as the chain length. The final message type at the end of the message dependency chain called as a terminating message type.

4.2 Message Dependent deadlocks

Resources along the message path inside each node and in the network are finite .Message dependencies occurring at network endpoints may prevent messages from sinking at their destination. When message dependencies are added to the complete set of resource dependencies knotted cycle may form along the escape resources, resulting in deadlock called as message dependent deadlocks.

One of the necessary conditions to have message dependent deadlock is that heterogeneous messages types are allowed to occupy the same subset of network interface resources.

A case has been taken where the resources in network interfaces are freely shared among heterogeneous messages .To simplify, all network interfaces considered as either input or output message queues. Assume only two message types in the system i.e. request and reply type as shown in Figure 15 which shows message-dependent deadlock represented by a resource dependency graph in which two nodes, node A and node B are each sending request messages to one another and expecting to receive reply messages over a network free from routing-dependent deadlock. If the arrival rate of request messages exceeds the consumption rate, a backlog starts to form at the input message queue IQA at node A. After a while, the backlog propagates backward at the output message queue OQB at node B [25, 26]. The backlog reaches the input message queue IQB at node B with the help of message type dependency, and, further, to the output message queue OQA at node A across the network. At this point, a deadlock is formed as no buffer space can be freed for reply messages needed by both nodes to continue execution. Main Limitation of this architecture is that both the nodes are waiting to receive acknowledgement and all the queues are full so no buffer is available to send the acknowledgement due to this deadlock occurs.

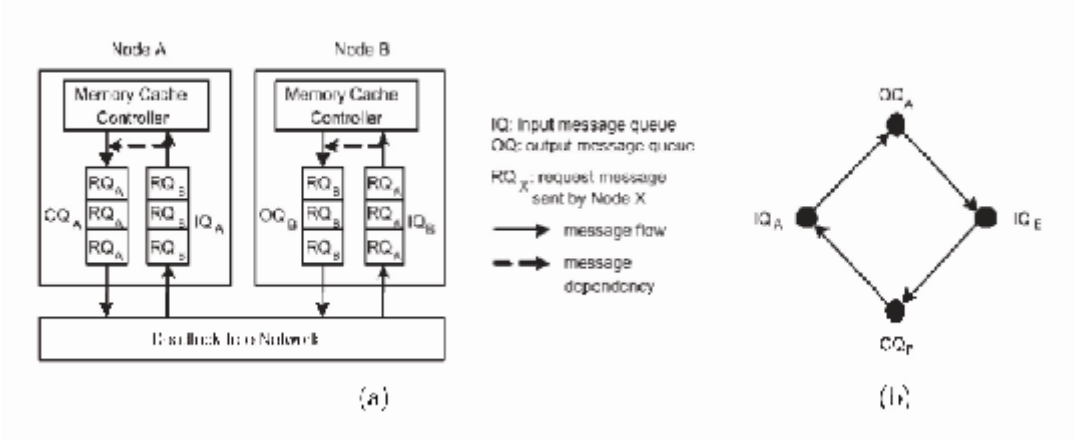

Figure 15: (a) A simple example of a message-dependent deadlock occurring is between two nodes connected by a network free from routing deadlock. (b) The corresponding dependency graph for resources at network endpoints when messages of different types share injection and reception queue resources.

4.3 Handling Message Dependent Deadlocks

4.3.1 Strict Deadlock Avoidance

Message-dependent deadlock can be avoided while allowing cyclic dependencies on escape resources by requiring some subset of escape resources to be large enough such that they can never become fully occupied. Also they can be strictly avoided by enforcing routing restrictions on network resources used to escape deadlock, such that all dependencies on those resources, including message dependencies, are acyclic [16]. Since sufficient resources and/or routing restrictions on a set of resources always prevent the formation of deadlock, these techniques for handling deadlock are said to be based on deadlock avoidance. The avoidance technique based on sufficiency of escape resources can be implemented by providing enough buffer space in each node's network interface message queues to hold at least as many messages as can be supplied, as in [25,26,27]. Although simple to implement, this technique is not very scalable since the size of the message queues grows as O (P ×M) messages, where P is the number of processor nodes and M is the number of outstanding messages allowed by each node. The avoidance technique based on routing restriction is more scalable and more commonly used. One way of guaranteeing acyclic dependencies on escape resources is to provide logically

independent communication networks for each message type, implemented as either physical or virtual channels. The partial ordering on message dependencies defined by the communication protocol is transferred to the logical networks so that the usage of network resources is acyclic.

Figure 16. Illustrates this for the example of a four node ring system, which allows a message dependency chain length of two, i.e., request and reply, but no greater.

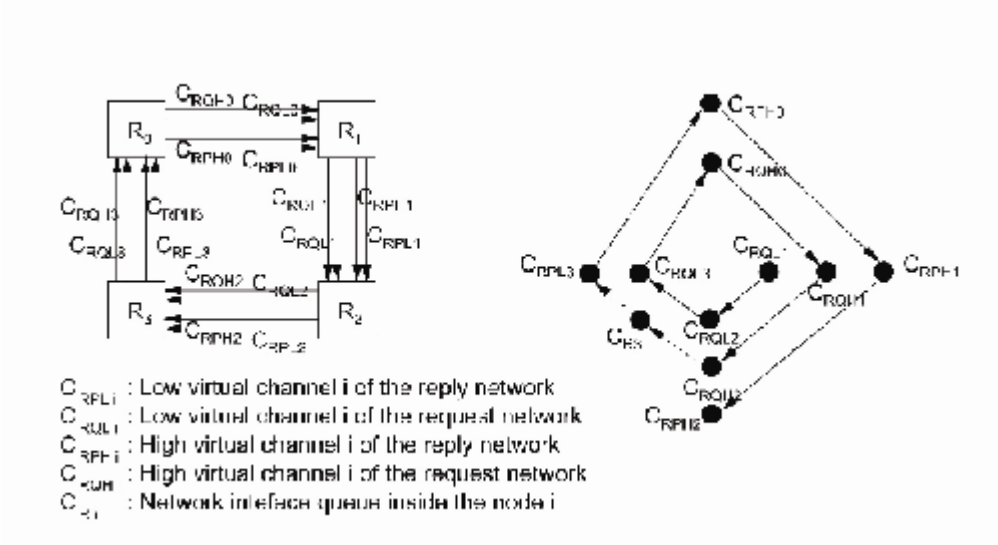

Figure 16:(a) Separation of request and reply networks avoids cyclic dependencies in the channel dependency graph, but it reduces channel utilization. (b) The channel waits-for graph for the case where R1 and R0 are the requester and responder, respectively.

With this technique, the size of network resources does not influence deadlock properties, but at least as many logical networks are required as the length of the message dependency chain[25]. Such partitioning of network resources decreases potential resource utilization and overall performance, particularly when message dependencies are abundant and resources (i.e., virtual channels) are scarce.

In general, the available virtual channels that can be used is limited to (1+ (C/L–Er)) of the C virtual channels implemented, where L is the message dependency chain length, Er is the minimum number of virtual channels required to escape from routing-dependent deadlock for a given network, Em = L × Er is the minimum number

of virtual channels required to escape from message-dependent deadlock for a given network, and $C \geq E_m$. Channel availability can be increased if all channels other than the minimum number required to escape from message- dependent deadlock are shared amongst all message types. That is, the upper limit on virtual channel availability is increased to $(1+(C - E_m))$. Main disadvantage of avoiding deadlock is by disallowing cyclic dependencies on escape resources is the number of partitioned logical networks required. It is possible to reduce the number of partitions by allowing different message types to use the same logical network and removing message(s) from cyclic dependencies only when a potential deadlock situation is detected. Since a detection mechanism and recovery action are required to resolve the potential deadlock situation, this technique for handling message-dependent deadlock is based on deadlock recovery

4.3.2 Deadlock Detection and Recovery

There are many ways in which potential message-dependent deadlock situations can be detected and resolved. They can be detected at a node's network interface when three conditions are met, as in [25, 26]:

a) Both the input and output queues allocated to a message type and its subordinate message type fill up beyond a threshold value,
b) The message type at the head of the input queue is one that generates a non-terminating message type, and
c) The first two conditions continue for more than a threshold time-out period.

Detected deadlocks can be resolved by killing and later re-injecting messages called regressive recovery, deflecting messages out of resources involved in cyclic dependency by converting them from a non-terminating type to a terminating type called deflective recovery, or by progressively routing messages using resources along a path that is guaranteed to sink called progressive recovery.

4.4 mDisha

After describing how the deadlock occurs when network resources are finite and dependency between different message types occurs. This architecture overcomes the limitation of previous one by adding extra buffer but still there is no provision for handling messages in this architecture.

Assumptions taken for mDisha technique:

1. When the memory controller needs to send a response message to n individual destination nodes, it is assumed that n messages are generated instead of one multicast message.

2. When the memory controller processes and removes a message from the input message queue of the network interface only if output message queue has sufficient space to receive the subordinate messages produced.

3. When the memory controller of a node has an output message which requires one or more subordinate messages to return to the node, it pre-allocates all the resources in order to drop the subordinate messages successfully.

In mDisha, token must visit all network interfaces attached to each router node, and a deadlock message buffer must also be provided in each network interface [25, 26]. The size of the DMB is determined by the minimum unit of information on which end-to-end error detection/protection is performed. For packet level the deadlock buffers need to be at least packet-sized. This is also the minimum size of the network interface input and output message queue. Network interface input/output queues at network endpoints can be independent of message type because all the network resources can be completely shared by all message types. When resource allocation and routing restrictions to any network are relaxed, it maximizes the resource utilization but does not strictly prevent message-dependent deadlock from potentially forming. So this alone is not sufficient for appropriately handling message-dependent deadlocks. It is possible that the destination node has no available buffer space to drop the packet, which forces the packet to stay in the DMB until the input buffer can accept the packet, which is not guaranteed and if the recovery operation ends without dropping the packet to an input buffer, other recovery operations may not successfully complete in that case DMB holding that packet. This introduces the possible deadlocks along with recovery resources.

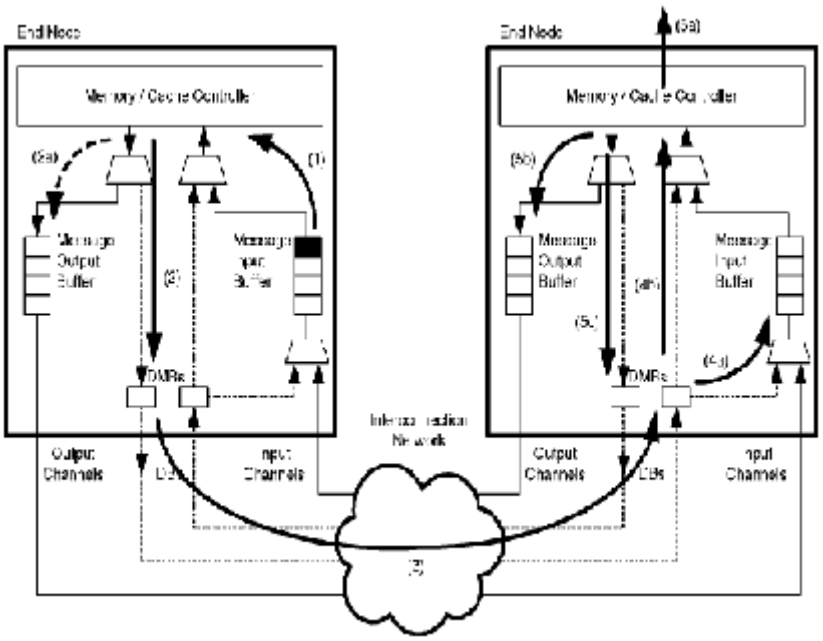

Figure 17. Flow of messages undergoing [16]

Block diagram of mDisha recovery procedure is shown in Figure 17. After the potential message dependent deadlock can be detected at network interface. The circulating token is captured at the network interface and the non terminating message type at the head of the input queue mi is processed by the memory/cache controller at step (1). The generated subordinate message type mj which is partially ordered to mi is put into the DMB at the network interface at step (2) and routed over the recovery lane to its destination node DMB with the token at step (3) this satisfies when the output queue is full at source node. The source and the destination of the message are referred to as token sender and token receiver respectively. To prevent the resources for being involved in deadlock recovery, the message arriving at the DMB of its destination should be ensured to exit the DMB. Hence if the input message queue at the destination is not full, the message drops to the queue at step (4a) and the recovery operation ends by returning the token to the token sender. Else if the queue at the destination is full, then the memory controller is preempted after it completes its current operation and processes the message at step (4b). If the processed message mj is a terminating message type goes out from the process at step (5a) or if it is non-terminating and the newly generated subordinate message mk which is partially ordered to mj is put into the output queue at step (5b) and the token is returned back to

the token sender over the DB lane. Otherwise, the recovery process continues by putting the newly generated message type mk into the DMB at that node for routing over the recovery lane at step (5c). In this case, the token will be reused to deliver mk through the set of DBs, where the node becomes a token sender with respect to the subordinate message type. This recovery process continues throughout the length of the message dependency chain until either the subordinate message is delivered to an output queue of the generating node or an input queue of the destination node or a terminating message type is eventually generated, which will be consumed by the memory controller or by an input queue[25,26]. Each token receiver returns the token to the associated token sender. If the token returns to the node which initiated the recovery process and the node has no more messages to deliver with the token, the deadlock recovery process ends. At this point, the token is released for re-circulation. This mDisha technique maximizes routing freedom and resource utilization by allowing messages to freely use all resources except minimal deadlock recovery resources. Whereas the limitation of this scheme is that no reliable token management system i.e. when token lost then no provision from recovering it.

So, this work proposes an algorithm for token management in the architecture which shows that if all the input and output queue of sender and receiver becomes full due to the difference in speed of sending and receiving of packets and also because of the controller which is busy in generating their own messages so that will not receive packets then one message path which is given as an alternate path will always works because it preempts the controller. This algorithm gives the solution of the problem that when token is lost in the network then it regenerate and retransmit the packet along the same path and also all 4 nodes are connected in a ring.

4.5 Algorithm for managing token in m- disha deadlock recovery technique:-

Initialization: (S, D, IO, OQ, DMB)

/*

S is the source node

D is the destination node

IQ is the Input Queue

```
        OQ      is the Output Queue
        DMB     is the deadlock buffer
    */
    {
        For (i = S to D)
    /* i is the node of n/w */
        {
            If (size (IQ (i)) == max)
            {
                Print "Deadlock";
                Call safe path (DMB (IQ (i)));
            }
            Else
            {
                Message pass;
            }
        }
        If (Message type (IQ (i)) == packet)
        {
If (Message type (OQ (i)) == acknowledgement)

        {
            If (Arrival time(S) > hop time)
            {
                Return false;
                Print "Acknowledgement Lost & retransmit ";
            }
        }
```

Algorithm for token management based on events:

Initialization:

For i = 1 to n

{

Input Queue (i) = NULL;

Output Queue (i) = NULL;

DMBI (i) = NULL;

Switch (event)

{

Event 1:- Circulating token over network

While (! token call)

{

Current token = token (i);

If (i = n)

Then

```
            i =1;
            }
        }
```

Event 2:- Node Ni handle request

```
        While (tokencall = true)
        {
            If (Input Queue (i) ! = NULL) MC
            (i) = Input Queue (i) [front]; Else if
            (Output Queue (i) = FULL)
            {
                Print "Deadlock";
                Call event 5;
                Send to DMBI (i) +1;
            }
        }
```

Event 3:-Node Ni receive request for message m

```
        If (Input Queue (i) =NULL)
        {
            Input Queue (i) +1 [Rear] = DMBI (i);
            Call event 5;
        }
        Else
            MC (i) = DMBI (i);

        If (Terminating message)
        {
          If (Transmission time (i) > Hop Time)
            {
```

```
            Return “retransmitting “;
            Call Event 5;
            }
            Send token back to sender;
            Exit;
        }
        Else
        {
            If (Output Queue (i) ! =NULL)
            Output Queue (i) [Rear] =MC (i);
            }
            Else
            {
            DMBO (i) =MC (i);
            }
        }
```

Event 4:-Node Ni generates its own request

```
        While (tokencall == true)
        {
            If (Output Queue (i) = NULL)
            {
            OutputQueuei [Rear] = MC (i);
            }
        }
```

Chapter 5 Results and Findings

5.1 Two Nodes Simulation

Case 1:

Communication between 2 nodes where when packet loss then no provision for handing the packet as shown in Figure 19.

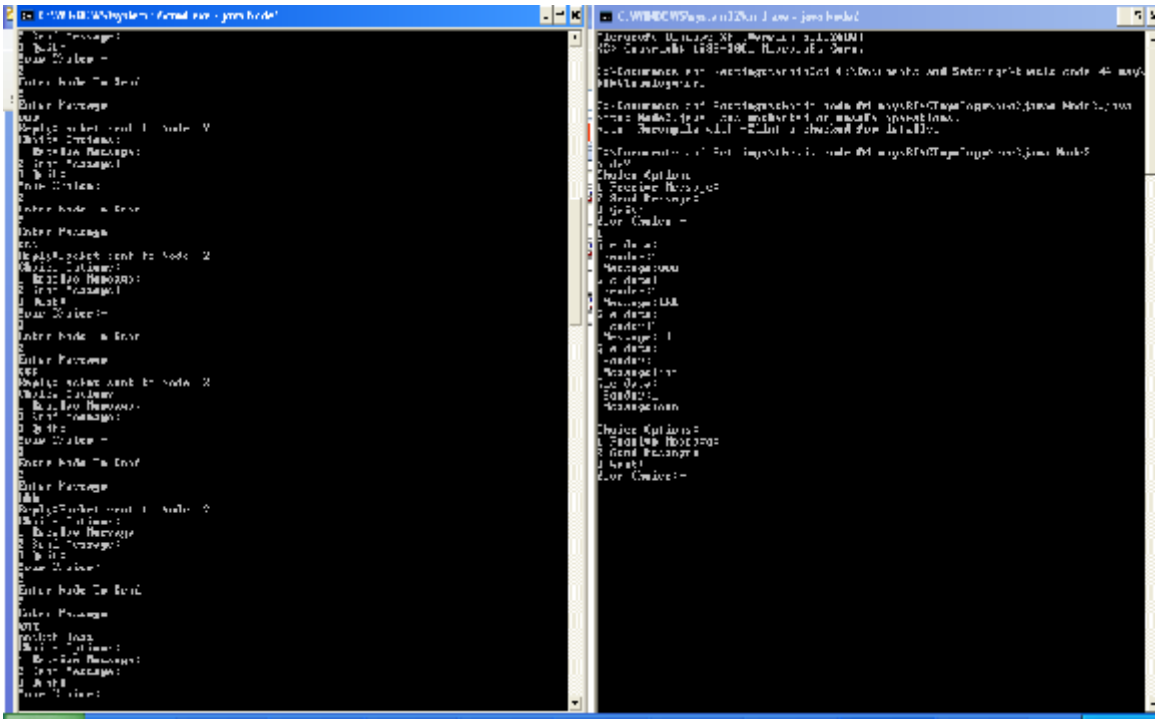

Figure 19. Communication between 2 nodes and showing packet loss

5.2 Different cases in case of 4 node Architecture

Case 1:

This shows in Figure 20 that communication between different node where all the four node send and receive messages as well as acknowledgement.

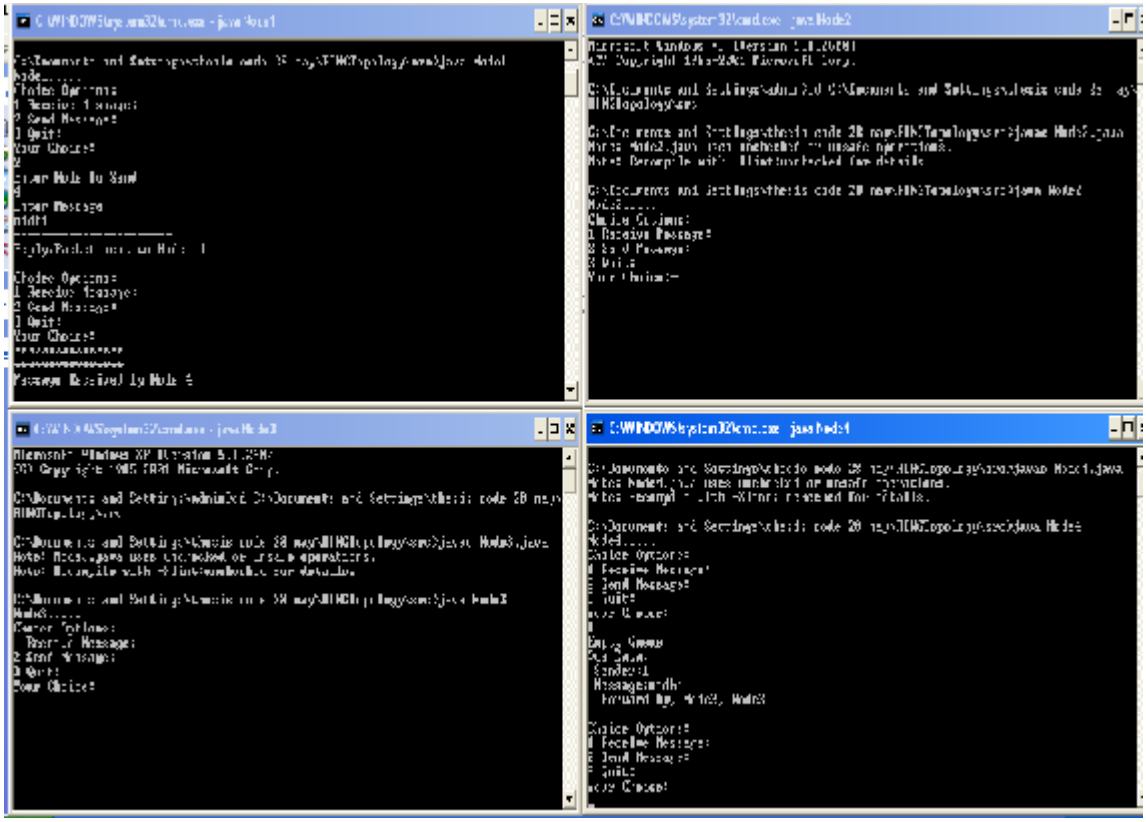

Figure 20. Communication between different nodes

Case 2:

This case shown in Figure 21 that no more messages are sent to a particular node because that node will not receive messages and the message will send through alternate path (DMB). Node 1 sending messages to node 4 but that node will not receive messages of any reason then node 1 and 4 comes in deadlock then in that case messages will be send through alternate path which per-empts controller to stop his work any forces to receive messages.

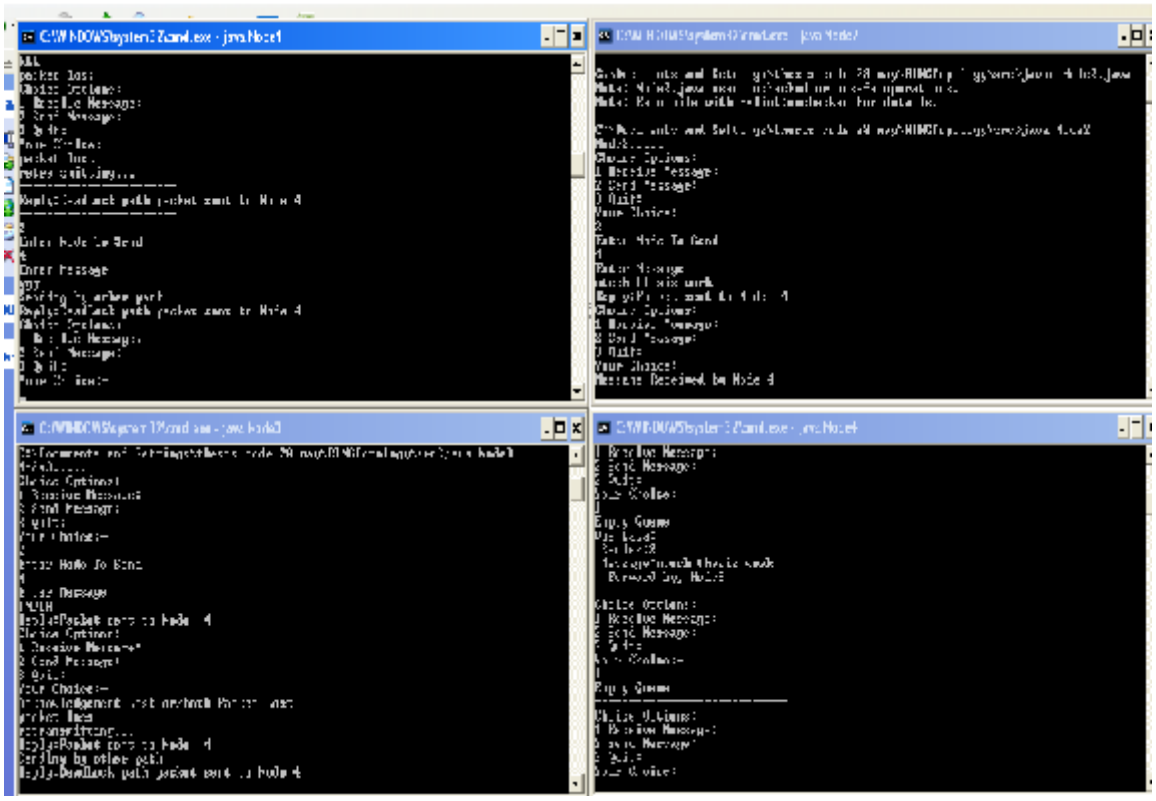

Figure 21. Sending packet through alternate path

Case 3:

Figure 25 shows that node 1 sends messages to node 3 and node 4 then some packet or acknowledgement will be lost in the network due to congestion and also node 2 send messages to node 4 then in that case sending node again re-transmit the packet till it will not received the acknowledgement by taking an account of arrival time and hop time.

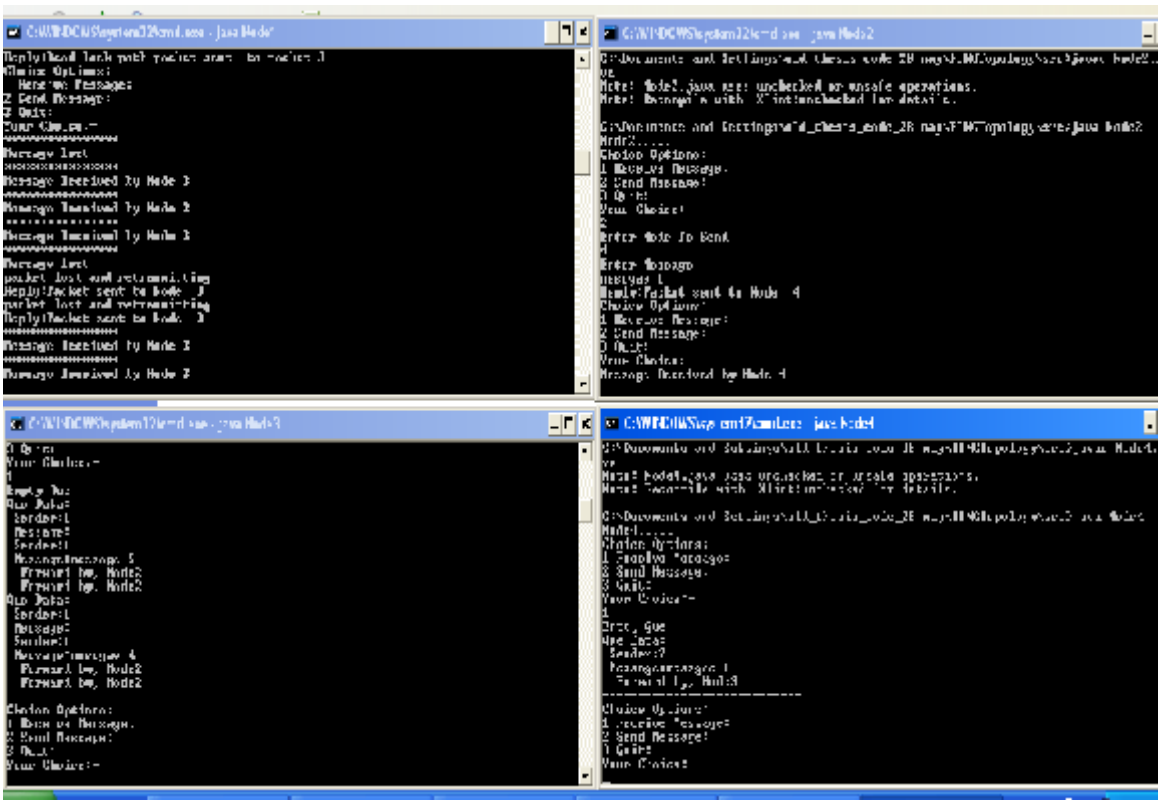

Figure 25. Showing two Packets or Acknowledgement lost and retransmitting

Case 4:

When intermediate node is in deadlock then messages send from node 1 to node 4 this cannot be possible while normal path but done through alternate path as shown in Figure 26.

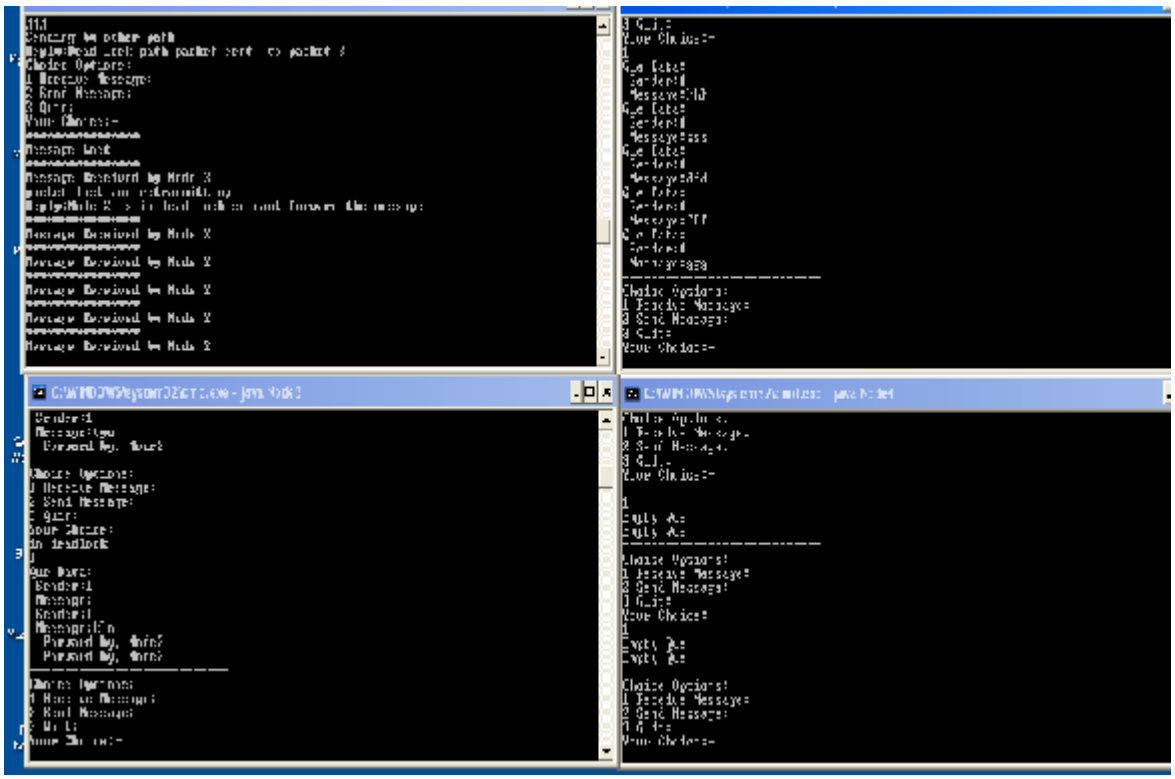

Figure 26. Showing deadlocked intermediate node and successful sending of packet to its next node through alternate path

These cases shows that when any packet or acknowledgement lost in the network then that particular node regenerate that packet and resend it till the acknowledgement will not be received no matter that, node is in deadlock or not because an alternate path always work because it pre-empts the controller. If any node is in deadlock then packet successfully reaches to its destination through alternate path and acknowledgement also received successfully.

Chapter6 Conclusion

6.1 Conclusions

i. Message –dependent deadlocks occur due to the interactions and dependencies among heterogenous message types at network end-points. While routing dependent deadlocks occur due to the cyclic dependencies only on network resources, message-dependent deadlocks span not only between network endpoints but also across endpoints.

ii. Parallel computer systems must take steps to handle message-dependent deadlocks by either avoiding or recovering from them if detected. mDisha technique relaxes restrictions on routing freedom and message buffering at network end nodes, allowing much more efficient routing of packets and handling of message-dependent deadlocks, particularly when network resources are scarce but no provision of token recovery when lost.

iii. The proposed algorithm sufficiently handle token in message dependent deadlocks.

iv. Simulation shows that token lost in the network is considered when acknowledgement is not received in a pre-defined arrival time. In that case node resend the token again to its destination till it is not received.

6.2 Summary of Contributions

a) Study the various deadlock Avoidance and Recovery Techniques which shows that routing dependent deadlocks and message dependent deadlocks can be avoided but by providing different virtual channel path to different types of messages which is not possible. Hence in those cases deadlock can be recovered.

b) The issue in message dependent deadlock is that the token will not be managed so this work proposes an algorithm to manage token.

c) Implement this algorithm in Java which shows the lost of packet or acknowledgement in that case regenerate the packet and retransmit it.

d) Analyze existing and proposed Architecture which shows that when packet or acknowledgement lost in the network then not possible to recover from it but in proposed how the lost acknowledgement will be recovered it by assigning per-defined time.

6.3 Future Research

1. This architecture can handle only one deadlocked message at a time so no provision for handling this. This can be done by concurrently handle deadlocked messages so without increasing the hardware cost i.e. increasing number of buffers we can handle this with using only single buffer.

2. The study can be extended to other networks like Network-on-chips.

REFRENCES

[1] Al-Awwami H., Obaidat M.S., and Al-Mulhem M., "ZOMA: A preemptive Deadlock Recovery Mechanism for fully Adaptive Routing in Wormhole Networks", *IEEE Transactions on Computers,* vol. 50, no.8, pp. 811-823, 2001.

[2] Al-Awwami Z.H., Obaidat M.S. and Al-Mulhem M., "A New Deadlock Recovery Mechanism for Fully Adaptive Routing Algorithms ",in Proceedings of the *IEEE International Performance ,Computing and Communications Conference*, vol.9, no.2 , pp. 132-138, 2000.

[3] Anjan K.V., Pinkston T.M. and Duato J., "Generalized Theory for Deadlock-Free Adaptive Wormhole Routing and its Application to Disha Concurrent", *in Proceedings of the 10th International Parallel Processing Symposium*, 1987, pp. 815-821.

[4] Anjan K.V. and Pinkston T.M., "Disha: An Efficient, Fully Adaptive Deadlock Recovery Scheme", *in Proceedings of the 22nd annual international symposium of Computer Architecture,* vol. 23, Issue 2, May 1995.

[5] Atagoziyev M., "Routing Algorithm for on chip networks", in Dissertation of The Graduate School of natural and applied sciences of Middle East Technical University, December 2007.

[6] Choi Y. and Pinkston T.M., "Evaluation of Crossbar Architectures for Deadlock Recovery Routers", *Journal of Parallel and Distributed Computing*, vol. 61, no. 1, pp. 49–78, January 2001.

[7] Dally W. and Seitz C., " Deadlock-free Message Routing in Multiprocessor Interconnection Networks", *IEEE Transactions on Computers*, vol.36, no. 5, pp. 547–553, May 1987.

[8] Duato J., Yalamanchili M.S. and Ni L., "Interconnection Networks: An Engineering Approach", Morgan Kauffman publisher an imprint of Elsevier Science: 2002.

[9] Duato J. and Pinkston T.M., "A General Theory for Deadlock-Free Adaptive Routing Using a Mixed Set of Resources", *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 12, pp. 1219–1235, Dec 2001.

[10] Hansson A., Goossens K., Radulescu A., "Avoiding Message-Dependent Deadlock in Network –Based System on Chips", *VLSI Design*, vol. 2007, Article 1D 95859, pp. 10 , 2007.

[11] Holt R.C., " Some Deadlock Properties on Computer Systems", in *ACM Computer Surveys,* vol. 4, no. 3, pp. 179–196, September 1972.

[12] Jurczyk M., "Interconnection Networks for Parallel Computers", IEEE Computers, vol. 20, no. 6, pp. 1343-1350, 1997.

[13] Khonsari A., Farahani A. and Ould-Khaoua M., "Disha: A Performance Model of a True Fully Adaptive Routing Algorithm in k-ary n-Cubes*", in Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, & Simulation of Computer & Telecommunications Systems*, Texas, USA, 2002.

[14] Khonsari A., Shahrabi A., Ould-Khaoua M. and Sarbazi-Azad H., "Performance comparison of Deadlock Recovery and Deadlock Avoidance Routing Algorithms in Wormhole-switched Networks", *IEEE Computer Digital Technology*, vol. 150, no. 2, pp. 1297-1304, March 2003.

[15] Kim J.H., Liu Z. and Chien A.A., "Compressionless Routing: A Framework for Adaptive and Fault-Tolerant Routing", *IEEE Transactions on Parallel and Distributed Systems*, vol.8, no.3, pp. 229-244, March 1997.

[16] Lankes A., Wild T., Herkersdorf A., Sonntag S. and Reinig H.M. , "Comparison of Deadlock Recovery and Avoidance Mechanisms to Approach Message Dependent Deadlocks in on-Chip Networks", *in Proceedings of the 2010 Fourth*

ACM/IEEE International Symposium on Networks-on-Chip, IEEE Computer Society ,Washington DC, USA, 2010.

[17] Lee S., "Deadlock Detection and Recovery for True Fully Adaptive Routing in Regular Wormhole Networks", *Journal Of Information Science And Engineering* vol 25, no. 2, pp. 465-479, 2009.

[18] Lysne O., Reinemo S.A., Skeie T., Solheim A.G. and Sodring T., "Interconnection Networks: Architectural Challenges for Utility Computing", IEEE *Computer Society*, vol. 41, no. 9, pp. 62-69, Sept. 2008.

[19] Martinez J.M., Lopez P., Duato J. and Pinkston T.M., "Software-Based Deadlock Recovery Technique for true Fully Adaptive Routing in Wormhole Networks", in Proceedings of the IEEE *International Conference on Parallel Processing*, pp. 8108-8186, 1997.

[20] Mohapatra P., "Wormhole Routing Techniques for Directly Connected Multicomputer Systems", Journal of *Parallel and Distributed Systems,* vol. 30, no. 3, pp. 600-607, December 1994.

[21] Pinkston T.M., Warnakulasuriya S., "On Deadlocks in Interconnection Networks", *in Proceedings of the 24th annual International symposium on Computer Architecture,* vol. 25, no. 2, 1997.

[22] Pinkston T.M., "Flexible and Efficient Routing Based on Progressive Deadlock Recovery", *IEEE Transactions on Computers*, vol. 48, no. 97, pp. 649-669,1999.

[23] Seo S.W. and Feng T., "The Composite Banyan Network", *IEEE Transactions on Parallel and Distributed systems*, vol. 6, no. 10, pp. 1043-1054, October 1995.

[24] Silla F., Robles A. and Duato J., "Improving Performance of Networks of Workstations by using Disha Concurrent", in *Proceedings of International Conference on Parallel Processing*, pp. 80-87, Aug. 1998.

[25] Song Y.H., "Architectural Support for Efficient Utilization of Interconnection Network Resources", in Dissertation of Faculty of the Graduate School University of Southern California, August 2002.

[26] Song Y.H. and Pinkston T.M., "A Progressive Approach to Handling Message-Dependent Deadlock in Parallel Computer Systems", *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 3, pp. 259-275, March 2003.

[27] Song Y.H. and Pinkston T.M., "Efficient Handling of Message Dependent Deadlocks", *in Proceedings of the 15th International Parallel and Distributed Processing Symposium,* pp. 80, April 2001.

[28] Takabatake T., Kitakami T. and Ito H., "Escape and Restoration Routing: Suspensive deadlock Recovery in Interconnection networks", *IEEE Transactions on Computers*, vol.50, no. 8, pp. 811-823, 2001.

[29] Wang X., Liu P., Yang M., Jiang Y., "Resolving Deadlocks for Pipelined Stream Applications on Network-on-Chips", *in Proceedings of the 4th IEEE International Conference on Computer Science and Information Technology* , vol.18 , no.5, pp. 1436-1438, 2010.

[30] Warnakulasuriya S. and Pinkston T.M., "A Formal Model of Message Blocking and Deadlock Resolution in Interconnection Networks", *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 3, pp. 212-229 , March 2000.

Paper Published

1) Garg N. and Agarwal R.R," A Review of deadlock recovery techniques in interconnection networks" Journal of Computer Science and Engineering, Volume 6 , Issue 1 , 2011.

2) Garg N. and Aggarwal R.R,"An Efficient Algorithm Token Management Algorithm for Message Dependent Deadlocks Recovery Architecture", Journal of Advances in Information Technology Special issue on Advanced Algorithm (**Communicated).**