Fault Tolerant Distributed routing algorithm for Network-on-Chip

***Abstract*—The recent trends in high-performance computing are currently switching to multi-core architectures and this paradigm is a highly required change as performance with single core processors has hit the power consumption wall. For communication between the multiple processors, Network-on-Chip(NoC) is emerging as an advantageous and promising alternative to the traditional bus based communication. However, manufacturing faults or hardware failures may affect the regularity of an NoC topology by introducing link failures. In such circumstances, efficient routing becomes a challenge. Although work in the off-chip domain provides solutions using routing/forwarding tables at switches, this does not scale in the on-chip domain due to its memory requirements. Though, a few routing strategies like LBDR [1] and Fault-Tolerant XY algorithm take care of a few faults, these do not tackle varying injection rates which may severely affect the throughput in the network. On the other hand our proposed algorithm sees to it that the packet, tries to move around the faults in a non-minimal path and finally reach its destination. The simulation results demonstrate the advantage of the proposed routing algorithm in terms of overall throughput and percentage of packet loss rate as compared to those of LBDR and Fault-Tolerant XY routing algorithm.**

Keywords*—*Network on chip, Fault Tolerance, Non-minimal Routing.

I. Introduction

As the performance and power scalability of monolithic microprocessor cores is running into physical barriers, industry is shifting to multi-core architectures for designing high performance microprocessors. Chip architectures with such a large number of cores require a high performance on-chip interconnect for efficient communication between cores and with cache blocks and memory controllers. Current chip implementations are based on bus or ring structures. However, as the number of cores increases, such interconnect fabrics become the bottleneck of the system, as they do not scale. For highly integrated systems, on-chip interconnects(NoC) are likely to provide the needed bandwidth and also meet the stringent latency requirements. Also, it has been proved that these interconnects deal with the communication scalability challenges while meeting tight power, area and latency design constraints.

Fault-tolerance is becoming a major concern in NoCs and CMPs due to the physical mechanisms that make designing on nanoscale technologies a challenging task. While transient faults cause reliability issues, these may be addressed by means of physical or circuit level design techniques, manufacturing imperfections or hardware failures result in defective IP cores, wires or switches which may hamper operation of the whole system. A regular NOC topology may become irregular due to the above mentioned reasons. While choosing a regular topology for NoC, A 2D mesh topology is usually preferred for the chip layout because of its regularity and matching with the 2D silicon surface. However, hardware failures owing to manufacturing, wear-out, aging and other reasons may disrupt the regularity of a 2D mesh.

Since most of the chip area is still fully functional, it has to be ensured that the network is able to work also with the new irregular topology, which has implications at least on routing design. Sustaining routing under these circumstances becomes a major issue. Routing in a topology after the introduction of faults or link failures can be dealt with the help of routing tables. However, due to the use of forwarding tables, memories do not scale as much as logic in terms of latency, power and area, thus proving them impractical for large NoCs. The Existing routing techniques that have been proposed to handle link failures are either a routing table based or require a complex constrained switch design with additional hardware overhead. Thus designing an efficient routing strategy to address the link failures in the network still remains an open issue. This motivates us to design an efficient and distributed routing algorithm which handles all the irregularities generated in a 2D mesh as a result of link failures. To achieve the desired objectives, we aim to develop an algorithm which drastically reduces the packet loss rate thus simultaneously increases the throughput. The proposed algorithm tackles nodes with one or two link failures. The main advantage of our proposed work would be the non-usage of any form of routing tables or any excessive additional hardware. Additionally it would be deadlock free because of the routing restrictions imposed. The algorithm aims to target the performance metrics namely: Latency, Throughput, Packet Loss Percentage and Power. In NoC, latency and throughput are the most widely used performance evaluation statistics. Latency depicts the delay (number of clock cycles) in transmitting a packet from a given source to desired destination. Throughput is expressed in terms of the number of flits received per unit time interval. The desired outcome of any efficient routing algorithm would be high throughput and low latency.

II. Related Work

Solutions based on routing tables are flexible in dealing with topologies with link failures but these suffer from scalability, as the memory overhead is huge and this makes the cost of the tables shoot up. This paved way for some work on the reduction of memory requirements for NoCs to be done. Some of the approaches with the above mentioned objective would be Interval routing and the FIR method which is an extension of interval routing. While both of them work well for regular topologies, they are not applicable for irregular topologies.

On the other side, low cost deterministic solutions such as Flexible Dimension Order Routing or other adaptive approaches offer limited flexibility and coverage as compared to routing using forwarding tables. LBDR is the logic based implementation of distributed routing algorithms with minimum logic and no usage of routing table. But the major drawback would be that LBDR can be applied to a combination of topologies and routing algorithms with particular characteristics. An example would be that LBDR works only for those topologies in which all the end-nodes communicate with the rest of nodes through a minimal path defined in the original mesh topology as shown in Fig.1.

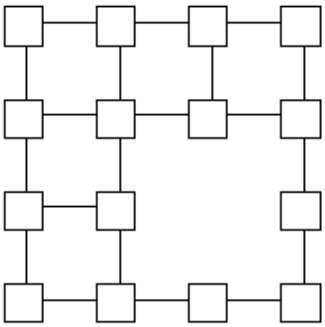

Fig. 1. Non-minimal topology -Not supported by LBDR.

uLBDR is a logic based distributed routing solution to handle irregularity without the requirement of any routing table. Different flavours of uLBDR provide different levels of coverage but these impose several restrictions on the switch design.

III. Proposed Algorithm Description

In this paper our aim is to devise a fault-tolerant routing strategy that can handle link failures that have been incorporated in a regular 2D mesh as shown in Fig.2.

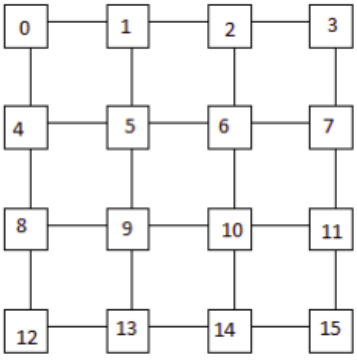

Fig. 2. 2D 4x4 Regular mesh topology with 16 nodes.

In a regular 2D mesh, each node/tile is connected to its neighbouring nodes by four different channels which are bi-directional. These channels have ports through which the connections are established. The four ports are labeled as N for North port, E for East port, W for West port and S for South port. Each switch or router is identified by a unique router id and two variables namely Xcurr and Ycurr which specify the row and column to which the router belongs. Also,the messages being sent/delivered are routed using X and Y coordinates, with the assumption that the X and Y co-ordinates of the destination namely Xdest and Ydest are involved in the header.
We can observe that, $0 \leq$ Xcurr $\leq$ K-1.
Similarly, $0 \leq$ Ycurr $\leq$ K-1.
The following notations have been used in the description of our algorithm:
n - Total number of routers(switches)
K - Number of routers in a row or column
Xcurr - Current Router ID/K (current row)
Ycurr - Current Router ID%K (current column)
Xdest - Destination Router ID/K (destination row)
Ydest - Destination Router ID%K (destination column)
N - North
E - East
W - West
S - South
η_a- Neighbour of η in the direction a
$R_{rs}-$ Routing restriction bit depicting available routing options
$F_{rs}-$ Fault bit which indicates if a link in the r port of it's neighbour in the s direction is faulty
F_r- Fault bit which indicates if the connecting link through port r is faulty
$in_channel-$Channel/port through which a flit has been received by the current router
$currID-$Current router ID

Our proposed algorithm assumes the topology of the 2D mesh to be regular. Our proposed approach makes use of two types of bits namely routing bits and connectivity bits which are explained below. Routing bits are used to specify the restrictions that have been imposed on the packets due to the available routing options. Fault bits indicate the link failures at neighbouring nodes if any.
1. **Routing Bits** : Routing bits specify the routing restrictions which are imposed on the flits by the underlying algorithm. These restrictions prevent the flits from taking certain turns and thus help in avoiding deadlocks. If a routing bit $R_{rs}$is set to 1, it implies that a packet/flit routed in the R direction can continue to take a turn in the S direction at the next switch. Inorder to capture the available routing permissions allowed at a switch, the algorithm requires a usage of 8 routing bits.
2. **Faulty Bits** : These bits indicate if the links connecting them to their neighbouring nodes are faulty or not. This helps in choosing the switches that are connected without faulty links during routing. The F_{rs} bit is set to 1 if the link between η_r and η_s is faulty and is set to 0 otherwise.Also, the bit F_r indicates if the link connected through the R port of the current router is faulty or not. This bit can also be helpful in specifying the connections for the nodes which are present in the corners and borders of the mesh.We know that the in a mesh all nodes do not have the same number of neighbouring nodes. Routers in the corners have only 2 adjacent nodes whereas nodes present in the borders are connected to 3 of their neighbours. These differences in symmetry can be dealt with the help of the fault bits itself.For example, a node

present in one of the corners of the mesh has $F_N$and F_W set to zero indicating the loss of connection for a path in the north and west directions.

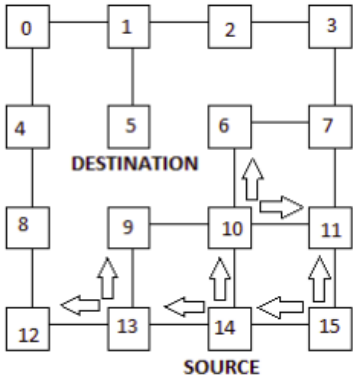

Fig. 3. Example 4x4 mesh with link failures.

The illustration of the above explained bits is shown with the help of an example 4x4 mesh with faulty links and routing restrictions as shown in Fig.3. The Fig.4. shows the corresponding routing bits for SR (Segment based Routing) for the mesh with link failures. The links between the nodes 2-6,4-5,5-6,5-9 and 8-9 are faulty and thus have been removed for better understanding as shown in the Fig.3. Depending on the link failures in the example figure, the fault bits in the table have been set accordingly. For example, at switch 9, the links through it's north and west ports are fault and hence F_N andF_W are set to 1 in the table. Similarly, the bits $F_{N,E}$ and$F_{N,W}$ have been set to 1 indicating the faulty links between 5-6 and 4-5 respectively. As mentioned above, the routing bits are set according to Segment based routing after taking the faulty links into consideration. For example, the routing bits $R_{W,N}$ and $R_{S,W}$ at switch 11 have been set to 1 indicating that if a flit takes the West port it cannot turn North in the subsequent move. Similarly, if a flit at switch 11 takes the South port it cannot turn west in its subsequent move.

ID	F_N	F_E	F_W	F_S	F_{NN}	F_{EE}	F_{WW}	F_{SS}	F_{NE}	F_{NW}	F_{EN}	F_{ES}	F_{WN}	F_{WS}	F_{SE}	F_{SW}	R_{NE}	R_{SE}	R_{EN}	R_{ES}	R_{WN}	R_{WS}	R_{SE}	R_{SW}
0	1	0	1	0	1	0	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
2	1	0	0	1	1	1	0	0	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
3	1	1	0	0	1	1	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
4	0	1	1	0	1	1	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
5	0	1	1	1	1	0	1	0	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1
6	1	0	1	0	1	1	1	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	0	1
7	0	1	0	0	1	1	1	0	1	0	1	1	1	0	1	0	1	1	1	1	1	1	1	1
8	0	1	1	0	0	0	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1
9	1	0	1	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0
10	0	0	0	0	1	1	1	1	0	1	0	0	1	0	0	0	1	1	1	1	1	1	1	0
11	0	1	0	0	0	1	0	1	1	0	1	1	0	0	1	0	1	1	1	1	0	1	1	0
12	0	0	1	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1
13	0	0	0	1	1	0	1	1	0	1	0	1	0	1	1	1	1	1	0	1	1	1	1	1
14	0	0	0	1	0	1	0	1	0	0	0	1	0	1	0	0	1	1	0	1	1	1	1	1
15	0	1	0	1	0	1	0	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1

Fig. 4. Table showing the Routing and Fault bits for the above example.

Algorithm

Our algorithm follows a level wise logic similar to the LBDR implementation. The first level of logic constitutes of a comparator which compares the co-ordinates of the current router with that of the destination router. The inputs to the comparator are Xcurr,Y curr which are the co-ordinates of current router and Xdest,Ydest which are the co-ordinates of destination router. The output determines the relative direction of destination router with respect to the current router by setting atmost two signals of N',S',E' and W' high. Also, it is important to note that both N' and S' can never be simultaneously high. The same applies to signals E' and W' as well. For example, if the output of comparator determines the values of S' and W' to be high, this indicates that the destination router is present in the South-West (SW) quadrant with respect to the current router. It is also important to note that, once both the current router and destination router co-ordinates become equal, none of the signals become high and the flit/packet is then forwarded to the local port. These flits are excluded from the second part of the logic. The comparator is shown in the Fig.5.(a)

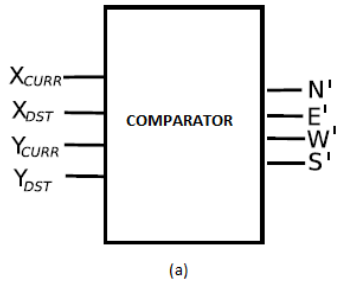

Fig. 5. Routing Logic First Level - Comparator

The second level of routing logic computes the suitable output port to be chosen based on a few conditions. We make use of the bits F_r,$F_{r,s}$,$F_{r,r}$and $R_{r,s}$ in the routing logic. Owing to the similarity in the logic for the port calculations, we present a detailed explanation of logic when the output port chosen is North(N).

The logic first computes the difference between the current router co-ordinates and destination router co-ordinates and stores them in variables dx and dy where,
dx= Xdest-Xcurr and
dy= Ycurr-Ydest
The North port is selected for routing after any one of the following conditions are met:
1. If the destination is in the same column with dy=1 (and dx=0) and F_N is 0 indicating that the link through north port is not faulty.
2. If the destination is in the same column and is more than 1 hop away(dx=0 and dy$>$1). We then check to see the $F_{N,N}$ fault bit of the current router. If the above specified fault bit and F_N of the current router are both 0 (indicating non-faulty links), we forward the flit/packet through the north port.
3. If the destination is in the north-east(NE) direction that is, dx$\geq$1 and dy$\geq$1. We then check F_N and $F_{N,E}$. If both the bits are zero, we now check if the turn is allowed or restricted. If $R_{N,E}$ equals 1, only then we choose the North port for routing.
4. If the destination is in the north-west(NW) direction that is, dx$\geq$1 and dy$\geq$1. We then check F_N and $F_{N,W}$. If both the bits are zero, we now check if the turn is allowed or restricted. If $R_{N,W}$ equals 1, only then we choose the North port for routing.

When the second level logic, generates more than one output port suitable for routing, there is a third level logic called the Selection logic, which selects a single output port out of all the options available for routing. This is done in the following way:
1. When more than one output port is chosen, for obvious reasons it will be one of North port(N) or South port(S) and one of either West port(W) or East port(E).For example when the destination is in the South-West quadrant(SW) and dx=1 and dy¿1, and after the second level of logic, both South port and West port are made available for routing, the selection logic attempts to chose South port over the West port as choosing the west port would make dx=0 at the next switch and the flit/packet has to travel only in the South direction to reach it's destination. Now if any of the links present in the South direction turn out to be faulty, the packet has to take a non-minimal to get routed towards it's destination. This causes unnecessary increase in the number of hops. If the port chosen is south, in the case of any faults in the south direction, it can then switch to west and continue southward till it reaches it's destination. This simple step thus helps us in decreasing the number of non-minimal paths. 2. When more than one output port is chosen and both dx>1, dy>1, we can choose one of the two ports at random to route the packet. For example, if the destination lies in the North-East quadrant(NE) and after calculation we get, dx>1 and dy>1. Now, we can choose either the North(N) or the East(E) port at random in this case.

Let us consider the case where the second level logic does not generate an output port because of the non-satisfaction of the conditions mentioned in the logic. In such cases, the packets get stuck at the router and cannot be forwarded further and thus get discarded. This increases the packet-loss and thus the time-delay. Therefore, for such packets we have another logic for port selection. Consider an example where the packet has its destination located in the North-East quadrant(NE). As mentioned we denote the neighbour of the current router in the South direction by η_S. Assume that the packet has been forwarded to the current router through its south inlet channel. Now, after the second level logic we know that the north and east ports have been blocked for routing due to faults. Also, the south port through which the packet has been forwarded cannot be chosen for routing as it creates the possibility of a LiveLock which leads to a lot of delay. This leaves us with the only option of West port being chosen for routing. Thus instead of discarding the packet and increasing packet-loss, we route the packet through the west port. Though the path now chosen will become non-minimal,care is taken that the packet eventually reaches its destination. As the packet is always re-routed through an alternative port at each router, the packet loss occurs only when there is congestion in the network where the packets have to be discarded without any choice. If in the above case the current router was the source itself, both west and south ports will be available for alternative routing and thus either of them could be chosen at random for forwarding. This logic for alternative routing decreases packet loss enormously.

Walk-through Example The above algorithm is now applied for the example shown in Fig 3. As shown in the figure, router with currID 14 is the source and the router with currID 5 is the destination.

At router 14, the first level logic is applied which makes N and W bits high indicating the presence of destination in the North-West quadrant. The second level logic selects both N and W as the possible output ports.Now dx and dy values are calculated,dx=1 and dy=2. Therefore, output port N is chosen over W because dy>1. The packet is forwarded to router with currID 10.
At router 10, the first level logic is applied which makes N and W bits high indicating the presence of destination in the North-West quadrant. The second level logic does not select any output port due to faulty links. As the packet has been forwarded through the router's south in_channel the east port is chosen for alternative routing.
At router 11, the first level logic is applied which makes N and W bits high indicating the presence of destination in the North-West quadrant. But the W port is eliminated as the packet was forwarded through the west in_channel. The second level logic selects only the north port and thus the packet is forwarded to router 7.
At router 7,the first level logic is applied which makes the W bit high indicating the presence of destination in the only along the West. The second level logic does not select any output port because of the presence of faults. As the packet had been forwarded through the south in_channel, the only available option for routing is north (As the node lies along the sides of the mesh, the east port is unavailable for routing).
At router 3, the S and W bits are made high. The second level logic selects both S and W ports but the south port is eliminated as the packet forwarding had been done through that port. Thus the packet goes to router 2 through the west port.
At router 2, the S and W bits are made high. The second level logic selects only W port for forwarding.
At router 1, only S is made high. The packet is thus forwarded to router 5.

IV. Result Analysis

In this section, we evaluate the proposed algorithm using a cycle accurate NoC simulator NIRGAM(NoC Interconnect Routing and Application Modelling). NIRGAM is developed using SystemC. We have implemented our method by adding additional Routing algorithms to the existing ones. We also modified the code to reflect changes in link failures. All simulations were performed on an 8 x 8 mesh topology with the usage of wormhole switching mechanism. The input channel buffer size was set to 4. The packet size was set to 50. Each simulation was run for 5000 cycles with warm-up sessions of 1000 cycles and generates traffic for 5000 cycles.As a performance metric we have evaluated the Overall Average Latency (in clockcycles per flit) which is defined as the number of cycles between the departure of a flit from it's source and the arrival at it's destination. The second performance metric that has been evaluated for all the algorithms was the produced Average Throughput (in flits/cycle/nic).Also, for each algorithm the percentage of loss of packets with varying faults has been calculated according to the formula, Percentage Loss= ((Total Packets Generated- Total Packets Received)/(Total Packets Generated))X 100. Performance values are taken at various simulation points, each with a different injection rate varying from 0 to 1(inclusive).In each plot, the

horizontal axis represents the injection rate and the vertical axes represent the performance metrics that are desired to be evaluated namely, percentage loss, Average Latency, Power and Throughput. All the simulations were performed using three synthetic traffic generator patterns available in NIRGAM namely: Uniform Traffic Pattern, Bit-Complement Traffic Pattern and Bit-Reversal Traffic Pattern with the percentage of link failures varying from 0 to 20(inclusive).

A. *Uniform Traffic*

The systhetic traffic pattern generators try to mimic the way of an on-chip communication by determining the communication pattern of the cores. When a core(tile) generates a packet, the Traffic pattern informs the core about the destination of the packet generated. In the uniform traffic model, each tile(core) generates packets and sends them to the other tiles using a uniform probability distribution. Fig 6. shows the comparision of the three algorithms in terms of the average Throughput values calculated.

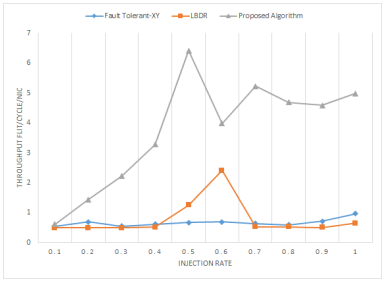

Fig. 6. Throughput for 5% link failures in Uniform traffic

As seen in the figure, a desired high throughput is obtained for the proposed algorithm in comparision to the other to algorithms at hand. In the case of 0 or 5 percent link failures the observed throughput is at it's maximum and with the increasing number of link failures the value gracefully degrades. This degradation in performance is mainly due to the presence of failures affecting total number of available paths between source and destination nodes. Also, for the same uniform traffic, the values of power dissipated have been plotted against the injection rates for all the algorithms as shown in the Fig. 7.

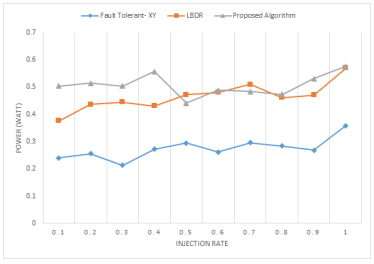

Fig. 7. Power for 20% link failures in Uniform traffic

It can be observed that the power dissipation values for the proposed algorithm are slightly high when compared to those of LBDR and Fault-Tolerant XY algorithms.

B. *Bit-Complement Traffic*

In this traffic, the destination router ID is the complement of the source router ID. This form of traffic represents the request-reply communication pattern or a pair of collaborating processes. Fig 8 shows the comparision of Fault-Tolerant XY, LBDR and the proposed algorithm in terms of the percentage loss incurred for Bit-Complement traffic.

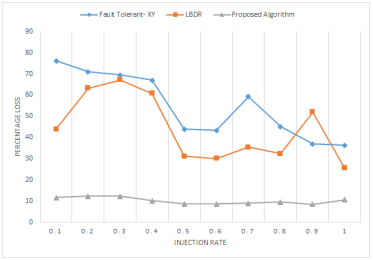

Fig. 8. Loss for 10% link failures in Bit-Complement traffic

As clearly observed from the figure, the percentage loss of the proposed algorithm is very low when compared to that of LBDR or Fault-Tolerant XY Routing algorithm. Also, the loss almost remains constant or slightly increases with the increasing injection rate and increasing number of link failures introduced into the network.

C. *Bit-Reversal Traffic*

In this traffic, the destination router ID is the source ID written in the reverse order. As shown in the Fig 9, the average latency values obtained in clock cycles per flit have been plotted against the injection rates for the algorithms.

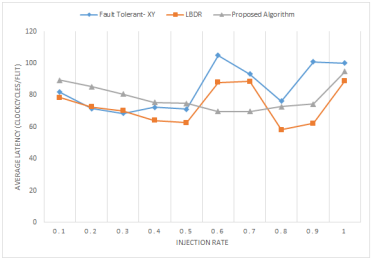

Fig. 9. Average Latency for 10% link failures in Bit reversal traffic

When closely observed, the graph makes it clear that the latency values for the proposed algorithm are almost the same or slightly higher when compared to the values of Fault-Tolerant XY algorithm. The slightly high latency values can be accounted to the non-minimal paths chosen by packets during routing.

V. Conclusion

We have proposed a fault-tolerant algorithm in a compact manner for distributive routing in NOC, which is capable of adapting to irregularities in network caused by the introduction of link failures in the topology. This algorithm requires the

usage of a few fixed configuration bits at each node(router), thus reducing the memory overhead associated with the routing table mechanism. The algorithm also guarantees connectivity and deadlock freedom with the help of routing bits and thus doesn't need additional hardware requirement like virtual channels. While achieving graceful performance degradation, our algorithm presents an unmistakeble superiority while tackling faults which can be seen from the very less percentage of packets that have been lost during routing.