ADAPTIVE ROUTING ON TRANSPUTER NETWORKS WITH TORUS 2-D TOPOLOGY

Antonio Flores and J. M. Garcia

Facultad de Informatica
Universidad de Murcia
Campus de Espinardo s/n
30080 Murcia (Spain)

Email: jmgarcia@dif.um.es

ABSTRACT.- Second generation multicomputers use wormhole routing, allowing a very low channel setup time and drastically reducing the dependency between network latency and internode distance. The network routing can be deterministic or adaptive. In this paper, we present the development and implementation of a fully adaptive routing algorithm. Our work is carried out in a multicomputer with the C104 routers and the T9000 transputers. We describe several implementation details and give some simulation results. These results show the performance improvement that can be achieved using fully adaptive routing. Our simulations have been carried out on a torus 2-D topology.

INDEX TERMS.- Adaptive routing, multicomputers, transputers, wormhole routing.

1. Introduction.

A lot of research effort has been dedicated during the last decade to improve the performance and cost of multicomputers. The interconnection network used in a multicomputer plays a major impact on such aspects. A wide variety of interconnection networks have been proposed, either direct or indirect networks. Direct networks incorporate the processing elements within the network itself, allowing for direct communication between processors. These networks are gaining in popularity and have been employed in many recent existing or proposal machines.

The most commonly used direct networks are variants of the *k-ary n-cube* [2]. The k-ary n-cube consists of $N=k^n$ nodes, arranged in n dimensions, with k nodes per dimension. Each node is connected via a direct link to its nearest neighbours in each of n dimensions. Links can be bi- or unidirectional. The generality and flexibility of the k-ary n-cube make it an excellent choice for interconnection network.

A key issue in these networks is *routing algorithm* and *flow control* [7]. Routing determines the path chosen by a message in the network. Flow control deals with the allocation of channel and buffer resources to a message as it travels along this path.

Most multicomputers use *deterministic routing*. With deterministic routing, the path followed by a packet is determined solely by its source and destination. This method is also referred to as oblivious routing. If any channel along this path is faulty the packet cannot be delivered. In *adaptive routing*, for a given source and destination, the path taken by a

particular packet depends on dynamic network conditions, such as the presence of faulty or congested channels. In this way, adaptive routing improves both the performance and fault tolerance of an interconnection network.

Flow control deals with the allocation of channels and buffers to a packet as it travels along a path through the network. Early direct networks used *store-and-forward* switching. In this approach, when a packet reaches an intermediate node, the entire packet is stored in a packet buffer. A newer approach is known as wormhole routing. In this technique, each message is split into a sequence of data units, called flow control units, or flits. The flit at the head of a message determines the route. As the header flit is routed, the remaining flits follow it in a pipeline fashion. If the message header is blocked at some intermediate node, then all the links that hold packets are blocked until the header can move forward. Wormhole routing avoids using storage bandwidth in the nodes through which messages are routed. Also, it makes the message latency largely insensitive to the distance in the network.

Routing must be performed in a manner that is deadlock-free. Deadlock appears when a set of packets became blocked forever in the network. This situation occurs whenever there is a cyclic dependency for resources: buffers or channels. Networks that use deterministic routing usually avoid deadlock by ordering channels so that messages travel along paths of strictly increasing (or decreasing) channel numbers [1]. This ordering eliminates cycles in the channel dependency and thus prevents deadlock. However, there is a unique path from source to destination nodes.

A different approach must be considered for adaptive routing, in which messages usually have several options in each node. In this case, it is not necessary to eliminate all the cyclic dependencies. For store-and-forward networks, it is sufficient with the existence of a connected channel subset C_1 whose channels are not involved in cyclic dependencies. However, wormhole routing can reserve several channels simultaneously, being possible for a message to reserve a channel belonging to C_1 and then another no belonging to C_1, which may be involved in a cyclic dependency. Then, it is very important the development of deadlock-free algorithms. In [3] we can see two deadlock-free adaptive routing algorithms using virtual channels. Recently, Duato [5] developed a necessary and sufficient condition for deadlock-free adaptive routing, known as Duato's protocol [6]. Our paper is focused on this recent theory.

In this paper we implement the fully adaptive routing algorithm for k-ary n-cubes based in the Duato's theory on C104 routers and T9000 transputers. The T9000 is the second-generation transputer, which has some bandwidth and performance improvements over its predecessors. The C104 routing switch adds a further lever of performance, allowing any T9000 to communicate with any other T9000 in the network without having to through-route messages through intermediate processors. Usually, the transputer networks use a deterministic routing. In this paper, we show the implementation of an adaptive algorithm and the benefits can be obtained. We also perform a comparative study between this algorithm and the deterministic routing algorithm [2]. Section 2 deals with the Duato's adaptive algorithms introducing the theory's foundations and the new kinds of channel dependency. Section 3 proposes an implementation of the former algorithm on C104 Routers and T9000 Transputers. Section 4 shows the results achieve through simulations. Finally, in section 5, some conclusions are drawn.

2. Duato's Fully Adaptive Algorithm for the k-ary n-cube topology.

2.1. Foundations.

In this section, we are going to present the Duato's fully adaptive routing algorithm. Firstly, we give the necessary theoretical foundations for this new theory. This presentation is described in an informal way.

The objective of this theory is to find under what assumptions an adaptive routing can be deadlock-free. Duato introduces a new kind of dependency among channels to prove the necessary and sufficient condition for an adaptive algorithm to be deadlock-free. These new dependencies are called *indirect dependency, direct cross dependency,* and *indirect cross dependency* [4], [5].

Indirect dependency appears in the following situation. Let C_1 be a connected channel subset that is not involved in cyclic dependencies. When a message uses a channel belonging to C_1, then another channel no belonging to C_1 and then it tries to use a channel belonging to C_1, there is an *indirect dependency* between both channels belonging to C_1. If cyclic dependencies are not allowed between channels belonging to C_1, even when indirect dependencies are considered, then we guarantee that the escape paths will not block forever. Therefore, the adaptive routing is deadlock-free. But this condition is only sufficient.

We can go one step further by conditionally assigning each channel to the escape paths, depending on the destination of the message. Then, the maximum flexibility is achieved in the definition of the escape paths, but new problems appear. Effectively, a message M can reserve a channel c_i which does not belong to any escape path for the destination of M, and then another channel belonging to the escape path for the destination of M. If c_i belongs to the escape paths for other destinations, other messages may need it to reach their destination. However, that channel may be occupied forever by M, which in turn may be blocked by another message requesting c_i. It must be noticed that the problem arises when there are cyclic dependencies among channels belonging to the escape paths and channels which do not belong to the escape path for the message occupying them, but belong to the escape paths for others destinations. Again, the solution consists in defining a new kind of dependency, the *direct cross dependency*. Moreover, if this dependency is combined with the indirect dependency, the *indirect cross dependency* is obtained. By forbidding the cycles in the *extended channel dependency graph* of $C_1 \subseteq C$, in which a pair of channels are connected if there is either a direct, indirect, direct cross or indirect cross dependency, the deadlock-free condition is achieved. Moreover, this condition is, not only sufficient, but necessary.

2.2. Adaptive Algorithms for k-ary n-cube.

From the previous design methodology, several adaptive routing algorithms can be derived for any topology. For k-ary n-cube, Duato proposes two algorithms, partially adaptive and fully adaptive. Firstly, we explain the partially adaptive algorithm. In simpler case, we have n=1, that is, an unidirectional ring. For simplicity, we consider the ring with four nodes

(k=4) denoted n_i, i={0,1,2,3} and two channels connecting each pair of adjacent nodes, except nodes n_3 and n_0 which are linked by a single channel. In figure 1 we can see the network.

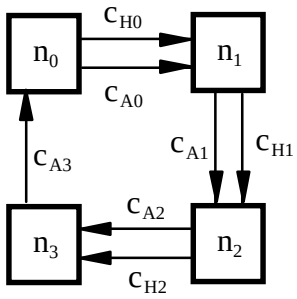

Figure 1. Unidirectional 4-ary 1-cube.

Let c_{Ai}, i={0,1,2,3} and c_{Hi}, i={0,1,2} be the outgoing channels from node n_i. The adaptive routing algorithm can be stated as follows: If the current node n_i is equal to the destination node n_j, store the message. Otherwise, use either c_{Ai}, $\forall j \neq i$ or c_{Hi}, $\forall j > i$. It is easy to show that there is not any deadlocked configuration by trying to build one. Also, it can be proved by using the extended channel dependency graph [5].

The next step is extending our adaptive algorithm for bidirectional rings. In this case, bidirectional channels must be split into two unidirectional channels. Then, our former algorithm is used, always forwarding messages through a minimal path. As messages follow a minimal path, once a message starts crossing channels in one direction, it cannot turn and continue in the opposite direction. Therefore, the use of bidirectional channels does not introduce any additional channel dependency and the routing algorithm is also deadlock-free.

Now, we are going to present the adaptive routing algorithm for k-ary n-cubes. It uses the algorithm for bidirectional rings inside each dimension. Channels can only be used crossing dimensions in ascending order. The result is a partially adaptive minimal routing algorithm which only requires two virtual channels per physical channel. As dimensions are crossed in ascending order, adding dimensions does not introduce any additional cyclic dependency and the routing algorithm is still deadlock-free.

Finally, to design a fully adaptive routing algorithm we start from the partially adaptive routing algorithm. Then, we add a third virtual channel to each physical channel. This third virtual channel is used for fully adaptive routing, crossing dimensions in any order following a minimal path. Obviously, the first and second virtual channels are used exactly in the same way as in the partially adaptive algorithm. The result is a fully adaptive minimal routing algorithm with three virtual channels per physical channel. It is easy to prove deadlock-freedom by including the additional virtual channel in $C\text{-}C_1$. In other words, the routing subfunction R_1 cannot use the fully adaptive virtual channels for any destination.

3. Implementation of the Fully Adaptive Algorithm on the C104.

In this section, we deal with the implementation on C104 routers and T9000 transputers of the algorithm proposal in the previous section. We focus on the k-ary 2-cube topology or torus 2-D. This network is a simple and popular topology, with many interesting properties as symmetric, rectangular structure, wirability and scalability. Many existing multicomputers are using such low-dimensional networks as Intel Paragon, Stanford Dash or MIT J-Machine. Previously, it is necessary to explain some details of the C104 router in order to achieve a fully understanding of the implementation made and the aroused problems.

3.1. The C104 router.

The ST C104 is a packet routing chip that contains a 32-way crossbar switch, where all the 32 inputs can be routed simultaneously to the 32 outputs [8], [9]. Routing delays are minimized by the use of wormhole routing, in which a packet can start to be output from a switch whilst it is still being input.

Wormhole routing requires an efficient routing strategy to decide which link a packet should be output from. The C104 uses a routing scheme called *interval labelling*, whereby each output link of a C104 is assigned a range, or interval, of labels. This interval contains the number of all the terminal nodes (i.e. microprocessors, gateway to another network, peripheral chip, etc.) which are accessible via that link. As a packet arrives at a C104 router the selection of the outgoing link is carried out comparing the header label with the set of intervals. As it is showed in figure 2, the intervals are contiguous and non-overlapping and assigned so that each header label can only belong to one of the intervals.

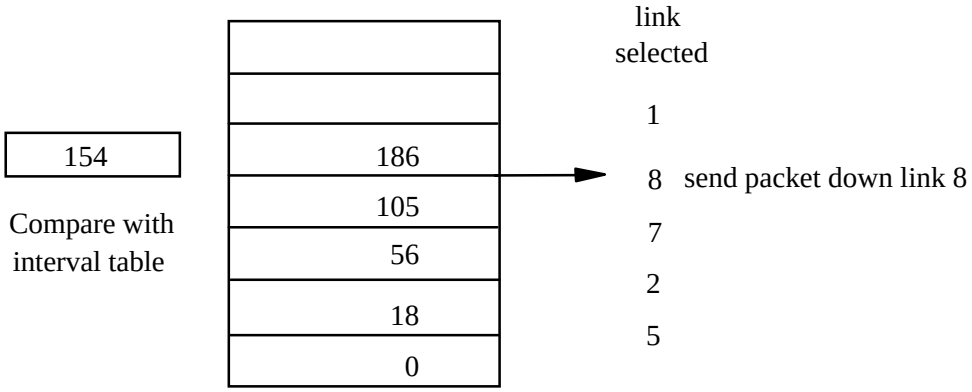

Figure 2. Interval labelling.

Interval routing is implemented on C104 by interval selector units. There is one interval selector unit per input link, so the C104 router permits to modify the output selected link in function of the input link that the incoming packet used. However, the adaptive routing algorithm used in Duato's theory does not depend on the input link used, therefore, all the C104's input links will have the same interval table.

Although the C104 Router does not restrict the size of the packets, the T9000 Transputer does. A packet can contain up to 32 data bytes. If a message is longer than 32 bytes then it is split up into a number of packets all, except the last, terminated by an *end of packet* token. The last packet of the message, which may contain less than a full 32 bytes, is terminated by an *end of message*. Shorter messages can be sent in a single packet, containing 0 to 32 bytes of data, terminated by the *end of message* token.

The C104 router allows deterministic or adaptive routing. The adaptive routing is named *grouped adaptive routing*. It is implement in the following manner. A set of consecutive numbered links can be configured to be grouped, so that a packet routed to any link in that set would be sent down any free link of the set. This feature will be used to implement the fully adaptive routing algorithm seen in the former section.

3.2. Implementation of the Duato's Fully Adaptive Routing Algorithm.

In this subsection, we will consider the following configuration for a torus 2-D with transputers:

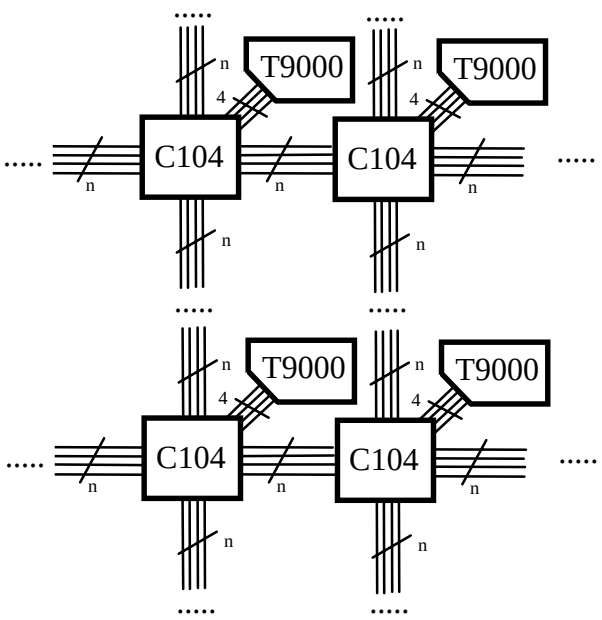

Figure 3. A k-ary 2-cube with Transputers.

where n=7 will be the maximum number of physical links in each direction.

Note the C104 router uses an interval labelling that has not to be overlapping. That fact hinders the implementation of some routing algorithms in which exists overlapping between the links that can be used by an arriving packet, in function of the destination node.

As earlier, we show our implementation of the adaptive routing algorithm for a bidirectional ring with k nodes numbered from 0 to k-1.

In this case, each node n_i, i=0,...k-1, has three virtual channels on the right side and three virtual channel on the left side, as it is showed in figure 4.

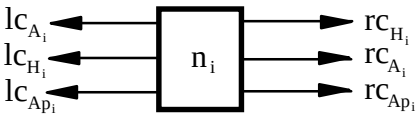

Figure 4. Virtual channels per node.

We have found three cases, depending on the position of the node, as we can see in figure 5.

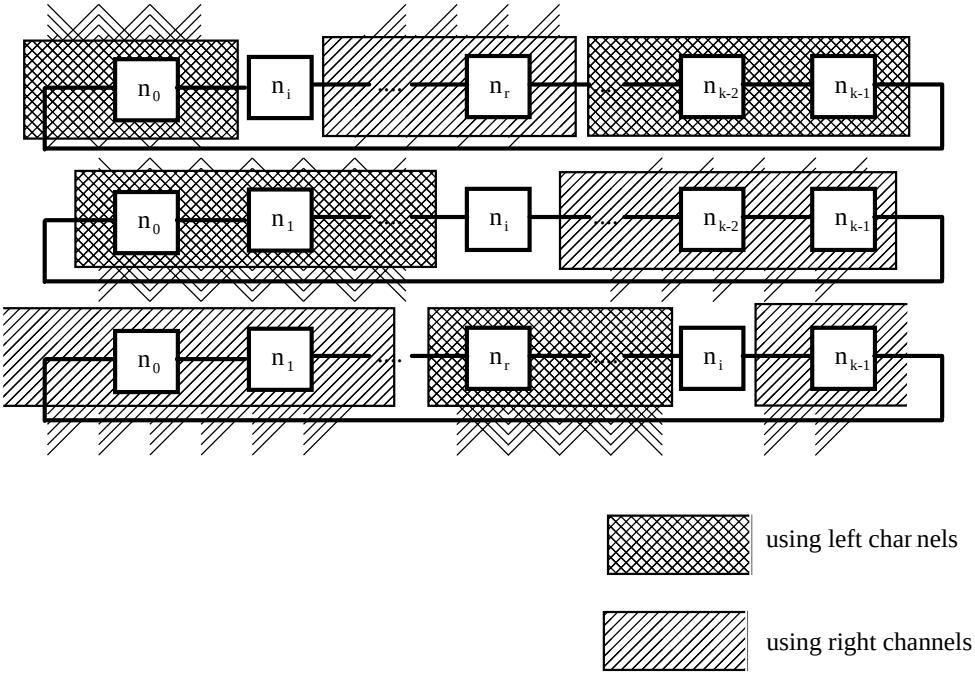

Figure 5. Routing cases depending on the position of the current node.

The implementation of the fully adaptive algorithm depends on each previous case. For each one, we obtain the following interval labelling:

$$\bullet\ i < \left\lfloor \frac{k-1}{2} \right\rfloor \Rightarrow \begin{cases} d \in [0,i) \Rightarrow \{lc_{A_i}, lc_{H_i}, lc_{Ap_i}\} \\ d \in \{i\} \Rightarrow \text{internal channels} \\ d \in \left[i+1, i+\left\lfloor \frac{k}{2} \right\rfloor + 1 \right) \Rightarrow \{rc_{A_i}, rc_{H_i}, rc_{Ap_i}\} \\ d \in \left[i+\left\lfloor \frac{k}{2} \right\rfloor + 1, k \right) \Rightarrow \begin{cases} i = 0 \Rightarrow \{lc_{A_i}, lc_{H_i}, lc_{Ap_i}\} \\ i \neq 0 \Rightarrow \{lc_{A_i}, lc_{Ap_i}\} \end{cases} \end{cases}$$

$$\bullet\ i = \left\lfloor \frac{k-1}{2} \right\rfloor \Rightarrow \begin{cases} d \in [0,i) \Rightarrow \{lc_{A_i}, lc_{H_i}, lc_{Ap_i}\} \\ d \in \{i\} \Rightarrow \text{internal channels} \\ d \in [i+1,k) \Rightarrow \{rc_{A_i}, rc_{H_i}, rc_{Ap_i}\} \end{cases}$$

$$\bullet\ i > \left\lfloor \frac{k-1}{2} \right\rfloor \Rightarrow \begin{cases} d \in \left[0, i - \left\lfloor \frac{k}{2} \right\rfloor\right) \Rightarrow \begin{cases} i = k-1 \Rightarrow \{rc_{A_i}, rc_{H_i}, rc_{Ap_i}\} \\ i \neq k-1 \Rightarrow \{rc_{A_i}, rc_{Ap_i}\} \end{cases} \\ d \in \left[i - \left\lfloor \frac{k}{2} \right\rfloor, i\right) \Rightarrow \{lc_{A_i}, lc_{H_i}, lc_{Ap_i}\} \\ d \in \{i\} \Rightarrow \text{internal channels} \\ d \in [i+1, k) \Rightarrow \{rc_{A_i}, rc_{H_i}, rc_{Ap_i}\} \end{cases}$$

where d is the destination node and i is the current node. Observe that we not use the H channel in some cases. As we stated, the Duato's theory not allows its use in the routing for some destinations to avoid cycles in the dependency graph.

As we can see, this representation cannot be implement on a C104 router yet, because there are overlapping between the groups of channels used by the different packets. These limitations can be save adding more channels in order to force a separation between these groups and matching each virtual channel defined above with a C014's different physical channel. We must take into account that each C104 router has 32 physical links. In this point, our implementation only takes 6 channels (three per side) for communication among routers and 4 channels for communication with transputer. Therefore, it is possible to add more physical channels to the previous partition. In this case we need four channels more (two per side) to avoid the interval overlapping. Next, in figure 6, we show the final configuration for one of the previous cases.

$i < \left\lfloor \frac{k-1}{2} \right\rfloor$ case

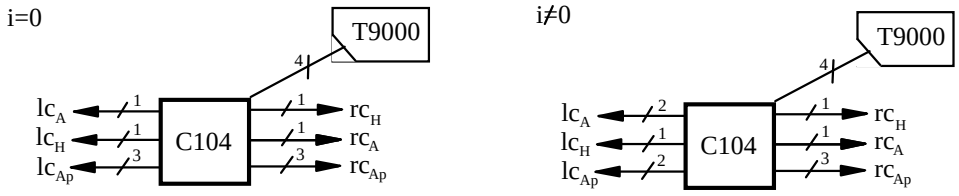

Figure 6. Final configuration of a k-ary 1-cube using adaptive routing.

The other two cases can be obtained easily by changing left and right channels. Remain physical channels of C014 router can be used to increase fault tolerance and performance of the network by duplicating the existing ones.

Now, we present our implementation of the fully adaptive routing algorithm on a torus 2-D. Firstly, we consider that each router is identified by an n-digit radix k number, in which the ith digit of the number represents the router's position in the ith dimension.we can describe the routing algorithm using interval labelling as before.

In this case, each node n_i, i=(0,0), ..., (k-1,k-1), has the virtual channels shows in figure 7.

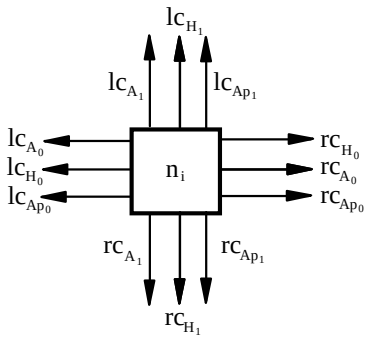

Figure 7. Virtual channels per node.

As earlier, we must consider the node relative position with respect the network centre. There are five possible locations: central node, up-right node, up-left node, down-right node and down-left node. Next, we show the implementation of two of them (The others can be obtained easily by exchanging right-left and/or up-down channels, depending of the node relative position).

$$\bullet\ (i, j) = \left(\left\lfloor \frac{k-1}{2} \right\rfloor, \left\lfloor \frac{k-1}{2} \right\rfloor \right)$$

$$\circ\ \forall r \in \{0,\ldots, i-1\}, \begin{cases} d \in [(r,0),(r, j)) \Rightarrow \{lc_{A_0}, lc_{H_0}, lc_{Ap_0}, lc_{Ap_1}\} \\ d \in \{(r, j)\} \Rightarrow \{lc_{A_1}, lc_{H_1}, lc_{Ap_1}\} \\ d \in [(r, j+1),(r+1,0)) \Rightarrow \{rc_{A_0}, rc_{H_0}, rc_{Ap_0}, lc_{Ap_1}\} \end{cases}$$

$$\circ\ r = i, \begin{cases} d \in [(r,0),(i, j)) \Rightarrow \{lc_{A_0}, lc_{H_0}, lc_{Ap_0}\} \\ d \in \{(i, j)\} \Rightarrow \text{internal channels} \\ d \in [(i, j+1),(i+1,0)) \Rightarrow \{rc_{A_0}, rc_{H_0}, rc_{Ap_0}\} \end{cases}$$

$$\circ\ \forall r \in \{i+1,\ldots, k-1\}, \begin{cases} d \in [(r,0),(r, j)) \Rightarrow \{lc_{A_0}, lc_{H_0}, lc_{Ap_0}, rc_{Ap_1}\} \\ d \in \{(r, j)\} \Rightarrow \{rc_{A_1}, rc_{H_1}, rc_{Ap_1}\} \\ d \in [(r, j+1),(r+1,0)) \Rightarrow \{rc_{A_0}, rc_{H_0}, rc_{Ap_0}, rc_{Ap_1}\} \end{cases}$$

$$\bullet\ (i, j) \left| \ i < \left\lfloor \frac{k-1}{2} \right\rfloor, j < \left\lfloor \frac{k-1}{2} \right\rfloor \right.$$

$$\circ\ \forall r \in \{0,\ldots, i-1\}, \begin{cases} d \in [(r,0),(r, j)) \Rightarrow \{lc_{A_0}, lc_{H_0}, lc_{Ap_0}, lc_{Ap_1}\} \\ d \in \{(r, j)\} \Rightarrow \{lc_{A_1}, lc_{H_1}, lc_{Ap_1}\} \\ d \in \left[(r, j+1),(r, j+\left\lfloor \frac{k}{2} \right\rfloor + 1)\right) \Rightarrow \{rc_{A_0}, rc_{H_0}, rc_{Ap_0}, lc_{Ap_1}\} \\ d \in \left[(r, j+\left\lfloor \frac{k}{2} \right\rfloor + 1),(r+1,0)\right) \Rightarrow \{lc_{A_0}, lc_{Ap_0}, lc_{Ap_1}\} \end{cases}$$

$$\circ\ r = i, \begin{cases} d \in [(r,0),(i, j)) \Rightarrow \{lc_{A_0}, lc_{H_0}, lc_{Ap_0}\} \\ d \in \{(i, j)\} \Rightarrow \text{internal channels} \\ d \in \left[(i, j+1),(i, j+\left\lfloor \frac{k}{2} \right\rfloor + 1)\right) \Rightarrow \{rc_{A_0}, rc_{H_0}, rc_{Ap_0}\} \\ d \in \left[(i, j+\left\lfloor \frac{k}{2} \right\rfloor + 1),(i+1,0)\right) \Rightarrow \begin{cases} i = 0 \Rightarrow \{lr_{A_0}, lc_{H_0}, lc_{Ap_0}\} \\ i \neq 0 \Rightarrow \{lc_{A_0}, lc_{Ap_0}\} \end{cases} \end{cases}$$

$$\circ\ \forall r \in \left\{ i+1,\ldots, i+\left\lfloor \frac{k}{2} \right\rfloor \right\}, \begin{cases} d \in [(r,0),(r,j)) \Rightarrow \{lc_{A_0}, lc_{H_0}, lc_{Ap_0}, rc_{Ap_1}\} \\ d \in \{(r,j)\} \Rightarrow \{rc_{A_1}, rc_{H_1}, rc_{Ap_1}\} \\ d \in \left[(r,j+1),(r,j+\left\lfloor \frac{k}{2} \right\rfloor +1)\right) \Rightarrow \{rc_{A_0}, rc_{H_0}, rc_{Ap_0}, rc_{Ap_1}\} \\ d \in \left[(r,j+\left\lfloor \frac{k}{2} \right\rfloor +1),(r+1,0)\right) \Rightarrow \begin{cases} i=0 \Rightarrow \{lc_{A_0}, lc_{H_0}, lc_{Ap_0}, rc_{Ap_1}\} \\ i \neq 0 \Rightarrow \{lc_{A_0}, lc_{Ap_0}, rc_{Ap_1}\} \end{cases} \end{cases}$$

$$\circ\ \forall r \in \left\{ i+\left\lfloor \frac{k}{2} \right\rfloor +1,\ldots, k-1 \right\}, \begin{cases} d \in [(r,0),(r,j)) \Rightarrow \{lc_{A_0}, lc_{H_0}, lc_{Ap_0}, lc_{Ap_1}\} \\ d \in \{(r,j)\} \Rightarrow \begin{cases} j=0 \Rightarrow \{lc_{A_1}, lc_{H_1}, lc_{Ap_1}\} \\ j \neq 0 \Rightarrow \{lc_{A_1}, lc_{Ap_1}\} \end{cases} \\ d \in \left[(r,j+1),(r,j+\left\lfloor \frac{k}{2} \right\rfloor +1)\right) \Rightarrow \{rc_{A_0}, rc_{H_0}, rc_{Ap_0}, lc_{Ap_1}\} \\ d \in \left[(r,j+\left\lfloor \frac{k}{2} \right\rfloor +1),(r+1,0)\right) \Rightarrow \begin{cases} i=0 \Rightarrow \{lc_{A_0}, lc_{H_0}, lc_{Ap_0}, lc_{Ap_1}\} \\ i \neq 0 \Rightarrow \{lc_{A_0}, lc_{Ap_0}, lc_{Ap_1}\} \end{cases} \end{cases}$$

As in bidirectional ring, these groups have to be modified in order to obtain disjoin sets of links in each option. Again, we add some channels to force the separation among groups. For example, if we consider the router which 2-digit radix k number is:

$$(i,j) = \left(\left\lfloor \frac{k-1}{2} \right\rfloor, \left\lfloor \frac{k-1}{2} \right\rfloor \right)$$

we obtain the next groups of channels, each one of them will have associated a C104's physical link:

$$\{lc_{A_{0,1}}, lc_{H_{0,1}}, lc_{Ap_{0,1}}, lc_{Ap_{1,1}}\}$$
$$\{lc_{A_{1,1}}, lc_{H_{1,1}}, lc_{Ap_{1,2}}\}$$
$$\{rc_{A_{0,1}}, rc_{H_{0,1}}, rc_{Ap_{0,1}}, lc_{Ap_{1,3}}\}$$
$$\{lc_{A_{0,2}}, lc_{H_{0,2}}, lc_{Ap_{0,2}}\}$$
$$\{rc_{A_{0,2}}, rc_{H_{0,2}}, rc_{Ap_{0,2}}\}$$
$$\{lc_{A_{0,3}}, lc_{H_{0,3}}, lc_{Ap_{0,3}}, rc_{Ap_{1,1}}\}$$
$$\{rc_{A_{1,1}}, rc_{H_{1,1}}, rc_{Ap_{1,2}}\}$$
$$\{rc_{A_{0,3}}, rc_{H_{0,3}}, rc_{Ap_{0,3}}, rc_{Ap_{1,3}}\}$$

This leads to the following configuration on the C104 router that use all 32 available channels:

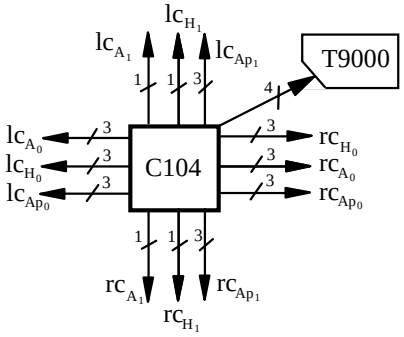

Figure 8. Configuration of the C104 router.

As we can see, this solution causes an asymmetry allowing more bandwidth in the first dimension. Moreover, there are some nodes in which the number of links that we will need to obtain disjoint sets exceed the maximum number of available links.

In this case, the best solution is the elimination of some of the allowing links. With this, we overcome both problems:

- Asymmetry in the topology.
- Insufficient number of links.

Furthermore, it is possible to eliminate these links without loss of adaptability by removing first one of the lc_A or lc_H channels (if both can be used). For example, considering the following modified routing algorithm for the central node

$$\circ\ \forall r \in \{0,\ldots,i-1\},\begin{cases} d \in [(r,0),(r,j)) \Rightarrow \{lc_{A_0}, lc_{Ap_0}, lc_{Ap_{1,1}}, lc_{Ap_{1,2}}\} \\ d \in \{(r,j)\} \Rightarrow \{lc_{A_1}, lc_{H_1}, lc_{Ap_1}\} \\ d \in [(r,j+1),(r+1,0)) \Rightarrow \{rc_{A_0}, rc_{Ap_0}, lc_{Ap_{1,1}}, lc_{Ap_{1,2}}\} \end{cases}$$

$$\circ\ r = i,\begin{cases} d \in [(r,0),(i,j)) \Rightarrow \{lc_{A_0}, lc_{H_0}, lc_{Ap_0}\} \\ d \in \{(i,j)\} \Rightarrow \text{internal channels} \\ d \in [(i,j+1),(i+1,0)) \Rightarrow \{rc_{A_0}, rc_{H_0}, rc_{Ap_0}\} \end{cases}$$

$$\circ\ \forall r \in \{i+1,\ldots,k-1\},\begin{cases} d \in [(r,0),(r,j)) \Rightarrow \{lc_{H_0}, lc_{Ap_0}, rc_{Ap_{1,1}}, rc_{Ap_{1,2}}\} \\ d \in \{(r,j)\} \Rightarrow \{rc_{A_1}, rc_{H_1}, rc_{Ap_1}\} \\ d \in [(r,j+1),(r+1,0)) \Rightarrow \{rc_{H_0}, rc_{Ap_0}, rc_{Ap_{1,1}}, rc_{Ap_{1,2}}\} \end{cases}$$

where four deterministic channels have been removed in order to add more adaptive channels, the configuration on the C104 router will be now:

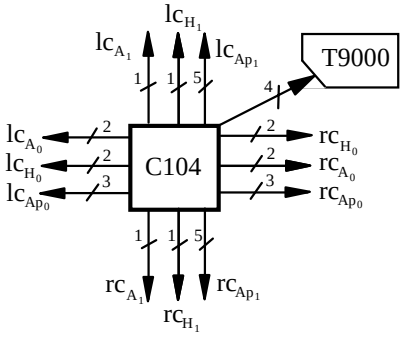

Figure 9. Configuration of the C104 router.

The same technique can be used for the other cases, obtaining a configuration that solves the asymmetry in the topology and needs an amount of links no greater than 32.

4. Simulations.

In order to evaluated the behaviour of our adaptive algorithm we have carried out some simulations. The most important performance measures are delay (additional latency required to deliver a message with respect to an idle network) and throughput (maximum amount of information delivered per time unit). Delay is measured in clock cycles. Throughput is measured in flits per node per cycle.

We have evaluated the performance on a torus 2-D with 8 nodes per dimension. Our simulator models the network at the flit level. In addition, the deterministic e-cube algorithm [2] for torus 2-D has been showed for comparison purposes.

The modelling parameters for a network make up C104 routers and T9000 transputer can be obtained taking into account the following concepts:

- The DS-Links have a transmission speed of 100 Mbits/s, but the overhead imposed by the various layers of protocol reduces this speed to 8.27 Mbytes/s for packets with 2 byte headers and 8.74 Mbytes/s for 1 byte headers. This data are given without considering latencies in the system that cause the link become idle for part of the time [8]. So a flit will need approximately 370 ns to cross a DS-Link.

- The C104 router need approximately 500ns to determine the output link on which the incoming packet will be router. A packet need 370 ns to cross the router [8].

- The proposal network has seven physical links in each direction, this can be simulated supposing a single physical link with a bandwidth seven times wider. So, a packet will need about 50 ns to be transmitted. It is also right to suppose that the routing and cross time will be decrease in absence of collisions between the output links selected by these physical links. Although it will be necessary a probabilistic study in order to determine the mean routing and cross time under this new assumption, we can suppose that it will be seven times faster without problems because these times will be used in both routing algorithms (deterministic and adaptive). So, it will not affect to the results.

- We only consider the time taken by the messages inside the network. So we do not consider the time for communications between C104 routers and T9000 transputers.

From these data, we have carried out several simulations with the following parameters. Each simulation has collected data from 50000 messages, discarding the messages corresponding to a transient period required to reach steady state (10000 messages). Message length is fixed to 256 flits. Messages are generated randomly following a uniform distribution. Message destination is also selected randomly. The clock unit is 10 μs.

Figure 10 shows, for a uniform distribution for destination of messages, the average message delay versus traffic in the network and the amount of time for which the message was blocked (average delay/average latency) versus traffic. These figures give an accurate information about the performance improvement in the network.

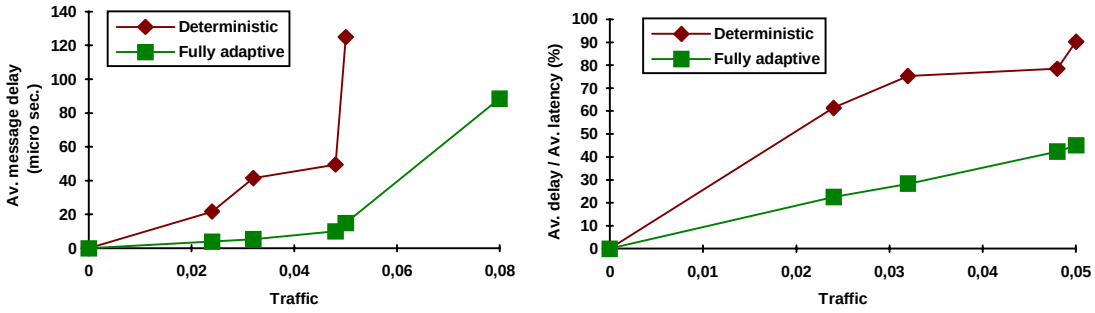

Figure 10. Comparative simulation results.

We see that adaptive algorithm increase performance over the deterministic one. Maximum traffic allowing before saturation (throughput) also increases, because adaptive routing permits a message to cross the dimensions in any order. Then, the routing function offers more choices, reducing channel contention and message delay accordingly. Also, the adaptive algorithm is able to use any minimal path, increasing channel utilization. As a consequence, throughput also increases.

Finally, the number of choices at each routing step increases with network size, almost nullifying the negative effect produced by higher traffic. Because of that, the adaptive algorithm scales very well with network size.

5. Conclusions.

We have carried out the implementation of fully adaptive routing proposal by Duato on a 2-D torus topology make up of C104 routers and T9000 transputers. This implementation improves drastically results achieved by deterministic routing such as channel contention and utilization, message delay, throughput, etc. It also scales well with network size overcoming one of the main problems of deterministic routing.

Finally, the utilization of low cost components as C104 routers and T9000 transputers makes possible a wider use of multicomputers based on these components.

6. References.

[1] W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks", IEEE Trans. Computers, Vol. C-36, No. 5, pp. 547-553, May 1987.

[2] W. J. Dally, "Performance analysis for k-ary n-cube interconnection networks," IEEE Trans. Comput., Vol. C-39, No. 6, pp. 775-785, June 1990.

[3] W. J. Dally and H. Aoki. "Deadlock-free Adaptive Routing in Multicomputers Networks Using Virtual Channels", IEEE Trans. Parallel and Distrib. Systems, Vol. 4, N0. 4, pp. 466-475, April 1993.

[4] J. Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks", IEEE Trans. Parallel Distributed Syst., Vol. 4, No. 12, pp. 1320-1331, Dec. 1993.

[5] J. Duato, "A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks". Proc. Int. Conf. on Parallel Processing, 1994.

[6] P. T. Gaughan and S. Yalamanchili, "Adaptive routing protocols for hypercube interconnection networks", IEEE Comput. Mag., Vol. 26, No. 5, pp. 12-23, May 1993.

[7] Lionel M. Ni and K. Mckinley, "A Survey of Wormhole Routing Techniques in Direct Networks", IEEE Comput., pp. 62-76, Feb. 1993.

[8] M. D. May, P. W. Thompson, P. H. Welch et al. "Networks, Routers & Transputers: Function, performance and application.". (IOS Press, Amsterdam, 1993).

[9] "ST C104". SGS-Thomson Microelectronics. June 1994 Datasheet.