University of Castilla-La Mancha

A publication of the
Computing Systems Department

Deadlock-Free Dynamic Network Reconfiguration Based on Close Up*/Down* Graphs

by

Antonio Robles-Gomez, Aurelio Bermúdez,
Rafael Casado, and Åshild Grønstad Solheim

Technical Report **#DIAB-08-02-1** February 2008

Part of this work has been submitted to the Euro-Par 2008 Conference

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS
ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD DE CASTILLA-LA MANCHA
Campus Universitario s/n
Albacete – 02071 – Spain
Phone +34.967.599200, Fax +34.967.599224

Deadlock-Free Dynamic Network Reconfiguration Based on Close Up*/Down* Graphs*

Antonio Robles-Gómez[1], Aurelio Bermúdez[1], Rafael Casado[1], and Åshild Grønstad Solheim[2,3]

[1] Instituto de Investigación en Informática de Albacete (I^3A)
Universidad de Castilla-La Mancha,
02071 Albacete, Spain
{arobles, abermu, rcasado}@dsi.uclm.es

[2] Networks and Distributed Systems Group
Simula Research Laboratory,
Lysaker, Norway
aashig@simula.no

[3] Department of Informatics
University of Oslo,
Oslo, Norway

Abstract. Current high-performance distributed systems use a switch-based interconnection network. After the occurrence of a topological change, a management mechanism must reestablish connectivity between network devices. This mechanism discovers the new topology, calculates a new set of routing paths, and updates the routing tables within the network. The main challenge related to network reconfiguration (the change-over from one routing function to another) is avoiding deadlocks. Former reconfiguration techniques significantly reduce network service. In addition, most recent proposals either need extra network resources (such as virtual channels) or their computation complexities are prohibitive. For up*/down* routed networks we propose a new reconfiguration method that supports a virtually unaffected network service at a minor computational cost. This method is

* This work has been jointly supported by the Spanish MEC and European Comission FEDER funds under grants Consolider Ingenio-2010 CSD2006-00046 and TIN2006-15516-C04-02; by JCCM under grant PBC05-007-1; by UCLM under grant TC20070061. It has also been supported by a FPI grant under the Spanish MEC TIC2003-08154-C06-02.

suitable for both source and distributed routing networks, and does neither restrict the injection of packets nor the updating of routing tables during the topology-change assimilation.

1 Introduction

The communication subsystem of most modern high-performance distributed systems includes a switch-based interconnection network. In recent years, several different interconnect technologies have been proposed, some based on source routing (Myrinet 2000 [8], Advanced Switching [1]), and others based on distributed routing (InfiniBand [5]).

In this kind of interconnects, after the occurrence of a fault or after the removal or insertion of a component in the network, a new routing function must be calculated based on the new topology that results from the change. For the purpose of path-computation the interconnection network is usually represented as a directed graph. The tasks of discovering the network topology and computing the new routing function are typically performed by a centralized management entity which is called *mapper* in Myrinet, *subnet manager* in InfiniBand, and *fabric manager* in Advanced Switching.

In the literature, the process of replacing one routing function with another is traditionally referred to as *network reconfiguration*. It is well-known that, although both routing functions are by themselves deadlock-free, updating fabric paths in an uncontrolled way may lead to deadlock situations since packets that belong to one of the routing functions may take turns that are not allowed in the other routing function.

Early reconfiguration mechanisms (designed for Autonet [15] and Myrinet [2] networks) solved this problem by emptying the network of packets before replacing the routing function.

Such a simple approach is referred to as *static reconfiguration*, and has a negative impact on the network service availability since for a period of time the network cannot accept packets.

Recently, several schemes have been proposed for distributed routing systems in order to increase network availability during the change-over from one routing function to another. These mechanisms allow injection of data traffic while the routing function is being updated, and are known as *dynamic reconfiguration* techniques. In order to guarantee deadlock-freedom the dynamic reconfiguration schemes are, in general, more advanced than the static reconfiguration schemes are.

In this paper we propose a new dynamic reconfiguration strategy which is based on the up*/down* routing algorithm [15] and is applicable to both source and distributed routing networks. The main idea behind the proposed strategy is to transform an invalid up*/down* graph (that results from a topological change) into a valid up*/down* graph, while ensuring that turns that are prohibited by one of the routing functions are not allowed by the other routing function. Section 2 provides background information on the up*/down* routing algorithm and on previous studies of reconfiguration of up*/down* routing networks. Section 3 formally introduces the concept of a close graph and presents the new reconfiguration strategy in detail. In this strategy a new directed graph (which is close to the old directed graph) is first computed by a centralized management entity. Then a new routing function is derived by considering both the old and the new graph. Thereafter, the new routing function is asynchronously distributed to the routing elements. This enables a dynamic update of the routing function. In Section 4 the performance of the proposed reconfiguration strategy is evaluated, before we conclude in Section 5.

2 Network Reconfiguration in Up*/down*-based Interconnects

2.1 Up*/down Routing

Our new reconfiguration scheme is based on up*/down* [15], a popular deadlock-free routing algorithm suitable for regular and irregular topologies. This algorithm is based on a cycle-free assignment of direction to the operational links in the network. This assignment is always possible, regardless of network topology. For each link, a direction is named *up* and the opposite one is named *down*. To avoid deadlocks, legal routes never use a link in the *up* direction after having used one in the *down* direction. Messages can cross zero or more links in the *up* direction, followed by zero or more links in the *down* direction. In this way, cycles in the channel dependency graph [4] are avoided, thus preventing deadlock.

A *sink node* [3] is a node in a directed graph that is not the source of any link. The up*/down* routing algorithm requires the existence of a single sink node in the graph. The reason is that there are no legal routes between two sink nodes because each possible route would require *down* to *up* transitions. This restriction is required for network connectivity.

A *break node* is a node that is the source of two or more links. In the up*/down* routing algorithm, these nodes prevent certain connections (input port - output port) from being used by the messages crossing them. These restrictions are necessary for deadlock freedom. There must exist one break node in every cycle, but its position is unrestricted.

In up*/down* routing, the associated directed graph will contain one and only one sink node. Additionally, that graph will be acyclic. A directed graph that is acyclic and contains a single

sink node is called a *correct graph*. A correct graph may include several break nodes within its topology, as many as necessary to break all the cycles.

Obviously, an *incorrect graph* is one that does not meet the restrictions imposed in the previous definition. This implies the absence of a sink node, the existence of more than one sink node, or the existence of cycles. If there is no sink node, then the graph will contain one or more cycles and up*/down* routing cannot guarantee deadlock freedom. If there are several sink nodes, then up*/down* routing cannot guarantee network connectivity. There is always at least one false break node between two sink nodes. A *false break node* is a break node in which two links with the *down* end connected to it do not belong to the same cycle in the undirected graph of the network. A false break node splits the network into two unreachable regions. Obviously, a correct graph contains no false break nodes.

Several alternative algorithms for building an up*/down* directed graph have been proposed in the literature. Traditional proposals are based on the computation of a network spanning tree rooted in one of the nodes, by using a breadth-first search (BSF) [15], a depth-first search (DFS) [14], or a propagation-order (POST) [13] strategy. Then, the links that are part of the spanning tree are given *up* directions pointing towards the sink node, whereas the direction of the links that are not part of the spanning tree must be carefully assigned in order to avoid introducing any cycle of *up* or *down* links.

2.2 Network Reconfiguration

With respect to the network reconfiguration, deadlock is not an issue when a static reconfiguration scheme is used 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. Updating the routing function while the network remains up and running, on the other hand, requires more advanced reconfiguration schemes. Next, we briefly depict some dynamic reconfiguration techniques proposed for networks that use distributed routing.

The Partial Progressive Reconfiguration (PPR) [3] and Skyline [6] approaches aim to repair an uncorrected up*/down* graph which includes several sink nodes. PPR computes the new graph in a distributed way, where an invalid up*/down* graph is transformed into valid sub-regions that constitutes a valid up*/down* graph. Skyline is a technique to identify the region of the network that must be reconfigured after the change. Then, any connection method –for example PPR– can be applied over that part of the graph.

On the other hand, Double Scheme [10] and Simple Reconfiguration [7] can be used by any routing algorithm, including up*/down*. Double Scheme requires two disjoint sets of virtual channels to separate packets routed according to the old routing function from packets routed according to the new routing function. Simple Reconfiguration introduces a special packet (called a *token*) to govern the transition from one routing function to another. Deadlock is 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.

3 A Dynamic Reconfiguration Scheme Based on Close Graphs

3.1 Deadlock-Free Routing Update Based on Close Graphs

This section presents our new reconfiguration method that, after a topology-change, calculates a new routing function which ensures that packets that belong to the new routing function cannot

take turns that are prohibited in the old routing function, and vice versa. This guarantees that packets that belong to the old and new routing functions can unrestrictedly coexist in the network without causing deadlocks. The method is based on the concept of close up*/down graphs, which will be defined shortly. This section also presents lemmas to support that, when an up*/down* graph for the new topology is designed close to the up*/down* graph for the previous topology, the routing function can be updated without the risk of transient deadlocks.

Definition 1. Assume that two up*/down* directed graphs, G_1 and G_2, represent the same network topology. Then, G_1 and G_2 are *close* iff each cycle in G_1 and G_2 is broken in the same node or in neighboring nodes.

Lemma 1. Assume that an up*/down* directed graph G_1 is incorrect due to the presence of several sinks. Then, it is always possible to obtain a correct graph G_2 which is close to G_1.

Proof. In a correct up*/down* graph, each possible cycle must contain at least one node with two incoming up-links and at least one node with two outgoing up-links. For each possible cycle in G_2, we have three options with respect to break node placement (in the same node as in G_1, or in one of the two neighboring nodes). Thus, it is always possible to construct a correct graph G_2 which is close to G_1.

Lemma 2. Assume that a correct up*/down* directed graph G includes a break node $n \in G$. Then, after the suppression of one of the outgoing up-links of n, G remains correct and connected.

Proof. It is not possible to generate a new cycle by suppressing one of the outgoing up-links of a break node. Also, it is not possible to generate a new sink node by suppressing one of the outgoing up-links of a break node (by definition a break node has more than one outgoing up-link).

Lemma 3. Assume that two up*/down* directed graphs G_{old} and G_{new} exist, where G_{new} is correct and close to G_{old}. Then, it is possible to obtain an up*/down* routing function R that satisfies the routing-restrictions (prohibited down-link to up-link transitions) imposed by both G_{old} and G_{new}.

Proof. Assume that G_{sub} is a subgraph of G_{new} in which each link that connects a break node in G_{old} with the corresponding break node in G_{new} is suppressed, and that G_{new} is a correct up*/down* graph. Then, according to lemma 2, we can guarantee that G_{sub} is also a correct up*/down* graph.

Thus, it is possible to define a fully connected deadlock-free routing function R_{sub} over G_{sub}. R_{sub} satisfies the routing-restrictions of both G_{old} and G_{new} since all the break nodes either have the same locations or have been removed.

3.2 Construction of G_{new} close to G_{old}

We have shown above that a new up*/down* graph G_{new} for the current topology can always be constructed close to the previous up*/down* graph G_{old} such that deadlocks cannot form during the routing function update. This section presents an algorithm for the construction of G_{new} close to G_{old}, but first we define some concepts.

Definition 2. An *exploration process* is the procedure in which the manager goes through the network to discover the links between nodes and the direction assigned to them.

Definition 3. Assume that the exploration process from a node n_1 reaches another node n_2. Then, this link is explored in *downward direction* (from the point of view of the exploration process) if its *up*/down** direction is $n_1 \leftarrow n_2$. Similarly, the link is explored in *upward direction* if the direction is $n_1 \rightarrow n_2$. These cases will be referred to as l^{downward} and l^{upward}, respectively.

Definition 4. A *frontier link* is a link currently under evaluation by the exploration process (such as the link from n_1 to n_2 above).

Definition 5. Two directed links are *partners* if they have the same source node.

Definition 6. Assume that the exploration process has reached a node n, that l is a link connected to n, and that l has not been processed from a previously visited node. Then, a *neighboring link* of l is another link connected to n which has not been processed from a previously visited node.

Definition 7. A node is *ancestor node* of a link if, starting from that link, the node can be reached by traversing only up-links.

Definition 8. Two links are *relatives* if they have common ancestor nodes.

Algorithm.

Let X be the set of frontier links and let l denote a link between two nodes in the network.

To construct G_{new} the manager starts an exploration process in one of the sink nodes of G_{old}:

$\forall l$ connected to the start node of the exploration process, evaluate inclusion of l in X (*see below*).

While X is not empty do:

- If $\exists l \in$ X explored in downward direction, then
  - Remove l from X.
  - $\forall l_1$ which neighboring link of l evaluate inclusion of l_1 in X.
- Else the following invariant applies: $\exists l_1 \in$ X explored in upward direction which guarantees that $!(\exists l_2 \in X)$ such that
  - l_1 is partner of l_2.
  - The destination of l_1 is ancestor node of l_2.
  - With l_1 do:
    - Remove l_1 from X.
    - $\forall l_3$ which is neighboring link of l_1, evaluate inclusion of l_3 in X.
    - If $\exists l_2$ which is partner of l_1 and relative to $l_3 \in$ X, then remove l_3 from X.
    - Change the direction of l_1.

Evaluation before inclusion of a link in X.

Before including a link l in X we need to do the following evaluation:

- If l has previously belonged to X, then discard l.
- Else if $l^{upward} \in$ X and l is now explored in downward direction, then update the direction of the link such that $l^{downward} \in$ X.
- Else if $!(l \in X)$, then
  - If there is no direction assigned to l, then assign the downward direction to l.
    - If l contributes to form a cycle, that is, the source of l is also its ancestor node, then change the direction of l.
  - Else l is included in X with the direction found by the exploration process.

4 Performance Evaluation

In this section we present the simulation study that was undertaken to evaluate our novel dynamic reconfiguration scheme, presented in the previous section. For performance assessment this scheme was compared to an earlier static reconfiguration scheme, proposed in [12] for source routing networks based on the Advanced Switching technology. We compare the time required by each scheme to completely assimilate a topology change, and also their impact on the network service. Before presenting the simulation results, the experiment setup is described.

4.1 Experiment Setup

The simulation model [11] used for this work was developed with the OPNET Modeler [9], and embodies the physical and link layers of Advanced Switching, allowing the simulation of several network designs. The model implements 16-port multiplexed virtual cut-through switches, where each port uses a 1x lane (2.5 Gbps), and endpoints are connected with a single port.

This model provides the necessary support to design fabric management mechanisms as defined in the ASI specification [1]. It includes management entities, device capabilities, and management packets. In addition, the model considers the time required by the FM and fabric devices to process incoming management packets and perform tasks associated with the reception of such packets.

We evaluated several regular fabric topologies, including meshes, tori, and fixed-arity fat-trees. A complete list is presented in Table 1. For meshes and tori, we assume that each border switch has an endpoint attached. Although the management mechanisms analyzed do not require the use of several VCs, we used four virtual channels (VCs) per fabric port. According to the

Advanced Switching specification, a minimum of two VCs per port must be in place, where one VC is dedicated to management traffic, and the remainder to data traffic [1]. Management traffic gets higher scheduling priority than data traffic in switches and therefore it will always take a similar time to assimilate a topological change, independently of the amount of data packets in the network. The size of the input and output buffers associated with each VC is 8 Kbytes.

Table 1. The topologies evaluated

Topology	Switches	Endpoints	Total Devices
3×3 mesh, 3×3 torus	9	8	17
4×4 mesh, 4×4 torus	16	12	28
6×6 mesh, 6×6 torus	36	20	56
8×8 mesh, 8×8 torus	64	28	92
9×9 torus	81	32	113
4-port 2-tree	6	8	14
4-port 3-tree	20	16	36
4-port 4-tree	56	32	88
8-port 2-tree	12	32	44

For all simulations, the data packet length was fixed at 512 bytes. The traffic load that was applied was dependent on the fabric topology and the number (and size) of available VCs, representing 50% of the saturation rate in each case. The packet generation rate was uniform. The *perfect shuffle*, *bit reversal*, *matrix transpose*, and *uniform* traffic patterns were applied to obtain packet destination endpoints. The results for the different traffic patterns were very similar, and the results for the uniform traffic pattern were selected for presentation.

For each simulation there is an initial transient period in which fabric devices are activated, the manager gathers the original topology, and the fabric operation approaches its steady state. Later, a topological change (either the addition or removal of a randomly chosen fabric switch) is scheduled at time 2.0 seconds (after the fabric starts up). In order to increase the accuracy of the results, each experiment was repeated several times for each fabric topology. The number of

simulation runs for each topology is about 15% of the number of physical nodes (both switches and endpoints) for both switch additions and removals, and averages have been drawn from the solution set and presented.

4.2 Impact on the Management Time

We first analyze the time required to calculate the new up*/down* graph after a topology change (Fig. 1). For the proposed dynamic reconfiguration scheme we measure the time spent in the "Topology + Close Graph" phase, whereas for the static reconfiguration scheme the time spent in the "Topology + New Graph" phase is measured. Both schemes first discover the new fabric topology that results from the topology change. In the next step, the static scheme assigns new link directions without taking the previous up*/down* graph into account, whereas the dynamic scheme transforms the old graph into a close graph by following the algorithm described in the previous section. We compare the duration of the topology discovery and graph calculation phases for the two schemes. Fig. 1 confirms that the overhead of our proposed dynamic scheme is small when compared to the static scheme. In addition, the duration of the topology discovery and graph calculation phases is short when compared to the path computation and

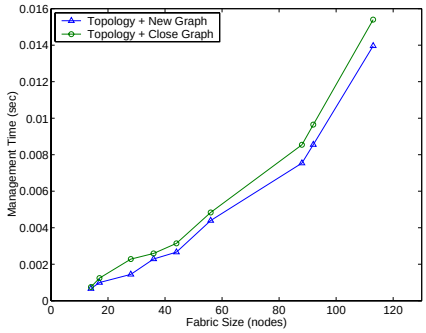
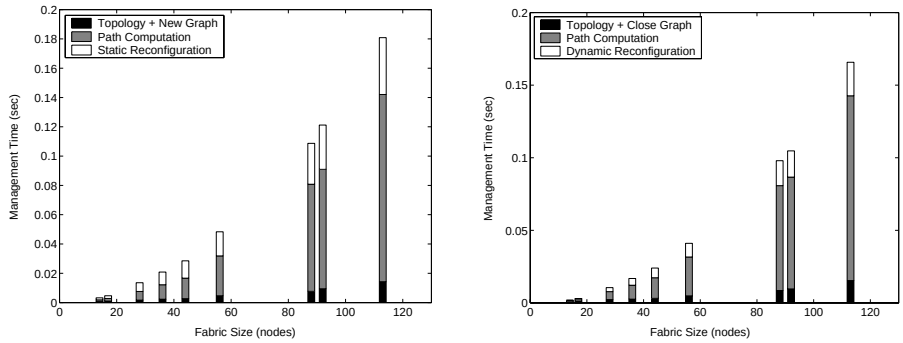


Fig. 1. The time required by the manager to build a new up*/down* graph for our dynamic reconfiguration scheme (Topology + Close Graph) and the static reconfiguration scheme (Topology + New Graph) as a function of fabric size



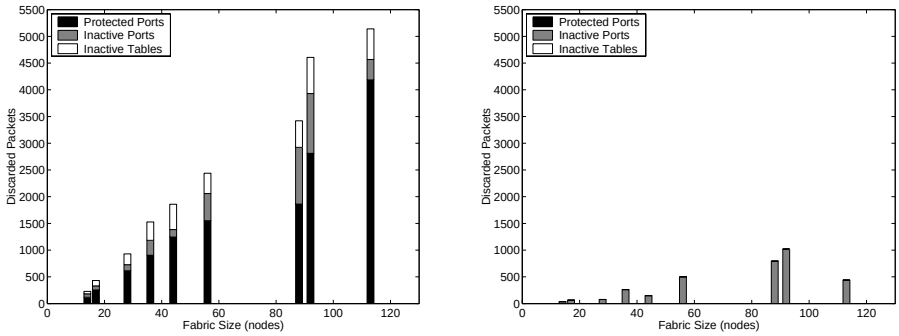
(a) Static reconfiguration (b) Dynamic reconfiguration

Fig. 2. The time required by the static and dynamic reconfiguration schemes to completely assimilate a topology change (both switch removal and addition are included)

reconfiguration phases (Fig. 2).

As a function of the fabric size, Fig. 2 shows the time required by the static and dynamic reconfiguration schemes to discover the new topology and build the new routing graph, compute the new set of routes, and update the routing tables after a topology change (both switch removal and addition are included). We observe that the "Dynamic Reconfiguration" phase spends less time updating the routing tables than the "Static Reconfiguration" phase does. The "Static Reconfiguration" phase comprises four steps: deactivation of the fabric ports to only allow management packets into the network, removal of the information stored at endpoint routing tables, distribution of the new routing paths to the endpoints and, finally, the reactivation of the fabric ports to allow data packets into the network. Routing tables are inactive during a certain period in the path distribution step. The "Dynamic Reconfiguration" phase, on the other hand, only comprises the distribution of the new set of routes.

This also explains why the "Dynamic Reconfiguration" phase uses significantly fewer management packets than the "Static Reconfiguration" phase does (these results are not shown here due to lack of space).



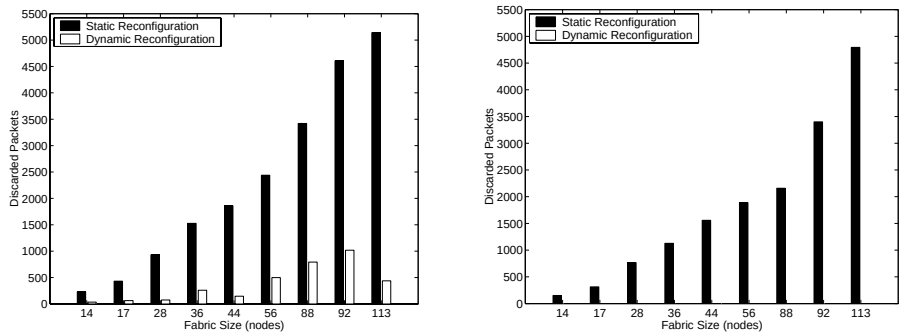
(a) Static reconfiguration (b) Dynamic reconfiguration

Fig. 3. The amount of application packets discarded as a consequence of a switch removal

4.3 Impact on the Network Service

The number of data packets that are discarded during the topology change assimilation process gives an indication of the level of service a network can provide to applications. As a function of the fabric size, Fig. 3 compares the amount of packets that are discarded due to a switch removal for the static and dynamic reconfiguration schemes. The bars in this figure represent the quantities that relate to three different reasons for packets being discarded. They are labelled "Protected Ports", "Inactive Ports" and "Inactive Tables". The "Protected Ports" label quantifies packets that are discarded when they reach (logically) inactive ports (i.e., in the *DL_Protected* state in which ports can only receive and transmit management packets) [1]. The "Inactive Ports" label refers to packets that attempt to cross the switch that has been removed from the network, in order to reach their destination. Finally, the "Inactive Tables" label refers to the packets that can not be injected into the network due to the endpoint routing tables are empty.

The results in Fig. 3 show that the rate at which data packets are discarded is notably lower for the new dynamic reconfiguration scheme than for the static reconfiguration scheme. The explanation is that whereas all three possible reasons for discarding packets apply for the static



(a) Switch removal (b) Switch addition

Fig. 4. The amount of application packets discarded for both switch removal and addition

scheme, for the dynamic scheme packets are only discarded when they attempt to cross the removed switch.

In Fig. 4, the number of data packets discarded for the static and dynamic schemes are compared, both for a removal (Fig. 4a) and addition (Fig. 4b) of a switch. Fig. 4a summarizes the results presented in Fig. 3a and Fig. 3b (this is done to enable easy comparison of the results for a switch addition and removal). No packets are discarded due to inactive ports because no old routes include the switch that was just added. Moreover, in this case, there is no packet discarding when using the dynamic reconfiguration scheme. The reason is that it does not require deactivating fabric ports and routing tables.

To conclude the evaluation, Fig. 5 illustrates the instantaneous behavior of both the static and dynamic schemes. For both schemes, we have scheduled a removal of a switch in a 6×6 mesh at time 2.0 sec. For all plots, the x-axis represents the simulation time. The top plot shows the aggregate amount of data packets discarded during the assimilation process. The bottom two plots show instantaneous network throughput, represented by the number of data packets sent/received per second in the whole fabric.

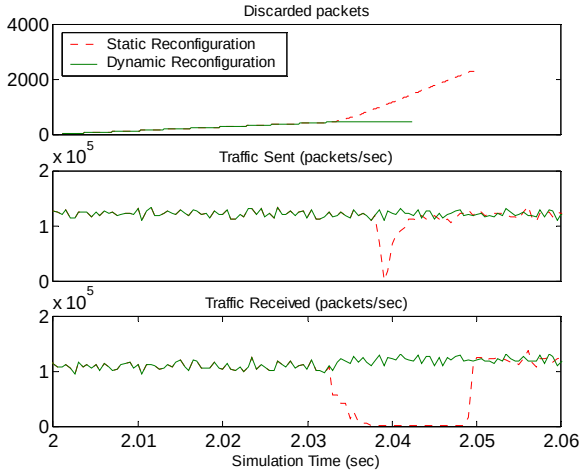


Fig. 5. The impact of a switch removal on application traffic (instantaneous results)

The static reconfiguration scheme has a detrimental effect on the network service, and we observe a gap in the instantaneous throughput plots coinciding to the reconfiguration phase. This gap is completely eliminated by using our dynamic reconfiguration scheme. This demonstrates that the introduction of the dynamic reconfiguration scheme enables an uninterrupted network service during the topology change assimilation.

5 Conclusions

We have proposed and evaluated a new deadlock-free dynamic reconfiguration mechanism for updating the routing function after a topological change in a network that applies the up*/down* routing algorithm. At a minor computational cost, the new routing function is designed to ensure that packets routed according to the old and the new routing functions can unrestrictedly coexist in the network, without the risk of forming deadlocks. Simulation results show that this significantly reduces the amount of packets that are discarded during the topology-change assimilation. From the point of view of upper-level applications, our new reconfiguration strategy virtually eliminates the problem of reduced network service availability, which is

characteristic of traditional reconfiguration proposals. In addition, our proposed strategy does not require additional fabric resources such as virtual channels, and it could easily be implemented in current commercial systems using either source or distributed routing.

References

1. Advanced Switching Interconnect Special Interest Group, Advanced Switching Core Architecture Specification Revision 1.0. December 2003, http://www.picmg.org
2. Boden, N.J., et al.: Myrinet: A gigabit per second LAN. IEEE Micro, February 1995
3. Casado, R., Bermúdez, A., Quiles, F. J., Sánchez, J. L., Duato, J.: A protocol for deadlock-free dynamic reconfiguration in high-speed local area networks. IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 2, February 2001
4. Dally, W. J., Seitz C. L.: Deadlock-free message routing in multiprocessor interconnection networks. IEEE Transactions on Computers, vol. 36, no. 5, 1987
5. InfiniBand Architecture Specification (1.2). November 2002, http://www.infinibandta.com/
6. Lysne, O., Duato, J.: Fast dynamic reconfiguration in Irregular networks. In Proc. Int. Conference on Parallel Processing, August 2000
7. Lysne, O., Montañana, J. M. Pinkston, T. M. Duato, J. Skeie, T., and Flich J.: Simple deadlock-free dynamic network reconfiguration, In Proc. of the International Conference on High Performance Computing, December 2004.
8. Myrinet, Inc.: Guide to Myrinet-2000 Switches and Switch Networks, http://www.myri.com/
9. OPNET Technologies, Inc., http://www.opnet.com/

10. Pinkston, T.M., Pang, R., Duato, J.: Deadlock-free dynamic reconfiguration schemes for increased network dependability. IEEE Transactions on Parallel and Distributed Systems, vol. 14, no. 6, June 2003
11. Robles-Gómez, A., García, E. M., Bermúdez, A., Casado, R., Quiles, F. J.: A Model for the Development of ASI Fabric Management Protocols. In Proc. of the Euro-Par 2006 Conference, September 2006
12. Robles-Gómez, A., Bermúdez, A., Casado, R., Quiles, F. J., Skeie, T., and Duato, J.: A proposal for managing ASI fabrics. To appear in Journal of System Architecture (JSA), 2008.
13. Rodeheffer, T. L., Schroeder, M. D.: Automatic reconfiguration in Autonet. SRC Research Report 77 of the ACM Symposium on Operating Systems Principles, October 1991
14. Sancho, J. C., Robles, A., Duato, J.: A new methodology to compute deadlock-free routing tables for irregular networks. In Proc. the 4th Workshop on Communication, Architecture, and Applications for Network-based Parallel Computing, January 2000.
15. Schroeder, M. D., Birrell, A. D., Burrows, M., Murray, H., Needham, R. M., Rodeheffer T. L., Satterthwate, E. H., Thacker, C. P.: Autonet: a high-speed, self-configuring local area network using point-to-point links. IEEE Journal on Selected Areas in Communications, vol. 9, no. 8, October 1991