ISSN 2047-3338

A Proficient Reconfiguration on Interconnection Networks

S. Mohiadeen Abdul Kadhar[1] and Dr. T. Revathi[2]
[1]National College of Engineering, Tirunelveli, India
[2]Mepco Schelenk Engineering College, Sivakasi, India

***Abstract*– Network reconfiguration situation may occur frequently in interconnection networks while add or remove the components under repair. This situation may block packet forward in the path and it leads to deadlocks. Our aim is to provide the block-free, deadlock-free path and maintain the performance in high level. Many techniques are implemented so far for the above situation still we are unable to achieve the maximum efficiency. In this paper, we propose the most efficient algorithms SRS (Sovereign Reconfiguration System) to make deadlock freedom and establish the right path to packet forwarding process in active manner under this reconfiguration situation.**

***Index Terms*– Deadlock, Efficiency, Networks, SRS and Interconnection**

I. INTERCONNECTION NETWORKS

HIGH-performance interconnection networks comprise the communication backbone in digital systems at several system levels. At the higher system levels, local-area networks (LANs) are used in clusters of PCs, networks of workstations and other distributed processing systems which serve as cost/performance effective alternatives to tightly-coupled massively parallel processing systems. System-area networks (SANs) are used for interconnecting processors, memories, and I/O devices in systems with the primary goal of increasing reliability in the presence of link/router failures. Storage-area networks (STANs) [1] are used to increase performance and reliability of large disk arrays by offering access to stored data by processors through multiple paths, thus providing continued service in the presence of processor failure.

Internet protocol router fabric (IPRF) networks are used within IP routers to handle IP traffic at high (multi gigabit) sustained line rates. Server I/O (SIO) and inter processor communication (IPC) [2], [3] networks are used to overcome many of the scalability limitations of multichip bus-based systems, allowing high-speed interconnections between memory controllers and I/O devices, direct access to disk from LAN adapters, and concurrent communication between processors, memories and I/O devices in multiprocessors. Likewise, at lower levels, networks-on-chip (NOCs) [4], [5], [6] are used to overcome many of the performance limitations of bus-based systems at the chip level. Parallel computing and communication systems built from the above networks require high-performance communication services [7] with high reliability, availability and dependability- collectively, high robustness. The performance [8] of the interconnection network is measured, in part, by packet delivery time [9] from source to destination (i.e., *latency*) and by the number of packets delivered per unit time (i.e., *throughput*). In essence, a high-performance network allows the maximum number of packets to make forward progress to their destinations in minimal time, preferably along shortest paths [10] to preserve network bandwidth. Likewise, the reliability, availability and dependability of a network equally impact the overall goodness (*quality of a system*). These attributes are measured, in part; by the network's ability to remain up and running at near normal levels even when events occur which change its configuration, possibly due to changes in users' needs and/or system state. Such reconfiguration events may include, for example, hot-swapping of components, failure or addition of links/nodes, activation or deactivation of hosts/routers, etc., in a LAN [24] environment. Since network resources are finite and, ultimately, are contended for, structural hazards on those resources are inevitable which delay or prevent packet transmission in the network.

This occurs even in networks that feature advanced router architectures. Such hazards cause packets to block which, eventually, can lead to network congestion and, possibly, *deadlock.* One of the more critical problems to be addressed in order to achieve high network performance and robustness is that of efficiently handling deadlock anomalies.

The rest of the paper organized as follows: Section 2 provides the idea description about the proposed system. Section 3 describes the related work about the system. Section 4 deals with limitations of reconfiguration. Section 5 provides network model and assumption related to the proposed system and also this section introduces the concept of Sovereign network Reconfiguration System (SRS).

A. Network and Router

Direct networks consist of a set of nodes interconnected by point-to-point links or channels. No restriction is imposed on the topology of the interconnection network. Each node has a router. We assume that the switch is a crossbar, therefore allowing multiple packets [11] to traverse a node simultaneously without interference. The routing and arbitration unit configures the switch [12], determining the output channel for each packet as a function of the destination node, the current node, and the output channel status. The routing and arbitration unit can only process one packet header at a time. If there is contention for this unit, access is round-robin [13]. When a packet gets the routing and arbitration unit but cannot be routed because all the valid output channels are busy, it waits in the corresponding input buffer until its next turn.

By doing so, the packet gets the first valid channel that becomes available when it is routed again. This strategy achieves a higher routing flexibility than strategies in which blocked packets wait on a single predetermined channel. Physical channels [14] are bidirectional full duplex. Physical channels may be split into virtual channels. Virtual channels are assigned the physical channel cyclically, only if they are ready to transfer a flit [21] (demand-slotted round-robin). Fig 1 shows the generic router model.

The interconnection network *I* is modeled by using a strongly connected directed graph with multiple arcs, *I = G(N, C)*. The vertices of the graph *N* represent the set of processing nodes. The arcs of the graph *C* represent the set of communication channels. More than a single channel is allowed to connect a given pair of nodes. Bidirectional channels [15] are considered as two unidirectional channels. We will refer to a channel and its associated edge buffer indistinctly. The source and destination nodes of a channel c_i are denoted s_i and d_i respectively. A routing algorithm is modeled by means of two functions: routing and selection.

The *routing function* [16] supplies a set of output channels based on the current and destination nodes. A selection from this set is made by the *selection function* [17] based on the status of output channels at the current node. This selection is performed in such a way that a free channel (if any) is supplied. If all the output channels are busy, the packet will be routed again until it is able to reserve a channel, thus getting the first channel that becomes available. As we will see, the routing function determines whether the routing algorithm is deadlock-free or not. The selection function only affects performance.

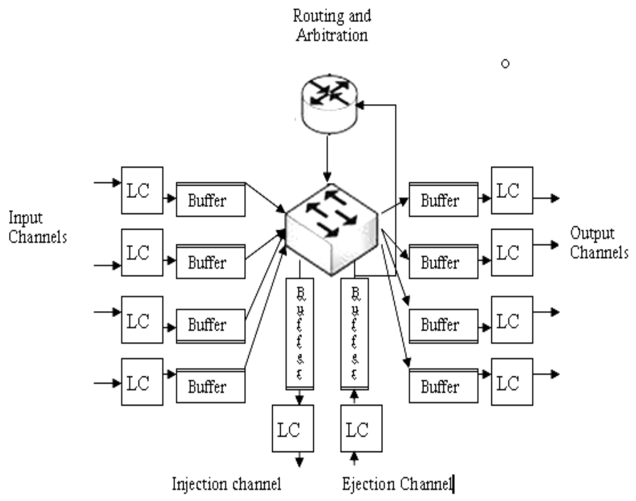

Fig. 1: Router Model

II. MOTIVATION

Communication sub-systems of interconnection networks are made up of advanced switches and routers. Based on the routing method, it is classified as distributed [22], and source routing models [23]. The sub-system are centrally monitored by autonomous system called subnet manger [20] (distributed routing) and fabric manager or mapper [19] (source routing). During the reconfiguration process, the mangers collects all information (knowledge) of the network based on change assimilation process and prepares the new path for packet forward. The ancient method to maintain deadlock free path preparation is static reconfiguration.

Here process happened in three phases. During the first phase, injection of packets in to the network is stopped. In second phase, the network has to wait until a packet in exiting network has to drain. And in third phase packets are injected in to the network with new routing paths. In this method we can achieve cent percent deadlock freedom but network down times are high. The recent popular methods are dynamic reconfiguration. It allows both old and new routing packets simultaneously but we need a separate mechanism to split old and new packets inside the network. In this paper, we propose the new protocol in dynamic manner, to reduce the reconfiguration occurrences and path establishment with deadlock freedom. In this strategy, a group is formed around the failure link or node and elects a node as leader to get the planning algorithm from gateway. Sovereign Reconfiguration System (SRS) will provide the right path with deadlock freedom.

III. RELATED WORK

The most recent method proposes that uses a close graph-based reconfiguration method, it works based on up*/Down*[30] routing method. Any topology change occurs during reconfiguration the centralized autonomous system [16] calculates new path by Fully Explicit Algorithm [33] for distributed network and dijkstra based algorithm [18] in source routing networks. Next recent approach is epoch marking system, it guarantees' that only packets potentially leading to a deadlock will be removed in saturation condition. It is based on regressive deadlock recoveries and used in distributed networks. Overlapping Static Reconfiguration (OSR) [31] is a static method introduce a TOKEN (a special packet) in-between the old and new routing functions. Deadlocks are avoided by ensuring that each link first transmits packets that belong to the old routing function, then the token, and finally packets that belong to the new one.

This method tested in both source and distributed routing. Another popular method, Double Scheme (DS) [28] requires additional resources in order to work with two disjoint sets of virtual channels to separate packets routed according to the old routing function from packets routed according to the new. NetRec [29] and LORE [32] have been specially designed for rerouting messages around a faulty node. The NetRec scheme requires every switch to maintain information about switches some number of hops away. LORE relies on the existence of virtual channels to reroute the packets headed toward the fault. The implementation of these techniques is not feasible in networks with source routing. The reason is that it is not possible for an intermediate switch to make the decision to route a packet through a different path. The Partial Progressive Reconfiguration (PPR) [27] and Skyline [25] methods to repair an uncorrected up* / down* graph and compute a new graph were base on spanning tree.PPR transforms the invalid up*/down* graph into valid sub regions that together form a valid up*/down* graph. Skyline is a technique to identify the region of the network that must be reconfigured after the change. The PPR and Skyline were designed for networks that use distributed routing, in which forwarding decisions are taken locally by each switch in the packet path. Finally the very ancient method of reconfiguration is static [26] one. When a static reconfiguration scheme is used deadlock situations are not an issue since packets that are routed according to the old routing function and packets that are routed according to the new routing function are not simultaneously present in the network. This method is not a efficient one because network down times are high. Virtual channel [21] is also another static method but it requires additional resources.

IV. LIMITATIONS ON RECONFIGURATION

Even though many solutions for interconnection network to recover from link failures have been proposed, they still have several limitations as follows. First, resource-allocation algorithms can provide (theoretical) guidelines for initial network resource planning. However, even though their approach provides a comprehensive and optimal network configuration plan, they often require "global" configuration changes, which are undesirable in case of frequent local link failures. Next, a *greedy* channel-assignment algorithm can reduce the requirement of network changes by changing settings of only the faulty link(s).

However, this greedy change might not be able to realize full improvements, which can only be achieved by considering configurations of neighboring mesh routers in addition to the faulty link(s). Third, fault-tolerant routing protocols, such as local rerouting or multipath routing can be adopted to use network-level path diversity for avoiding the faulty links. However, they rely on detour paths or redundant transmissions, which may require more network resources than link-level network reconfiguration. To overcome all, SRS protocol includes a monitoring protocol that enables to perform real-time failure recovery in conjunction with the planning algorithm.

The accurate link-quality information from the monitoring protocol is used to identify network changes that satisfy applications' new QoS demands or that avoid propagation of QoS failures to neighboring links. First, SRS's planning algorithm effectively identifies reconfiguration plans that maximally satisfy the applications' QoS demands, accommodating twice more flows than static assignment. Second, SRS avoids the ripple effect via QoS-aware reconfiguration planning, unlike the greedy approach. Third, SRS's local reconfiguration improves network throughput and channel efficiency over the local rerouting scheme.

V. PROPOSED NETWORK MODEL AND ASSUMPTIONS

The proposed network model is assumed to consist of mesh nodes, links, and one control gateway. Each mesh node and links assignments are initially made by using global channel/link assignment algorithms. During its operation, each mesh node periodically sends its local channel usage and the quality information for all outgoing links via management messages to the control gateway. Based on this information, the gateway controls the admission of requests for data flows. Then implement the SRS protocol to improve network performance during reconfiguration.

A. Sovereign Reconfiguration System (SRS)

We first present the overview of SRS protocol. Then, we detail the SRS's reconfiguration algorithms. Finally, we discuss the functionality of SRS protocol.

1) Overview

SRS is a distributed system that is easily deployable in interconnection networks. Running in every mesh node, SRS supports self re-configurability via the following distinct features.

• *Localized reconfiguration*: SRS generates reconfiguration plans that allow for changes of network configurations only in the vicinity where link failures occurred while retaining configurations in areas remote from failure locations.

• *QoS-aware planning*: SRS effectively identifies QoS-satisfiable reconfiguration plans by: 1) estimating the QoS satisfiability of generated reconfiguration plans; and 2) deriving their expected benefits in channel utilization.

The main modules of SRS protocol is shown in Fig. 2.

2) Algorithm of SRS Protocol

```
Algorithm: SRS Operation at mesh node
 Monitoring
1: for every link do
2: measure link-cost using passive monitoring;
3: end for
4: send monitoring results to a gateway;

Failure detection and group formation

5: if link violates link requirements then
6: request a group formation on channel of link;
7: end if
8: participate in a leader election if a request is
   received;

 Planning
9: if node is elected as a leader then
10: send a planning request message to a gateway;
11: else if node is a gateway then
12. synchronize requests from reconfiguration
    groups
13: generate a reconfiguration plan;
14: send a reconfiguration plan to a leader;
15: end if
```

```
Reconfiguration
16: if includes changes of node then
17: apply the changes to links;
18: end if
19: relay to neighboring members, if any
```

3) Functionality of SRS

• *Autonomous reconfiguration via link-quality monitoring*: SRS accurately monitors the quality of links of each node in a distributed manner. Furthermore, based on the measurements and given links, QoS constraints, SRS detects local link failures and autonomously initiates network reconfiguration.

• *Cross-layer interaction*: SRS actively interacts across the network and link layers for planning.

This interaction enables SRS to include a rerouting for reconfiguration planning in addition to link-layer reconfiguration. SRS can also maintain connectivity during recovery period with the help of a routing protocol.

Algorithm describes the operation of SRS as shown in Fig. 3. First, SRS in every mesh node monitors the quality of its outgoing links at every (e.g., 10s) and reports the results to a gateway via a management message.

Second, once it detects a link failure(s), SRS in the detector node(s) triggers the formation of a group among local mesh routers that use a faulty channel, and one of the group members is elected as a leader using the well-known bully algorithm for coordinating the reconfiguration.

Third, the leader node sends a planning-request message to a gateway. Then, the gateway synchronizes the planning requests—if there are multiple requests—and generates a reconfiguration plan for the request. Fourth, the gateway sends a reconfiguration plan to the leader node and the group members. Finally, all nodes in the group execute the corresponding configuration changes, if any, and resolve the group. We assume that during the formation and reconfiguration, all messages are reliably delivered via a routing protocol and per-hop retransmission timer. In what follows, we will detail each of these operations, including

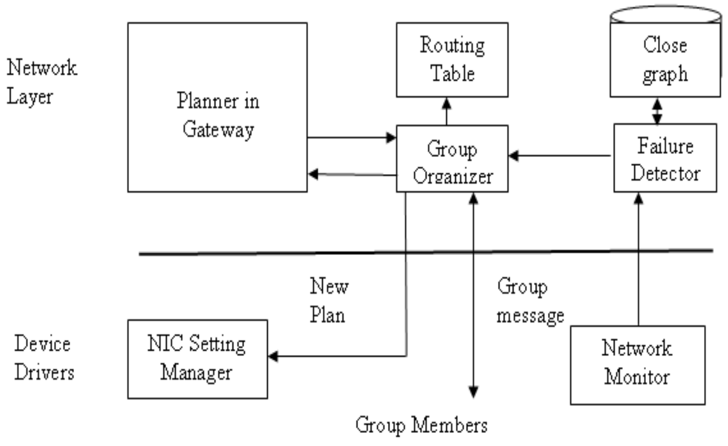

Fig. 2: Block diagram of SRS

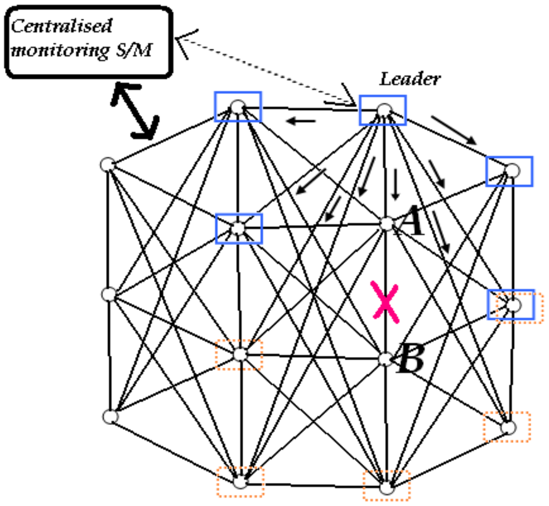

Fig. 3: Functionality of SRS

how to generate reconfiguration plans, how to monitor link conditions such as bandwidth and how much overhead SRS generates for the monitoring and for maintaining a reconfiguration group.

4) Planning for Localized Network Reconfiguration

The core function of SRS is to systematically generate localized reconfiguration plans. A reconfiguration plan is defined as a set of links' configuration changes (e.g., channel switch, link association) necessary for a network to recover from a link(s) failure on a channel, and there are usually multiple reconfiguration plans for each link failure. Existing channel-assignment and scheduling algorithms seek "optimal" solutions by considering tight QoS constraints on all links, thus requiring a large configuration space to be searched and hence making the planning often an NP-complete problem.

In addition, change in a link's requirement may lead to completely different network configurations. By contrast, ARS systematically generates reconfiguration plans that localize network changes by dividing the reconfiguration planning into three processes—feasibility, QoS, satisfiability, and optimality—and applying different levels of constraints, SRS first applies connectivity constraints to generate a *set* of feasible reconfiguration plans that enumerate feasible channel, link, and route changes around the faulty areas, given connectivity and link-failure constraints.

Then, within the set, SRS applies strict constraints (i.e., QoS and network utilization) to identify a reconfiguration plan that satisfies the QoS demands and that improves network utilization most.

Feasible Plan Generation: Generating feasible plans is essentially to search all legitimate changes in links' configurations and their combinations around the faulty area. Given multiple routes, SRS identifies feasible changes that help avoid a local link failure but maintain existing network connectivity as much as possible. However, in generating such plans, SRS has to address the following challenges.

Maintaining network connectivity and utilization: While avoiding the use of the faulty channel, SRS needs to maintain connectivity with the full utilization of resources. Because each node can associate itself with multiple neighboring nodes, a change in one link triggers other neighboring links to change their settings. To coordinate such propagation, SRS takes a *two-step* approach.

SRS first generates feasible changes of each link using the primitives, and then combines a set of feasible changes that enable a network to maintain its own connectivity. Furthermore, for the combination, SRS maximizes the usage of network resources by making each associate node itself with at least one link and by avoiding the use of same (*redundant*) node.

Controlling the scope of reconfiguration changes: SRS has to limit network changes as *local* as possible, but at the same time it needs to find a locally optimal solution by considering more network changes or scope. To make this tradeoff, SRS uses a hop reconfiguration parameter. Starting from a faulty link(s), SRS considers link changes within the first hops and generates feasible plans. If SRS cannot find a local solution, it increases the number of hops so that SRS may explore a broad range of link changes. Thus, the total number of reconfiguration changes is determined on the basis of existing configurations around the faulty area as well as the value of the number of hops.

VI. CONCLUSION

This paper proposed sovereign reconfiguration system (SRS) that enables to autonomously recover from link failures during reconfiguration. SRS will generate an effective reconfiguration plan that requires only local network configuration changes by exploiting channel, and path diversity. Furthermore, SRS will effectively identify reconfiguration plans that satisfy applications' QoS constraints, admitting up to two times more flows than static assignment, through QoS aware planning. SRS will decouple network reconfiguration from flow assignment and routing. Reconfiguration might be able to achieve better performance if two problems are jointly considered. Even though there have been a couple of proposals to solve this problem they only provide theoretical bounds without considering practical system issues. Even though its design goal is to recover from network failures as a best-effort service, SRS is the first step to solve this optimization problem.

VII. REFERENCES

[1] K. Malavalli, et al. Fibre Channel Switch Fabric-2 (FC-SW-2). NCITS 321- 200x T11/Project 1305-D/Rev 4.3 Specification, pages 57_74, March 2000.

[2] F. Petrini, W.C. Feng, A. Hoisie, S. Coll, and E. Frachtenberg. The Quadrics Network: High-Performance Clustering Technology. IEEE Micro, 22(1):213, January-February 2002.

[3] T.M. Pinkston, A.F. Benner, M. Krause, I.M. Robinson, and T. Sterling. InfiniBand: The "De Facto" Future Standard for System and Local Area Networks or Just a Scalable Replacement for PCI Buses? Cluster Computing, 6(2):95_104, April 2003.

[4] W. Dally and B. Towles. Route Packets, NotWires: On-Chip Interconnection Networks. In Proceedings of the Design Automation Conference (DAC), pages 684_689. ACM, June 2001.

[5] M.B. Taylor, W. Lee, S. Amarasinghe, and A. Agarwal. Scalar Operand Networks: On-Chip Interconnect for ILP in Partitioned Architectures. In Proceedings of the 9th International Symposium on High-Performance Computer Architecture, pages 341_353. IEEE Computer Society Press, February 2003.

[6] W.H. Ho and T.M. Pinkston. A Methodology for Designing Efficient On-Chip Interconnects on Well-Behaved Communication Patterns. In Proceedings of the 9th International Symposium on High-Performance Computer Architecture, pages 377_388. IEEE Computer Society Press, February 2003.

[7] E. Baydal, P. Lopez, and J. Duato. A Simple and Efficient Mechanism to Prevent Saturation in Wormhole Networks. In Proceedings of the 14th In- ternational Parallel and Distributed Processing Symposium, pages 617_622, 2000.

[8] M. Thottethodi, A.R. Lebeck, and S.S. Mukherjee. Self-Tuned Congestion Control for Multiprocessor Networks. In Proceedings of the 7th International Symposium on High Performance Computer Architecture, January 2001.

[9] L.-S. Peh andW. Dally. Flit-Reservation Flow Control. In Proceedings of the 6th International Symposium on High Performance Computer Architecture, pages 73_84. IEEE Computer Society Press, January 2000.

[10] J. Duato. A New Theory of Deadlock-free Adaptive Routing in Wormhole Networks. IEEE Transactions on Parallel and Distributed Systems, 4(12):1320_1331, December 1993.

[11] J. Duato. A Necessary and Suf_cient Condition for Deadlock-free Adaptive Routing in Wormhole Networks. IEEE Transactions on Parallel and Distributed Systems, 6(10):1055_1067, October 1995.

[12] J. Duato and T.M. Pinkston. A General Theory for Deadlock-Free Adaptive Routing Using a Mixed Set of Resources. IEEE Transactions on Parallel and Distributed Systems, 12(12):1219_1235, December 2001.

[13] S.Warnakulasuriya and T.M. Pinkston. AFormalModel of Message Blocking and Deadlock Resolution in Interconnection Networks. IEEE Transactions on Parallel and Distributed Systems, 11(2):212_229, March 2000.

[14] S. Warnakulasuriya and T.M. Pinkston. Characterization of Deadlocks in k-ary n-cube Networks. IEEE Transactions on Parallel and Distributed Systems, 10(9):904_921, September 1999.

[15] S. Warnakulasuriya and T.M. Pinkston. Characterization of Deadlocks in Irregular Networks. Journal of Parallel and Distributed Computing, 62(1):61_ 84, January 2002.

[16] T.M. Pinkston, R. Pang, and J. Duato. Deadlock-Free Dynamic Reconfiguration Schemes for Increased Network Dependability. IEEE Transactions on Parallel and Distributed Systems, 14(8):780_794, August 2003.

[17] V. Puente, R. Beivide, J.A. Gregorio, J.M. Prellezo, J. Duato, and C. Izu. Adaptive bubble router: A design to improve performance in torus networks. In Proceedings of the 28th International Conference on Parallel Process- ing (28th ICPP'99), Aizu-Wakamatsu, Fukushima, Japan, September 1999. University of Aizu.

[18] J.M. Martinez, P. Lopez, and J. Duato. FC3D: Flow Control Based Distributed Deadlock Detection Mechanism for True Fully Adaptive Routing in Wormhole Networks. IEEE Transactions on Parallel and Distributed Systems, 14(8):765_779, August 2003.

[19] S.S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D.Webb. The Alpha 21364 Network Architecture. In Symposium on High Performance Interconnects (HOT Interconnects 9), pages 113_117. IEEE Computer Society Press, August 2001.

[20] W. Barrett et al. An Overview of the Blue-Gene/L Supercomputer. In Proceedings of the 2002 ACM/IEEE Conference on Supercomputing, CD ROM, November 2002.

[21] J. Flich, P. Lopez, M.P. Malumbres, and J. Duato. Boosting the Performance of Myrinet Networks. IEEE Transactions on Parallel and Distributed Systems, 13(7):693_709, July 2002.

[22] Y.H. Song and T.M. Pinkston. A New Mechanism for Congestion and Deadlock Resolution. In The 2002 International Conference on Parallel Processing, pages 81_90. IEEE Computer Society, August 2002.

[23] C.B. Stunkel et al. The SP2 high-performance switch. IBM Systems Journal, 34(2):185_204, 1995.

[24] J.F. Martinez, J. Torrellas, and J. Duato. Improving the Performance of Bristled CC-NUMA Systems Using Virtual Channels and Adaptivity. In Proceedings of 13th International Conference on Supercomputing, June 1999.

[25] Y.H. Song and T.M. Pinkston. A Progressive Approach to Handling Message-Dependent Deadlock in Parallel Computer Systems. IEEE Transactions on Parallel and Distributed Systems, 14(3):259_275, March 2003.

[26] InfiniBandTM Architecture Specification Volume 1. InfiniBand Trade Association, October 24, 2000.

[27] O. Lysne and J. Duato. Fast Dynamic Reconfiguration in Irregular Networks. In The 2000 International Conference on Parallel Processing, pages 449_458. IEEE Computer Society, August 2000.

[28] F.J. Quiles, J.L. Sanchez, R. Casado, A. Bermudez and J. Duato. A protocol for deadlock-free dynamic reconfiguration in high-speed local area networks. Special Issue on Dependable Network Computing. IEEE Transactions on Parallel and Distributed Systems, 12(2):115_132, February 2001.

[29] R. Pang, T.M. Pinkston and J. Duato. Dynamic Reconfiguration of Networks with Distributed Routing: The Single Scheme. In Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), pages 2042_2048, June 2001.

[30] Antonio Robles-Gomez, Aurelio Bermudez, and Rafael Casado, A deadlock-free dynamic reconfiguration scheme for source routing networks using close up*/down* graphs , IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 10, October 2011.

[31] Olav Lysne, Jose Miguel Montanana, Jose Flich, Jose Duato, Timothy Mark Pinkston nad Tor Skeie, "An Efficient and Deadlock Free Network Reconfiguration Protocol", IEEE Transactions on Computers, vol. 57, No. 6, June 2008.

[32] Dong Xiang, Yueli Zhang and Yi Pan, "Practical Deadlock Free Fault Tolerant Routing in Meshes Based on the Planar Network Fault Model", IEEE Transactions on Computers, vol. 58, No. 6, May 2008.

[33] Jian Qiu, Mohan Gurusamy,, Kee Chaing Chua, and Yong Liu," Local Restoration with Multiple Spanning Trees in Metro Ethernet Networks", IEEE/ACM Transaction on networking,vol.19,No.2, April 2011.