JJEE
Volume 3, Number 2, 2017
Pages 102-111
Jordan Journal of Electrical Engineering
ISSN (Print): 2409-9600, ISSN (Online): 2409-9619

Fault-Tolerant Routing in Mesh-Connected Multicomputers based on Majority-Operator-Produced Transfer Direction Identifiers

Jamil S. Al-Azzeh

Department of Computer Engineering, Al-Balqa' Applied University, Amman, Jordan
e-mail: jamil.azzeh@bau.edu.jo

Received: February 8, 2017 *Accepted: April 27, 2017*

Abstract**—** The paper focuses on fault-tolerant 2D mesh-connected multicomputers, which can proceed operating even if some of their units and/or links are faulty, and more particularly, on the routing environment of the multicomputers capable of transferring packets between healthy processor units along the paths containing faulty components. A novel approach to the fault-tolerant packet routing is presented making it possible to increase the packet delivery probability by expanding the XY-routing scheme. The proposed approach is capable of recovering corrupted routing direction data contained in the address section of a packet by using the majority operator applied to the current direction IDs generated at the current and the two preceding hops on the route. Our simulation study shows that the successful routing probability increases at least by 40% with the new scheme; yet, it grows on as the route length increases compared to the XY-routing with no routing data recovery.

Keywords**—** Fault-tolerance, Mesh-connected multicomputers, Packet routing, Reliability.

I. INTRODUCTION

Mesh-connected networks are used widely in modern commercial and experimental multicomputers and multiprocessors, including system-on-a-chip multiprocessors [1] and [2]. The performance of such parallel systems depends critically on the efficiency of the packet routing (transfer) procedure employed because interprocessor communication takes place in the many routines invoked while parallel applications are running on a multicomputer. Packet transfer is organized in a sequence of hops. At each hop, a packet is processed by the corresponding processor unit and then relayed to a given direct neighbor according to the route-selection algorithm implemented. For example, if the XY-routing scheme is used, a packet needs to make a number of hops along the *X* axis until it reaches the destination column; then, it travels along the *Y* axis to the destination processor.

For fault-tolerant multicomputers, which can continue to operate correctly after losing some of their basic components (units, links) and are the focus of the present paper, a fault-tolerant routing procedure is assumed to be employed. Fault-tolerant routing has been a research focus for the past four decades; and many solutions have been suggested. Existing routing algorithms and schemes rely on specific faulty unit/link/region detection and identification methods combined with a set of fault bypass rules [3]-[16]. The key differences among various routing algorithms are associated with the topology of the communication network, types of faults treated, and implementation level (software, hardware, or hybrid). Some routing schemes are applicable to a number of topologies and take into account different types of faults (separate unit faults, link faults, faulty regions of a specific form, e.g., convex or concave). Most commonly, fault-tolerant routing algorithms assume that if there is a fault in a unit, the unit is considered non-healthy as a whole. This means that no packets should be routed along any path including this unit. In practice, this routing limitation may be a bit too stringent. For example, a faulty unit's communication hardware may still be healthy; and packets may travel through this unit with no data loss or corruption. By contrast, the

Corresponding author's e-mail: jamil.azzeh@bau.edu.jo

communication hardware of a healthy unit may add errors to packet transfer because of undetected local faults, so packets may be delivered to wrong destination processors.

In the present paper, we propose a new approach to fault-tolerant routing; the approach might be considered as an addition to existing routing schemes applicable to a number of network topologies. Its key idea is that at each hop, the output channel (direction) to transfer a packet is determined by applying the majority operator to the three direction IDs, one of which is produced by the current unit according to the route selection algorithm and the other two are read from the address section of the packet. As a result, if one direction ID is invalid, the majority-operator-based scheme generates a correct direction ID to route the packet. The direction IDs read from the packed address section are assumed to be determined by the two predecessors when the packet passes through these units. We compare our scheme to the *XY*-routing with no routing data recovery to demonstrate the successful routing probability increase using a number of simulation studies.

II. BASIC PRINCIPLES AND ASSUMPTIONS

Our approach has no specific requirements for route selection, but in this paper, we illustrate how it works assuming the *XY*-routing algorithm is used. We consider *XY*-routing for an eight-direct-neighbor mesh system (see Fig.1), which is denser than the four-direct-neighbor meshes employed typically and provides a greater number of redundant paths to the destination.

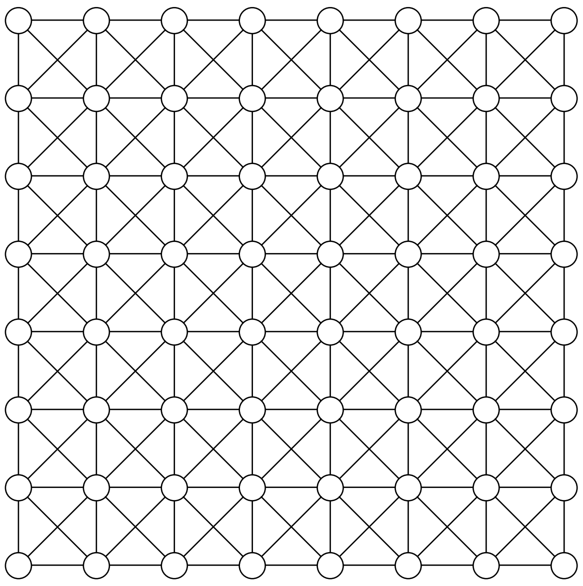


Fig. 1. Eight-direct-neighbor mesh topology under consideration

The proposed fault-tolerant routing scheme is based on the following generic principles. First, each processor unit determines the transfer directions for the next hops (i^{th}, $(i+1)^{th}$, and $(i+2)^{th}$ hops) and saves the resulting direction IDs for the $(i+1)^{th}$ and the $(i+2)^{th}$ hops to the packet destination field. Second, each processor unit calculates the packet transfer direction ID by applying the majority operator to the next direction ID provided by the current processor and the two direction IDs determined previously at the preceding hops ($(i-1)^{th}$ and $(i-2)^{th}$ hops), and read from the packet destination field.

Using the above principles, we formulated packet destination field manipulation rules that make it possible to recover valid direction IDs while routing packets in a communication network containing faulty nodes or regions.

Let i be the current routing hop for a packet, and m_i denote the current processing unit containing the packet. Let c_i^j be the packet direction ID for processor unit m_i calculated previously by processor unit m_j, $j \in \{i-2, i-1, i\}$. Thus, the actual packet direction ID c_i is determined according to the following formula:

$$c_i = \#\left(c_i^{i-2}, c_i^{i-1}, c_i^i\right) \tag{1}$$

where the symbol # stands for the majority operator; and it is supposed initially that $c_1^{-1} = c_1^0 = c_1^1$, $c_2^0 = c_2^1$. The # operator is defined according to the following rule:

$$\#(x, y, z) = xy \vee xz \vee yz$$

with x, y and z standing for arbitrary binary variables. Based on the above assumptions, the packed address field needs to include the following 5 subfields:

$$\mathbf{C}^{-2}.\mathbf{C}^{-1}.\mathbf{C}^{+1}.\mathbf{DA}.\mathbf{SA}$$

where DA and SA are the destination and source address subfields, respectively (SA is optionally based on the routing mechanism utilized); $\mathbf{C}^{-2}$ and $\mathbf{C}^{-1}$ are the ith hop direction ID subfields, whose content is calculated by processors m_{i-2} and m_{i-1}, respectively; and $\mathbf{C}^{+1}$ stands for the (i+1)th hop direction ID subfield, determined by processor m_{i-1}. Note that subfields $\mathbf{C}^{-2}$ and $\mathbf{C}^{-1}$ contain IDs c_i^{i-2} and c_i^{i-1}, respectively, and are used to produce ID c_i according to (1). Subfield $\mathbf{C}^{+1}$ contains ID c_{i+1}^{i-1}, which is supposed to be used at the (i +1)th hop to calculate ID c_{i+1} according to (1).

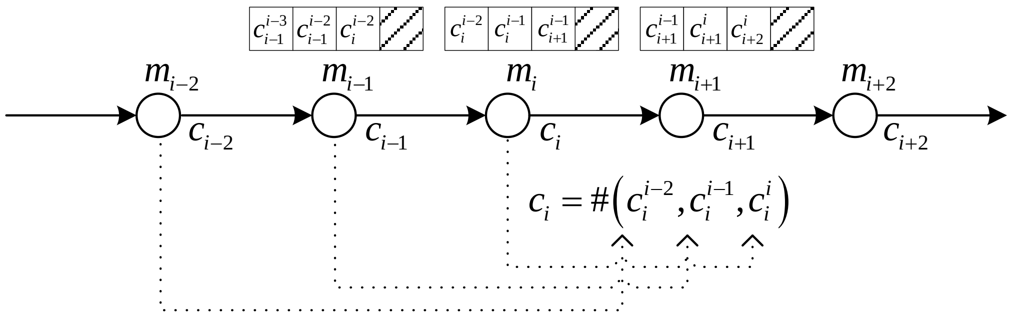
So long as a packet keeps traveling along a given route, subfields $\mathbf{C}^{-2}.\mathbf{C}^{-1}.\mathbf{C}^{+1}$ are modified at the start of each hop $(i, i > 2)$ according to the following replacement rule:

$$\begin{cases} \mathbf{C}^{-2} \leftarrow \mathbf{C}^{+1}; \\ \mathbf{C}^{-1} \leftarrow c_{i+1}^{i}; \\ \mathbf{C}^{+1} \leftarrow c_{i+2}^{i}. \end{cases} \tag{2}$$

If $i = 1$, the above rule (2) transforms to

$$\begin{cases} \mathbf{C}^{-2} \leftarrow c_2^1; \\ \mathbf{C}^{-1} \leftarrow c_2^1; \\ \mathbf{C}^{+1} \leftarrow c_3^1. \end{cases} \tag{3}$$

Fig. 2 shows the formulated routing control principles and rules (2), (3), given an arbitrary route for a packet to move along to the destination. The circles denote the processing units belonging to the route; and the arrows denote the routing direction.

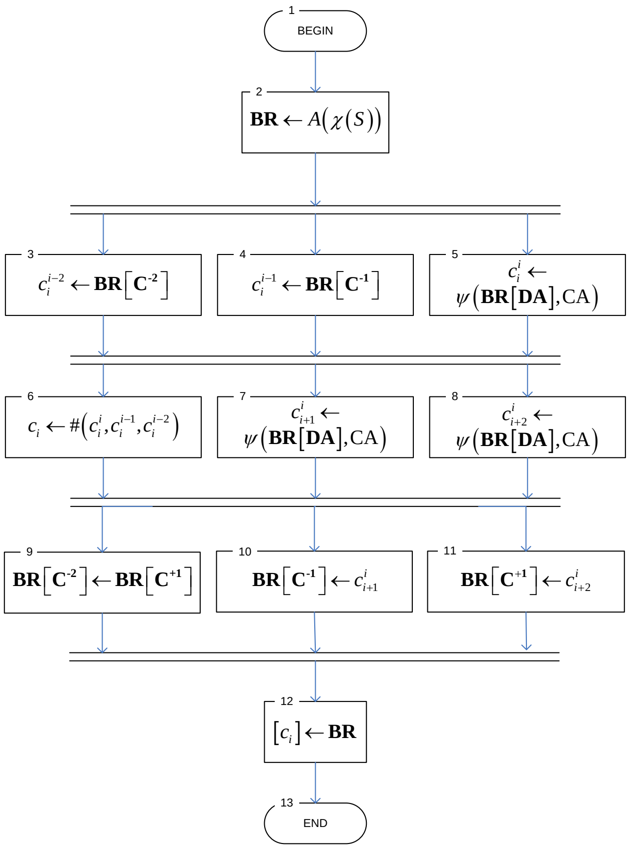


c_{i-1}^{i-3} c_{i-1}^{i-2} c_i^{i-2}
c_i^{i-2} c_i^{i-1} c_{i+1}^{i-1}
c_{i+1}^{i-1} c_{i+1}^{i} c_{i+2}^{i}
m_{i-2} m_{i-1} m_i m_{i+1} m_{i+2}
c_{i-2} c_{i-1} c_i c_{i+1} c_{i+2}
$c_i = \#\left(c_i^{i-2}, c_i^{i-1}, c_i^{i}\right)$

Fig. 2. Packet address field modification according to rules (2) and (3)

III. Routing Control Algorithm

In Fig. 3, the parallel routing control algorithm that implements the above principles and rules (2) and (3) is presented. In the flowchart shown in Fig. 2, packet fields are denoted by a bold text. Additionally, the following identifiers are used. The backward arrows «←» denote the assignment operator; BR stands for the buffer employed to temporarily store processed and routed packets; DA is the packet destination address; BR[*x*] denotes the contents of field *x* of a packet stored in BR; CA is the current processor address; *A(x)* denotes the address part of a packet routed; *S* stands for the set of packets arriving at the current processor inputs; and $[c_i]$ is the value transferred to channel c_i .



1 BEGIN
2 $\mathbf{BR} \leftarrow A(\chi(S))$
3 $c_i^{i-2} \leftarrow \mathbf{BR}[\mathbf{C}^{-2}]$
4 $c_i^{i-1} \leftarrow \mathbf{BR}[\mathbf{C}^{-1}]$
5 $c_i^{i} \leftarrow \psi(\mathbf{BR}[\mathbf{DA}], \mathrm{CA})$
6 $c_i \leftarrow \#(c_i^{i}, c_i^{i-1}, c_i^{i-2})$
7 $c_{i+1}^{i} \leftarrow \psi(\mathbf{BR}[\mathbf{DA}], \mathrm{CA})$
8 $c_{i+2}^{i} \leftarrow \psi(\mathbf{BR}[\mathbf{DA}], \mathrm{CA})$
9 $\mathbf{BR}[\mathbf{C}^{-2}] \leftarrow \mathbf{BR}[\mathbf{C}^{+1}]$
10 $\mathbf{BR}[\mathbf{C}^{-1}] \leftarrow c_{i+1}^{i}$
11 $\mathbf{BR}[\mathbf{C}^{+1}] \leftarrow c_{i+2}^{i}$
12 $[c_i] \leftarrow \mathbf{BR}$
13 END

Fig. 3. Routing control algorithm

The algorithm shown in Fig. 3 is generic. Therefore, arbitrary packet ordering and selection algorithms may be used. The route selection algorithm itself may be arbitrary. The selection of a packet out of set *S* is done according to a given algorithm χ. The selection of a route for the packet is done according to a predefined routing function ψ. Note that function ψ is used to determine the output directions for the current (see statement 5) and the next two routing hops (see statements 7 and 8).

In Table 1, an extended version of the *XY* routing scheme is presented to recover route data according to the proposed routing control algorithm (note that we assume an 8-direct-neighbor mesh-connected network).

The second column of the table contains all possible relations between destination address $\mathrm{X.Y}$ read from the corresponding field of the packet being processed and the current processor unit address $\mathrm{X1.Y1}$. The rest of the table determines how a packet should be routed at the current hop ($s=1$, value c_i^i) and the following two hops ($s=2$, value c_{i+1}^i; $s=3$, value c_{i+2}^i). In the same fashion, it is possible to extend other routing schemes. To use Table 1 in the routing process, one should consider that the order in which c_i^i, c_{i+1}^i, and c_{i+2}^i are calculated according to the algorithm shown in Fig. 3 is significant.

TABLE 1
THE EXTENDED VERSION OF THE XY ROUTING SCHEME

No.	Relations Between Destination Address $\mathrm{X.Y}$ and Current Processor Address $\mathrm{X1.Y1}$	Packet transfer direction (ψ_s)		
		$s=1\left(c_i^i\right)$	$s=2\left(c_{i+1}^i\right)$	$s=3\left(c_{i+2}^i\right)$
1	$\mathrm{X}>\mathrm{X1}+(s\text{-}1), \mathrm{Y}>\mathrm{Y1}+(s\text{-}1)$	↗	↗	↗
2	$\mathrm{X}>\mathrm{X1}+(s\text{-}1), \mathrm{Y}<\mathrm{Y1}-(s\text{-}1)$	↘	↘	↘
3	$\mathrm{X}<\mathrm{X1}-(s\text{-}1), \mathrm{Y}>\mathrm{Y1}+(s\text{-}1)$	↖	↖	↖
4	$\mathrm{X}<\mathrm{X1}-(s\text{-}1), \mathrm{Y}<\mathrm{Y1}-(s\text{-}1)$	↙	↙	↙
5	$\mathrm{X}>\mathrm{X1}+(s\text{-}1), \mathrm{Y}=\mathrm{Y1}$	→	→	→
6	$\mathrm{X}>\mathrm{X1}+(s\text{-}1), \mathrm{Y}=\mathrm{Y1}+1$	↗	→	→
7	$\mathrm{X}>\mathrm{X1}+(s\text{-}1), \mathrm{Y}=\mathrm{Y1}-1$	↘	→	→
8	$\mathrm{X}>\mathrm{X1}+(s\text{-}1), \mathrm{Y}=\mathrm{Y1}+2$	↗	↗	→
9	$\mathrm{X}>\mathrm{X1}+(s\text{-}1), \mathrm{Y}=\mathrm{Y1}-2$	↘	↘	→
10	$\mathrm{X}<\mathrm{X1}-(s\text{-}1), \mathrm{Y}=\mathrm{Y1}$	←	←	←
11	$\mathrm{X}<\mathrm{X1}-(s\text{-}1), \mathrm{Y}=\mathrm{Y1}-1$	↙	←	←
12	$\mathrm{X}<\mathrm{X1}-(s\text{-}1), \mathrm{Y}=\mathrm{Y1}+1$	↖	←	←
13	$\mathrm{X}<\mathrm{X1}-(s\text{-}1), \mathrm{Y}=\mathrm{Y1}-2$	↙	↙	←
14	$\mathrm{X}<\mathrm{X1}-(s\text{-}1), \mathrm{Y}=\mathrm{Y1}+2$	↖	↖	←
15	$\mathrm{X}=\mathrm{X1}, \mathrm{Y}>\mathrm{Y1}+(s\text{-}1)$	↑	↑	↑
16	$\mathrm{X}=\mathrm{X1}, \mathrm{Y}<\mathrm{Y1}-(s\text{-}1)$	↓	↓	↓
17	$\mathrm{X}=\mathrm{X1}, \mathrm{Y}=\mathrm{Y1}$	-	-	-

IV. SAMPLE HARDWARE-LEVEL IMPLEMENTATION

In Fig. 4, the structure of a communication unit of an n-direct-neighbor network is diagrammed by implementing the proposed routing control algorithm.

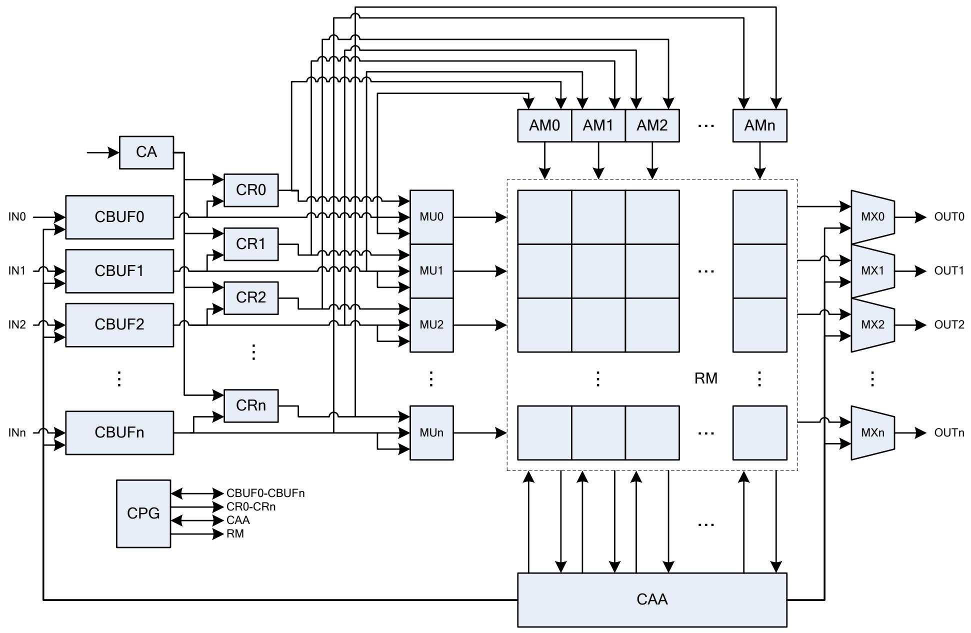


Fig. 4. Structure of a communication unit that implements the proposed routing control algorithm

The proposed structure consists of a set of input buffers CBUF0-CBUF*n*, register matrix RM, packet allocation analyzer CAA, current processor address generator CA, route selection units CR0-CR*n*, a set of output multiplexers MX0-MX*n*, packet address modification units AM0-AMn, a set of majority units MU0-MU*n*, and a clock pulse generator CPG.

The operation of the above unit as part of a communication network may be represented as follows. Packets eventually arrive at the inputs IN0-IN*n* of the current unit (note that IN0 connects the unit to the processor core; and the inputs IN1-IN*n* are used to communicate with the neighbor communication units). Having arrived at the input IN*i*, a packet is written to the buffer CBUF*i*; and the packets written to the buffers CBUF0-CBUF*n* are then read in groups according to algorithm χ. When a packet is fetched from the corresponding buffer, address fields $\mathbf{C}^{-2}\bullet\mathbf{C}^{-1}$ are extracted to be transformed according to statements 3 and 4 of the algorithm shown in Fig. 3. At the same time, route selectors CR0-CR*n* produce routing direction IDs c_i^i according to statement 5 of the algorithm.

Then, selectors CR0-CR*n* generate pairs of values c_{i+1}^i, c_{i+2}^i according to statements 7 and 8 of the routing control algorithm; they as well generate the majority units MU0-MU*n* issue routing direction IDs c_i according to statement 6 (also see (1)). The process goes on; and units AM0-AM*n* modify the address fields of the packets being processed in accordance with statements 9-11 of the algorithm (see rule (2)). Immediately after the modifications are complete, the packets are transferred to the register matrix RM (rows of the matrix are selected based on the values of c_i). Then, CAA starts analyzing the packet allocation before it determines the order in which the packets should be transmitted to the outputs OUT0-OUT*n*. As soon as the transmission is complete, the packets are removed from the corresponding buffers CBUF0-CBUF*n*. Then, the next set of packets are processed and routed.

V. SUCCESSFUL ROUTING PROBABILITY EVALUATION

In the present section, we evaluate the magnitude of increase in the probability of successful routing when applying the proposed routing control technique. We assume that the errors that make routing directions wrong are independent of each other; and the probability that such an error occurs at a given hop is:

$$P(t) = e^{-\lambda t}$$

where λ denotes the rate parameter of the error distribution function (average number of errors per unit time).
For a given route including h hops ($h \geq 1$), if no specific fault-tolerant routing control is utilized, the probability of unsuccessful routing at a given moment t is

$$Q_0 = 1 - p^h \tag{4}$$

where p is the probability that the communication unit circuitry is healthy.
If the fault-tolerant routing control scheme is introduced, the above probability becomes

$$Q_\# = 1 - p^2\left(3p^2 - 2p^3\right)^{\chi(h-2)} \tag{5}$$

where $\chi(a)$ denotes a function such as $\chi(a) = a$ if $a > 0$ and $\chi(a) = 0$ if $a \leq 0$. Formula (5) represents the case when the majority scheme that implements (1) is considered healthy. Given the probability p_0 that the majority scheme is healthy, (4) transforms to

$$Q_\# = 1 - p^2\left[\left(3p^2 - 2p^3\right)\left(3p_0^2 - 2p_0^3\right)\right]^{\chi(h-2)} \tag{6}$$

Dividing (4) by (6), we can evaluate the relative increase in the successful routing probability achieved by employing the proposed routing control technique compared to the XY-routing as

$$\delta Q = \frac{Q_0}{Q_\#} = \frac{1 - p^h}{1 - p^2\left[\left(3p^2 - 2p^3\right)\left(3p_0^2 - 2p_0^3\right)\right]^{\chi(h-2)}} \tag{7}$$

In Fig. 5, the δQ versus h graphs presented were calculated using (7) for several predefined values of p with $h \geq 3$ and $p_0 = 0{,}99$ (note that with $h \geq 3$, we obtain $\delta Q = 1$).
Fig. 5 demonstrates that the proposed approach is more efficient for longer routes ($h = 4$ and higher) and higher probability values p.
To validate our theoretical results, we performed a simulation to investigate the operation of a communication unit that can implement the proposed routing control scheme as part of a mesh-connected communication network. We used a well-known Q-chart model to represent the behavior of a unit in the simulation study and ran a number of experiments in the Visual Q-chart Simulator environment developed by Zotov (South-West State University, Russian Federation). The developed Q-chart model is the graph shown in Fig. 6.

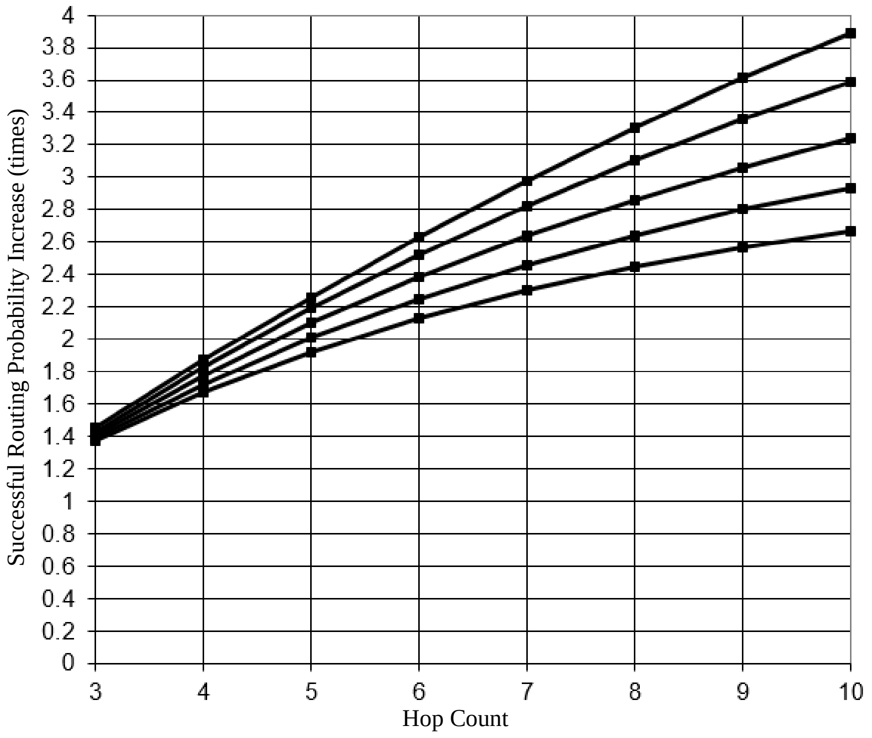


Fig. 5. Successful routing probability increase δQ versus hop count *h* graphs for different values of *p* (subject to $h \geq 3,\ p_0 = 0,99$)

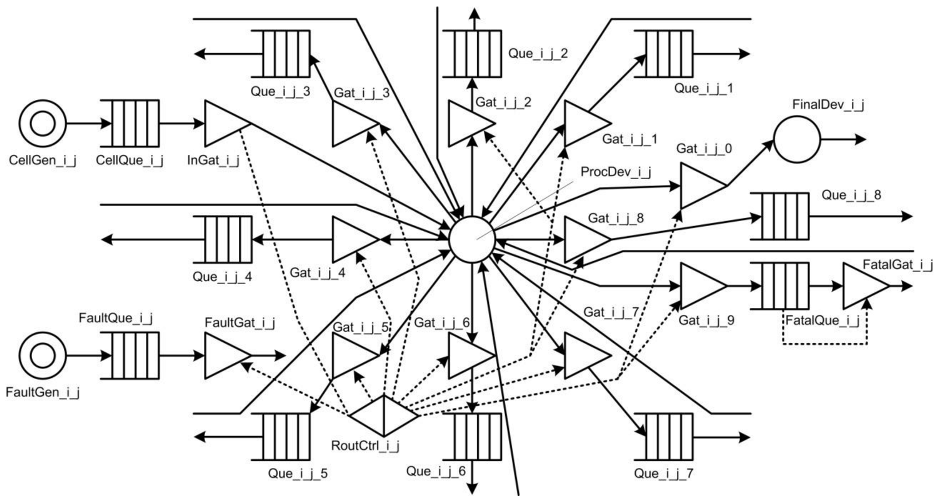


Fig. 6. Q-chart model representing a unit that implements the proposed routing control algorithm

The operation and functions of the Q-chart elements are as follows. CellGen_*i_j* is a generator that simulates the arrival of packets issued by the *ij*th processor core. FaultGen_*i_j* is also a generator; and it simulates the fault flow for the *ij*th unit. If a fault occurs, then queue FaultQue_*i_j* becomes non-empty, and its new state is taken into account by the random controller RoutCtrl_*i_j*, which executes the proposed routing control algorithm. If the occurred fault is fatal, queue FaultQue_*i_j* will always be non-empty, meaning that this unit is not healthy anymore. If the fault is simply a glitch, RoutCtrl_*i_j* opens gate FaultGat_*i_j* at a randomly picked moment, so queue FaultQue_*i_j* becomes empty (and the unit is assumed healthy again). A packet is considered to be processed and sent to its destination as soon as it is held by device FinalDev_*i_j*. If packet transfer is not possible because of invalid routing data, which cannot be recovered by the routing control algorithm, the packet is sent to queue

FatalQue_*i*_*j* via gate Gat_*i*_*j*_9. The total count of all packets contained in queues FatalQue_*i*_*j* across the mesh gives us the number of packets lost.
Our study shows that the hardware redundancy of a unit that supports the proposed routing control scheme is less than 15%. This does not depend on the mesh size and grows slightly as the number of direct neighbors *n* (mesh connectivity) increases. Thus, system scalability is not affected.

VI. CONCLUSION

In the present paper, we proposed a new approach to fault-tolerant packet routing in mesh-connected multicomputer systems. The proposed approach makes it possible to recover the corrupted routing data contained in the address section of a packet by using transfer direction IDs produced by the majority operator applied to the current direction IDs generated at the current and the two preceding hops on the route. Our evaluation and simulation results show a significant increase in successful routing probability attainable with the new scheme, which grows considerably as the average packet route length increases. We found that the hardware redundancy of a unit implementing the proposed routing control scheme is as low as 15%.

REFERENCES

[1] J. Al-Azzeh, M. Leonov, D. Skopin, E. Titenko, and I. Zotov, "The organization of built-in hardware-level mutual self-test in mesh-connected VLSI multiprocessors," *Information Technology*, vol. 3, no. 2, pp. 29-33, 2015.

[2] A. Tumanov, J. Wise, O. Mutlu, and G. Ganger, "Asymmetry-aware execution placement on manycore chips," *Proceedings of Systems for Future Multicore Architectures Workshop*, pp. 1-6, 2013.

[3] J. Duato, "Theory of fault-tolerant routing in wormhole networks," *Proceedings of Parallel and Distributed Systems Conference*, pp. 600-607, 1994.

[4] P. Sui and S. Wang, "An improved algorithm for fault-tolerant wormhole routing in meshes," *IEEE Transactions on Computers*, vol. 46, no. 9. pp. 1040-1042. 1997.

[5] S. Chalasani and R. Boppana, "Communication in multicomputers with nonconvex faults," *IEEE Transactions on Computers*, vol. 46, no. 5. pp. 616-622. 1997.

[6] J. Al-Sadi, K. Day, and M. Ould-Khaoua, "Probability-based fault-tolerant routing in hypercubes," *The Computer Journal*, vol. 44, no. 5, pp. 368-373. 2001.

[7] D. Xiang and A. Chen, "Fault-tolerant routing in 2D tori or meshes using limited-global-safety information," *Proceedings of Parallel Processing Conference*, pp. 231-238, 2002.

[8] G. Wang and J. Chen, "A new fault-tolerant routing scheme for 2-dimensional mesh networks," *Proceedings of Parallel and Distributed Computing Conference, Applications and Technologies*, pp. 95-98, 2003.

[9] G. Wang, T. Li, and J. Chen, "A probabilistic approach to fault-tolerant routing algorithm on mesh networks," *Proceedings of Parallel and Distributed Systems Conference*, pp. 577-584, 2004.

[10] C. Ho and L. Stockmeyer, "A new approach to fault-tolerant wormhole routing for mesh-connected parallel computers," *IEEE Transactions on Computers*, vol. 53, no. 4. pp. 427-438. 2004.

[11] D. Xiang, J. Sun, J. Wu, and K. Thulasiraman, "Fault-tolerant routing in meshes/tori using planarly constructed fault blocks," *Proceedings of Parallel Processing Conference*, pp. 577-584, 2005.

[12] J. Wu and Z. Jiang, "On constructing the minimum orthogonal convex polygon for the fault-tolerant routing in 2-D faulty meshes," *IEEE Transactions on Reliability*, vol. 54, no. 3, pp. 449-458, 2005.

[13] M. Gomez, N. Nordbotten, J. Flich, P. Lopez, A. Robles, J. Duato, T. Skeie, and O. Lysne, "A routing methodology for achieving fault tolerance in direct networks," *IEEE Transactions on Computers*, vol. 55, no. 4, pp. 400-415, 2006.

[14] S. Khonsari, A. Dadlani, and A. Ould-Khaoua, "A probabilistic characterization of fault rings in adaptively-routed mesh interconnection networks," *Proceedings of Parallel Architectures, Algorithms, and Networks Symposium*, pp. 233-238, 2008.

[15] M. Kobayashi, T. Takabatake, T. Matsushima, and S. Hirasawa, "Probabilistic fault diagnosis and its analysis in multicomputer systems," *Proceedings of IEEE Systems, Man, and Cybernetics Conference*, pp. 1205-1211, 2011.

[16] Y. Nishiyama, Y. Hirai, and Y. Kaneko, "Fault-tolerant routing based on improved safety levels in pancake graphs," *Proceedings of Parallel and Distributed Computing, Applications and Technologies Conference*, pp. 76-81, 2014.