Reconfigurable Wormhole Networks: A Realistic Approach *

José L. Sánchez[1], José M. García[2], Francisco J. Alfaro[1]

[1] Universidad de Castilla-La Mancha, Escuela Politécnica
Campus Universitario, 02071 Albacete, Spain
[2] Universidad de Murcia, Facultad de Informática
Campus de Espinardo, 30071 Murcia, Spain

Abstract. The major problem in an interconnection network is related with the contention due to message blocking. In order to avoid this problem, many alternatives have been proposed in the literature, mainly adaptive routing, random routing and dynamic network reconfiguration. Our paper deals with dynamic reconfigurable networks. Reconfigurable networks are an alternative to reduce the negative effect that congestion produces on the performance of the network. Network reconfiguration can be done in different ways. Our researches are focused on dynamic reconfiguration. This technique consists basically of placing the different processors in the network in those positions which, at each computational moment and according to the existing communication pattern among them, are more adequate for the development of such computation.
In this paper we present the foundations of reconfigurable network architecture. The reconfiguration capacity is based on a reconfiguration algorithm distributed in each node. This algorithm is based on a cost function, requires only local information and picks up the characteristics of the reconfiguration adopted technique. We discuss these features and adjust the different parameters that the reconfiguration algorithm has. We have also studied the deadlock problem in reconfigurable interconnection networks and we detail the solution adopted in our approach. Finally, we have evaluated the performance of this technique under randomized hot-spot workload.

1 Introduction

In highly parallel machines, a collection of computing nodes works in concert to solve large application problems. The nodes communicate data and coordinate their efforts by sending and receiving messages through a routing network. Consequently, the achieved performance of such machines depends critically on the performance of their routing networks. So, it is important to increase the performance of the network. To meet this objective, the designer manipulates three independent variables [4]: topology, routing and switching.

---

* This work was supported in part by Spanish CICYT under Grant TIC97-0897-C04

Many recent multicomputer networks use cut-through or wormhole routing [2, 8], a technique which reduces message latency by pipelining transmission over the channels along a message's route. A major problem found in wormhole networks is blocking situations. A message spans multiple channels which couples the channels tightly together, therefore blockage on one channel can have immediate impact on anothers. In these networks, channel coupling effects make the performance quite sensitive to blockage problem. So, this congestion can reduce the network performance in a significant rate. Several techniques have been proposed to reduce or avoid congestion, such as adaptive routing, random routing or reconfigurable networks. The adaptive routing main disadvantage is the high overhead because of information monitoring and path changing. Random routing solution has a major disadvantage in the point that it doubles in mean the length of the paths that the messages travel. Therefore, we focus on the reconfigurable network approach as an attractive method to reduce the congestion network. In this paper we propose a network reconfiguration mechanism in order to try to improve the performance of these systems. The goal of a reconfigurable network is to increase the performance by minimizing the congestion of messages in the network.

In this paper we present the foundations of reconfigurable network architecture. The reconfiguration capacity is based on a reconfiguration algorithm distributed in each node. The algorithm decides when and how the reconfiguration will take place. Reconfiguration is limited, allowing nodes to interchange their places with their neighbours and preserving the original topology. In this way, routing algorithms remain unaltered. The algorithm evaluates the communication contention and decides when the reconfiguration is more favorable. This algorithm is based on a cost function and requires only local information.

The rest of the paper is organized as follows. In section 2 we introduce the reconfigurable network architectures and we present the algorithm for dynamic reconfiguration. In section 3 we show the characteristics of the reconfiguration technique. In section 4 we study the deadlock problem in reconfigurable networks. In section 5 we show and analyze the evaluation results, and finally, in section 6 some conclusions are given.

2 Reconfigurable Networks

Most message-passing systems are based on a fixed interconnection topology. In specialized architectures, the interconnection topology is selected so that it matches the communication requirements of a specific application. For more general purpose architectures, routing mechanisms must be implemented to allow a processor to communicate with a non-neighbour processor.

A reconfigurable network is adopted in order to reduce the cost of the communication. Basically, it consists of placing the different processors in the network in those positions which, at each computational moment and according to the existing communication pattern among them, are more adequate for the development of such computation.

A reconfigurable network has the following advantages [5]:

- Programming a parallel application becomes more independent of the target architecture because the architecture adapts to the application.
- It is easy to exploit the locality in communications.
- In wormhole networks, reconfigurable architectures alleviate the congestion in due to the blocking problem. This problem is more important in networks with deterministic routing.
- Finally, there are applications which communications pattern varies over time. For these, reconfigurable architectures can be very well suited.

2.1 The Reconfiguration Algorithm

A reconfigurable network is controlled by the reconfiguration algorithm. We show an updated version of the algorithm presented in [6] which was designed to work on networks using the store-and-forward routing technique. Now, most multicomputers use wormhole routing. Therefore, we have developed a new version accounting all features of this routing algorithm. The basic idea is the following: when messages arriving by a given channel to their destination nodes have supported an important delay, the algorithm will try to put the destination node close to the site that is producing those delays, by exchanging its position with its neighbour more close to the conflict zone.

The dynamically reconfigurable system consists of several nodes, a link connection switch which allows communication among nodes and a system controller which supervises the switch configuration. We have chosen a centralized control for the dynamic network reconfiguration. A reconfiguration protocol among the nodes and the control node has been developed for handling the reconfiguration of the network. This protocol adds little message traffic to the network.

When a pair of nodes decide that it is necessary to reconfigurate the network, one of them informs the system controller and it consults about the viability of the change. If the change is possible the system controller node sends a message to the nodes in this sense. Then, both nodes inform all their neighbouring nodes that they are going to interchange their positions and therefore those nodes should stop sending messages to them.

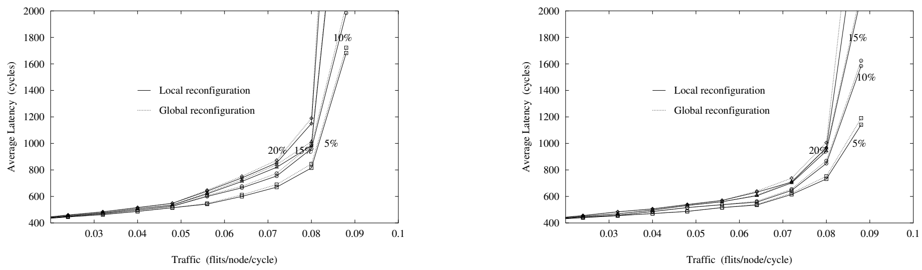
When the area affected by the change is inactive, the nodes notice to the system controller. The control node modifies the interconnection network topology, adapting it to the new circumstances. Once the new configuration has been established, the control node sends information about the new situation to the other nodes. The pair of nodes that have interchanged their positions, permit their neighbouring nodes to communicate with them again.

3 Characteristics of the Reconfiguration Technique

In this section, we are going to present several parameters we have used in our reconfiguration algorithm. We have been evaluating different approaches and adjust the different parameters to obtain the best performance for this technique. All these results have been obtained with the Pepe environment (see section 5).

The first approach is global or local reconfiguration, that is, we want to check if the possibility that several reconfiguration processes are developed in a simultaneous way it can introduce a positive factor in the global effect of the reconfiguration. Thus, we have compared the results that they are obtained considering this possibility with those that arise when this possibility is not admitted, that is to say a reconfiguration process can not activate until totally the one that is being developed in that moment does not conclude.

In figures 1 and 2 we can see the results obtained in both cases. We could think that the development of several reconfiguration processes at the same time could have a better behavior, since the nodes could be located more quickly in the most appropriate positions. However, we must keep in mind that so that a reconfiguration process concludes with a success, the area affected by the change should be completely inactive. That is to say, any message should not circulate for it. This fact causes an important congestion in the network during a certain interval of time. The originated congestion will be increased considerably when several areas, corresponding to several changes, they should be inactive. Therefore, the inconveniences have bigger weight that the advantages of maintaining several reconfiguration processes active in a simultaneous way.

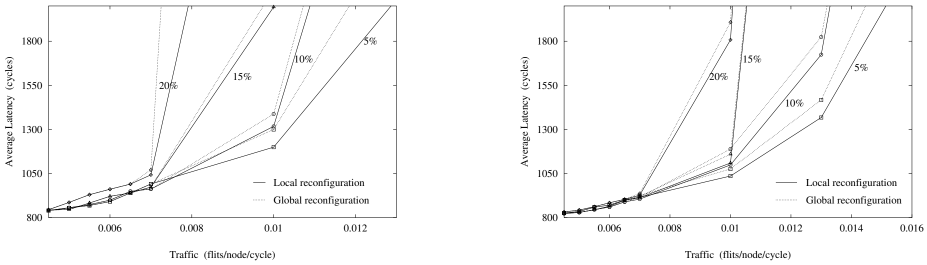


(a) One hot-spot

(b) Two hot-spots

Fig. 1. Average message latency versus traffic for 8×8 torus and crossbar switch

The second approach is in order to liberate of messages the area affected by a change. We have also considered several alternatives. In a first case, we have opted to eliminate all the messages that circulate for the channels belonging to the affected area, storing them in the intermediate nodes where they are their header flits. Once the change takes place, the messages will be again injected to the network and directed toward their true destinations. This injection will have priority on the emission of new messages. Another option consists on letting that the messages that circulate for the area of change in the moment to begin the



(a) One hot-spot

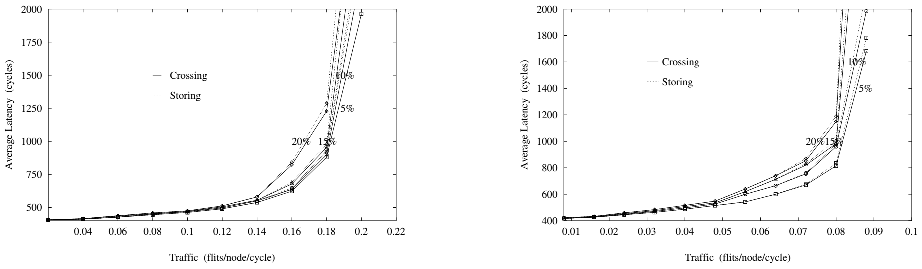
(b) Two hot-spots

Fig. 2. Average message latency versus traffic for 16×16 torus and Omega switch

reconfiguration process continue its path until they abandon it. In both cases, logically it will be prevented that other messages enter in the affected area, not assigning them the channels that they request for it.

Using the first option will be possible to liberate the channels affected more quickly than with the second. However, the latency of the stored messages will be increased.

In figure 3 the result is shown of applying both techniques on a reconfigurable system. As it can be seen, the second option offers better results. The drawback of the storing technique is the increase of the latency due to the temporary storage of the messages in intermediate nodes.



(a) 16 nodes

(b) 64 nodes

Fig. 3. Average message latency for 2D torus, one hot-spot and crossbar switch

The other characteristics have been adjusted in the same manner. So the dynamic network reconfiguration technique has the following features: It *preserves* *the topology* (after reconfiguration, the network has the same topology), it is *based on network contention* (a node can reconfigurate the network taking into account information about the contention in the network), it *produces a small alteration* (a node can only make an exchange with one of its neighbour nodes), it *uses two thresholds for network reconfiguration* and *a centralized control*.

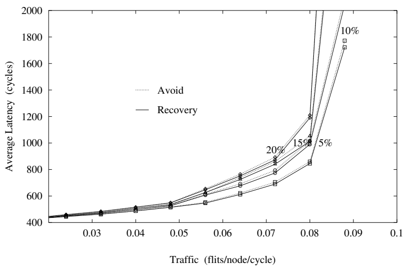
Finally, we have one last unresolved problem. This main problem is the deadlock problem. Because we have a reconfigurable network, we cannot guaranty the absence of these situations even if a deadlock-free routing algorithm for static networks is being used. In the next section we discuss it and we show the solution adopted in our algorithm.

4 Deadlock Problem in Reconfigurable Networks

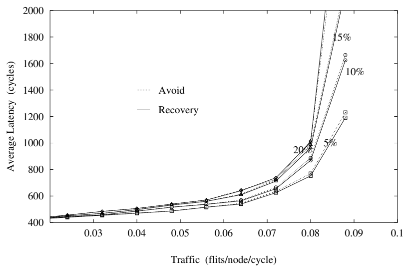
Deadlock problem can occur in many different situations and its main cause is to be found in the management of the available resources in each of them. Deadlock problem can appear in reconfigurable networks too. Deadlock occurs because at a given moment some messages are routed in such a way that the used strategy of channels assignation (increasing order of dimension) is broken. These situations appear as a consequence of the changes of position of the nodes in the network.

To solve this problem, we have evaluated two of the possible strategies [4]: Deadlock avoidance and deadlock detection (recovery). To avoid deadlock in reconfigurable networks, we must detect what situations can produce it. For k-ary n-cube networks with deterministic routing and wormhole flow control technique, deadlock detection process consists of checking the following conditions [2]: (1) The channel required by a message after it is routed in an intermediate node belongs to the same dimension as the channel from which the message has arrived at that node. The use of such channel implies: (a) to invert the direction of the message, or (b) to carry the message at a node which it had already gone through, and we would obtain a complete cycle. (2) The channel required by a message after it is routed in an intermediate node belongs to a minor (greater) dimension than the dimension from which the message is arriving at that node, and the routing algorithm establishes an ascending (descending) order for the channels allocation.

We have also considered deadlock detection because deadlock situations will be very few and the result of a deadlock can be tolerated. The solution consists of checking periodically the state of the messages and their associated channels to verify the appearance of any deadlock. In order to avoid a deadlock, in the first case above mentioned, and to break a situation which has already taken place, a simple solution has been adopted. It consists of storaging whole messages involved in a potential deadlock situation in the local memory of the node where their headers are blocked. So, the channels that these messages are using are allowed to be free, and the other messages can keep on moving. The messages that have been storaged in the local memory will continue their paths when the channels they need are free. This solution can introduce an important increment



(a) One hot-spot



(b) Two hot-spots

Fig. 4. Average message latency versus traffic for 8×8 torus and crossbar switch

on latency. However, we have chosen this strategy because our reconfiguration algorithm produces a low number of changes in the network and all the changes do not necessarily drive to deadlock states in the network. Taking into account different simulations, we have encountered that approximately in a 40% of the changes a blocked situation will occur. So, as the number of times that it occurs is low, the increment of latency is not significant.

We have carried out diverse tests to determine which behaves better of the two techniques. In figure 4 the results corresponding to a hot-spots workload are shown. It has been considered a situation in which the percentage of messages that go directed to hot-spots has been varied between 5% and 20%. It can be seen that the recovery technique behaves lightly better. This fact confirms that the deadlock situations are not excessive and that the cost associated to the treatment of all the deadlock potential situations that is carried out applying avoid technique is such that its use does not make advisable.

5 Performance Evaluation

In this section we are going to evaluate our network reconfiguration algorithm. The evaluation methodology used is based on the one proposed in [3]. The results have been obtained with Pepe environment [7], a programming and evaluating tool for multicomputers and multiprocessors. Pepe has two main phases: the first phase is more language-oriented, and it allows us to code, simulate and optimize a parallel program. The second phase has several tools for mapping and evaluating the architecture. The main feature included in Pepe is that it permits the network to be dynamically reconfigured. With the network simulator, we can evaluate the performance of the interconnection network for parallel applications and synthetic workloads. The simulator allows us to vary the parameters of the network and study how to improve its behaviour in real cases and predetermined situations.

We have used two switch types to simulate the reconfigurable network: A crossbar and a Omega multistage [9]. The former allows us to obtain very tiny size systems, up to 64 nodes. By means of the latter we will get larger size systems, being therefore the reconfiguration capacity much smaller than in the above case. We have evaluated the behavior of our reconfigurable system comparing its results with those obtained for a static network system. Having such a network a 2D torus topology and being both (static and reconfigurable networks) under a same workload. Therefore, we must establish the adequate connections in the crossbar (o multistage) to obtain the 2D torus topology.

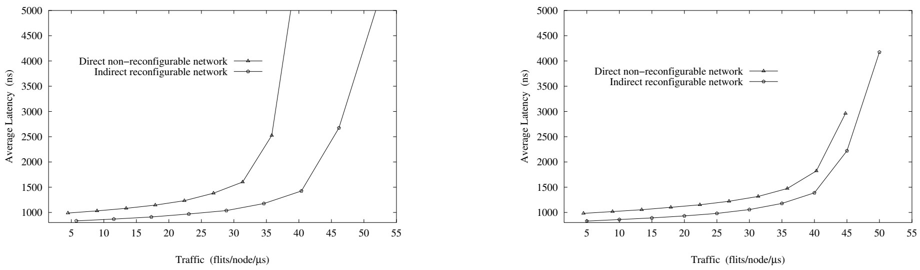
5.1 Parameters and Results of Simulations

We have considered hot-spot traffic model [10]. The situation that is going to be simulated is the following: Let us consider a network with a uniform distribution of message destinations. In this message pattern, message destinations are randomly chosen among all the nodes with the same probability. At a given moment, and with the network in steady state, the communication pattern changes, and a small number of hot-spots appear in the network. This situation repeats with a variable frequency and, in general, the hot-spots are different.

We have evaluated the performance of the reconfiguration algorithm on 8-ary 2-cube and 16-ary 2-cube networks. The deterministic algorithm proposed in [2] for the k-ary n-cube has been used. It has been modified so that it uses bidirectional channels with two virtual channels per physical channel. The proportion of messages at the hot-spot has been varied between 5% and 20%. For each simulation run, we have considered that message generation rate is constant and the same for all the nodes. Each simulation was run until the network reached steady state, that is, until a further increase in simulated network cycles did not change the measured results appreciably. Once the network has reached a steady state, the flit generation rate is equal to the flit reception rate (traffic). The number of hot-spots taken has been 1 and 2, and they have been obtained randomly. Finally, 16-flit and 100-flit messages have been considered.

For the characteristics of the used interconnection devices, we have applied the channel pipelining technique on the reconfigurable system. In figure 5 the results are observed of applying this technique. In both cases (network with 1 or 2 hot-spots) the reduction of average message latency is quite significative. The results show here correspond with a 8-ary 2-cube network, 128-flit messages and 5 % non-uniform component to hot-spots. We have born in mind a crossbar switch to connect the nodes in the system.

Another important result is that the number of changes necessary to reach these performance is very small. These results confirm that this technique as a very interesting one for systems with a distributed memory. Many more data and results on the incorporation of reconfigurable networks to massively parallel systems can be found in [11].



(a) One hot-spot (b) Two hot-spots

Fig. 5. Average message latency versus traffic for 8×8 torus and crossbar switch

6 Conclusions and Future Work

In this paper, we have presented a reconfigurable network model. Reconfigurable networks are an alternative to reduce the negative effect that congestion produces on the performance of the network as alternative to adaptive routing or random routing. Network reconfiguration can be done in different ways. Our researches are focused on dynamic reconfiguration. We have featured the reconfiguration technique supported by this model. We are analyzing the capabilities of several types of configuration, using a unique crossbar or by means of a multistage network.

We also prove how to easily solve the deadlock situations. For doing so we have taken into account that the number of these situations is very small.

Finally, the performance of our reconfiguration algorithm has been analyzed under hot-spot workload. We have chosen this problem because the congestion due to the presence of hot spots is an important and difficult problem that occurs in parallel machines ([10, 1]). We have presented here the obtained results for an average size network of 64 nodes and a crossbar reconfigurable network. Simulation was used to evaluated the proposed technique under certain assumptions about the execution environment and the network structure.

For future work, we would like to extend our study over larger sized networks, such as 256, 512 or 1024 nodes. To achieve this, we would use a multistage instead of a crossbar. Besides, we would like to study the behavior over other topologies such as mesh and 3-D torus, and under other models of load.

Finally, we would like to enlarge this work to the direct networks. Another interesting approach we are investigating is to combine reconfigurable networks with random routing and adaptive routing. We are very interested in getting to know the real capabilities of the reconfigurable interconnection networks in any kind of context.

References

1. Dandamudi, S.P., Eager, D.L.: Hot-spot contention in binary hypercube networks. IEEE Transactions on Computers **41**, No. 2 (1992) 239–244
2. Dally, W.J., Seitz, C.L.: Deadlock-free message routing in multiprocessor interconnection networks. IEEE Transactions on Computers **C-36** No. 5 (1987) 547–553
3. Duato, J.: A new theory of deadlock-free adaptive routing in wormhole networks. IEEE Transactions on Parallel and Distributed Systems **4** No. 12 (1993) 1320–1331
4. Duato, J., Yalamanchili, S., Ni, L.: Interconnection networks: An engineering approach. IEEE Computer Society Press (1997)
5. Fraboul, Ch., Rousselot, J.Y., Siron, P.: Software tools for developing programs on a reconfigurable parallel architecture. In D. Grassilloud, J.C. Grossetie, editors, *Computing with Parallel Architectures: T. Node* Kluwer Academic Publishers (1991) 101–110
6. García, J.M., Duato, J.: Dynamic reconfiguration of multicomputer networks: Limitations and tradeoffs. In P. Milligan, A. Nuñez, editors, *Euromicro Workshop on Parallel and Distributed Proces.*, IEEE Computer Society Press (1993) 317–323
7. García, J.M., Sánchez, J.L., González, P.: Pepe: A trace-driven simulator to evaluate reconfigurable multicomputer architectures. Springer Lecture Notes in Computer Science **1184** (1996) 302–311
8. Kermani, P., Kleinrock, L.: Virtual cut-through: A new computer communication switching technique. Computer Networks **3** (1979) 267–286
9. Lawrie, D.H.: Access and alignment of data in an array processor. IEEE Transactions on Computers **C-24** No. 12 (1975) 1145–1155
10. Pfister, G.F., Norton, V.A.: Hot spot contention and combining in multistage interconnect networks. IEEE Transactions on Computers **C-34** No. 10 (1985) 943–948
11. Sánchez, J.L., Duato, J., García, J.M.: Using channel pipelining in reconfigurable interconnection networks. 6^{th} Euromicro Workshop on Parallel and Distributed Processing (1998) 120–126