A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks

José Duato

Abstract

Second generation multicomputers use wormhole routing, allowing a very low channel set-up time and drastically reducing the dependency between network latency and internode distance. Deadlock-free routing strategies have been developed, allowing the implementation of fast hardware routers that reduce the communication bottleneck. Also, adaptive routing algorithms with deadlock-avoidance or deadlock-recovery techniques have been proposed for some topologies, being very effective and outperforming static strategies.

This paper develops the theoretical background for the design of deadlock-free adaptive routing algorithms for wormhole networks. Some basic definitions and two theorems are proposed, developing conditions to verify that an adaptive algorithm is deadlock-free, even when there are cycles in the channel dependency graph. Also, two design methodologies are proposed. The first one supplies algorithms with a high degree of freedom, without increasing the number of physical channels. The second methodology is intended for the design of fault-tolerant algorithms. Some examples are given, showing the application of the methodologies. Finally, some simulations show the performance improvement that can be achieved by designing the routing algorithms with the new theory.

Index terms

Adaptive routing, deadlock avoidance, design methodologies, fault tolerance, graph theory, multicomputers, virtual channels, wormhole routing.

1 Introduction

Multicomputers [1] rely on an interconnection network between processors to support the message-passing mechanism. The network latency [1] can be defined as the time from when the head of a message enters the network at the source until the tail emerges at the destination. In first generation multicomputers, a store-and-forward mechanism has been used to route messages. Each time a message reaches a node, it is buffered in local memory, and the processor interrupted to execute the routing algorithm. Accordingly, the network latency is proportional to the distance between the source and the destination.

However, second generation multicomputers are most distinguished by message routing hardware that makes the topology of the message-passing network practically invisible to the programmer. The message routing hardware uses a routing mechanism known as wormhole routing [10]. As messages are typically at least a few words long, each message is serialized into a sequence of parallel data units, referred to as flow control units, or flits [9]. The flit at the head of a message governs the route. As the header flit advances along the specified route, the remaining flits follow it in a pipeline fashion. If the header encounters a channel already in use, it is blocked until the channel is freed; the flow control within the network blocks the trailing flits.

This form of routing and flow control has two important advantages over the store-and-forward routing used in first generation multicomputers. Firstly, it avoids using storage bandwidth in the nodes through which messages are

routed. Secondly, this routing technique makes the message latency largely insensitive to the distance in the message-passing network. Since the flits move through the network in a pipeline fashion, in the absence of channel contention, the network latency equals the sum of two terms:

— $T_p d$ is the time associated with forming the path through the network, where T_p is the delay of the individual routing nodes found on the path, and d is the number of nodes traversed.

— L/B is the time required for a message of length L to pass through a channel of bandwidth B.

In second generation multicomputers, the network latency is dominated by the second term for all but very short messages.

Another improvement in message performance results from selecting the optimal topology for the implementation on printed circuit boards or VLSI chips. As message latency is dominated by the term L/B, more wirable network topologies will increase the bandwidth B at the expense of increasing the network diameter. An analysis [5, 7] shows that, under the assumption of constant number of wires through the network bisection, a two dimensional network minimizes latency for typical message lengths for up to 1024 nodes. For larger sizes, a three dimensional network achieves better performance. Among these networks, meshes are preferred because they offer useful edge connectivity, which can be used for I/O controllers. Also, meshes partition into units that are still meshes, simplifying the design of routing algorithms that are independent of the network size, as well as the implementation of space-sharing techniques.

However, deadlocks may appear if the routing algorithms are not carefully designed. A deadlock in the interconnection network of a multicomputer occurs when no message can advance toward its destination because the queues of the

message system are full. The size of the queues strongly influences the probability of reaching a deadlocked configuration. First generation multicomputers buffer full messages or relatively large packets. By contrary, second generation machines buffer flits, being more deadlock-prone. So, the only practical way to avoid deadlock is to design deadlock-free routing algorithms.

Many deadlock-free routing algorithms have been developed for store-and-forward computer networks [13, 15, 23]. These algorithms are based on a structured buffer pool. However, with wormhole routing, buffer allocation cannot be restricted, because flits have no routing information. Once the header of a message has been accepted by a channel, the remaining flits must be accepted before the flits of any other message can be accepted. So, routing must be restricted to avoid deadlock.

Dally [10] has proposed a methodology to design static routing algorithms under general assumptions. He defines a channel dependency graph and establishes a total order among channels. Routing is restricted to visit channels in decreasing or increasing order to eliminate cycles in the channel dependency graph. This methodology has been applied to the design of routing chips for multicomputers [9] and multicomputer nodes with integrated communication support [2]. It has also been applied to systolic communication [21, 2].

The restriction of routing, although it avoids deadlock, can increase traffic jams, especially in heavily loaded networks with long messages. In order to avoid congested regions of the network, an adaptive routing algorithm can be used. Adaptive strategies have been shown to outperform static strategies in store-and-forward routing [3] and in packet-switched communications [20, 24]. In general, adaptive routing needs additional hardware support.

Several adaptive algorithms have been developed for wormhole routing. A deadlock-free adaptive algorithm for the hypercube is the Hyperswitch algo-

rithm [4], which is based on backtracking and hardware modification of message headers to avoid congestion and cycles. Another deadlock-free adaptive algorithm has been proposed for the MEGA [14]. This algorithm always routes messages, sending them away from their destination if necessary, like the Connection Machine [16]. If the message arrives at a node without free output channels, deadlocks are avoided by storing the message and removing it from the network. In this respect, it is similar to virtual cut-through [19]. Jesshope [18] has proposed an algorithm for n dimensional meshes, by decomposing them into $2n$ virtual networks. Inside each virtual network, displacements along a given dimension are always made in the same direction, thus avoiding cycles and deadlock. A similar strategy for the development of adaptive and fault tolerant routing algorithms for k-ary n-cubes has been proposed in [22]. It is also based on the concept of virtual networks and requires the absence of cycles in the channel dependency graph. That strategy is only feasible for low-dimensional networks because the number of virtual channels increases very rapidly with the dimension.

An alternative way consists of recovering from deadlock. Reeves et al. [25] have used an abort-and-retry technique to remove messages blocked for longer than a certain threshold from the network. Aborted messages are introduced again into the network after a random delay. In [25] three adaptive routing strategies have been proposed and evaluated for a binary 8-cube.

Another approach is the use of multistage interconnection networks, in which deadlocks are easier to avoid. In [17] an adaptive algorithm based on interval routing has been proposed and evaluated. Although the results show that multistage networks outperform grids and hypercubes, the comparison does not take into account the implementation restrictions, in the way proposed in [7].

Finally, in [11] we have proposed a very simple methodology to design deadlock-free adaptive routing algorithms for wormhole networks. The routing algorithms obtained from the application of that methodology to 2D and 3D-meshes have been evaluated by simulation.

In summary, several adaptive routing algorithms have been proposed. In general, they outperform static algorithms. In order to avoid deadlocks, routing algorithms require either the absence of cycles in the channel dependency graph or additional hardware support. The first approach is too restrictive, limiting the number of alternative paths that can be used or requiring a lot of virtual channels. The second approach introduces additional delays (backtracking, message abortion, etc.).

This paper develops the theoretical background for the design of deadlock-free adaptive routing algorithms for wormhole networks. Some basic definitions and two theorems are proposed and proved, developing conditions to verify that an adaptive algorithm is deadlock-free, even when there are cycles in the channel dependency graph. Also, two design methodologies based on the above mentioned theorems are proposed. The first one supplies adaptive algorithms with a high degree of freedom, without increasing the number of physical channels. The second methodology is intended for the design of fault-tolerant algorithms. Some examples show the application of both methodologies, obtaining new adaptive routing algorithms. Finally, one of those algorithms is evaluated by simulation.

Section 2 develops the new theory for wormhole routing. Section 3 proposes two design methodologies, giving some examples of their application. The details concerning the simulation are described in section 4, showing the results in section 5. Finally, some conclusions are drawn.

2 Definitions and theorems

This section develops the theoretical background for the design of deadlock-free adaptive routing algorithms for networks using wormhole routing. This theory is also valid when messages are split into packets. However, as typical messages are short [7], we will consider that messages are not split. Otherwise, the following assumptions and definitions should refer to packets.

The basic assumptions are very similar to the ones proposed by Dally [10], except that adaptive routing is allowed. These assumptions are the following:

1. A node can generate messages destined for any other node at any rate.
2. A message arriving at its destination node is eventually consumed.
3. Wormhole routing is used. So, once a queue accepts the first flit of a message, it must accept the remainder of the message before accepting any flits from another message.
4. A node can generate messages of arbitrary length. Messages will generally be longer than a single flit.
5. An available queue may arbitrate between messages that request that queue, but may not choose among waiting messages.
6. A queue cannot contain flits belonging to different messages. After accepting a tail flit, a queue must be emptied before accepting another header flit. Then, when a message is blocked, its header flit will always occupy the head of a queue. This assumption is necessary to prove theorem 2. If it is not satisfied, it is easy to define a deadlocked configuration which invalidates that theorem, as will be seen in consideration 2 after the theorem proof.

7. The route taken by a message depends on its destination and the status of output channels (free or busy). At a given node, the routing function supplies a set of output channels based on the current and destination nodes. A selection from this set is made based on the status of output channels at the current node. So, *adaptive* routing will be considered.

Before proposing the theorems, some definitions are needed:

Definition 1 An *interconnection network* I is a strongly connected directed multigraph, $I = G(N, C)$. The vertices of the multigraph N represent the set of processing nodes. The arcs of the multigraph C represent the set of communication channels. More than a single channel is allowed to connect a given pair of nodes. Each channel c_i has an associated queue denoted $queue(c_i)$ with capacity $cap(c_i)$. The source and destination nodes of channel c_i are denoted s_i and d_i, respectively.

Definition 2 Let F be the set of valid *channel status,* $F = \{free, busy\}$. Let $T : C \rightarrow F$ be the status of the output channels in the network.

Definition 3 An *adaptive routing function* $R : N \times N \rightarrow \mathcal{P}(C)$, where $\mathcal{P}(C)$ is the power set of C, supplies a set of alternative output channels to send a message from the current node n_c to the destination node n_d, $R(n_c, n_d) = \{c_1, c_2, \ldots, c_p\}$. In general, p will be less than the number of output channels per node to restrict routing and obtain deadlock-free algorithms. As a particular case, $p = 1 \ \ \forall (n_c, n_d) \in N \times N, n_c \neq n_d$, defines a static routing function. Also, $R(n, n) = \emptyset, \ \forall n \in N$. The domain of R has been defined as $N \times N$ instead of $C \times N$ as in [10]. Otherwise, one of the theorems we propose could not be proved. This point will be discussed in detail after proving theorem 2 (see consideration 3). Also, defining the result of R as a set of channels (an element of $\mathcal{P}(C)$) allows us the use of the operations defined on sets. In particular, the $\in$ operator will be extensively used throughout this paper.

Definition 4 A *selection function* $S : \mathcal{P}(C \times F) \rightarrow C$ selects a free output channel (if any) from the set supplied by the routing function. From the definition, S takes into account the status of all the channels belonging to the set supplied by the routing function. The selection can be random or based on static or dynamic priorities. Also, in the same way the result of a static routing function may be a busy channel, if all the output channels are busy, any of them may be selected. The decomposition of the adaptive routing into two functions (routing and selection) will be critical while proving the theorems, because only the routing function determines whether a routing algorithm is deadlock-free or not. Then, the selection function will only affect performance. Moreover, it is possible to extend the definition of the selection function by taking into account additional information, either local to the node or remote. We will comment on this in section 3.

Definition 5 A routing function R for a given interconnection network I is *connected* iff

$$\forall x, y \in N, x \neq y, \exists c_1, c_2, \ldots, c_k \in C / \begin{cases} c_1 \in R(x, y) \\ c_{m+1} \in R(d_m, y), \quad m = 1, \ldots, k-1 \\ d_k = y \end{cases}$$

In other words, it is possible to establish a path between x and y using channels belonging to the sets supplied by R. Notice that the interconnection network is strongly connected, but it does not imply that the routing function must be connected.

Definition 6 A *routing subfunction* R_1 for a given routing function R and channel subset $C_1 \subseteq C$, is a routing function

$$R_1 : N \times N \rightarrow \mathcal{P}(C_1) / R_1(x, y) = R(x, y) \cap C_1 \quad \forall x, y \in N$$

Definition 7 Given an interconnection network I, a routing function R and a pair of channels $c_i, c_j \in C$, there is a *direct dependency* from c_i to c_j iff

$$c_i \in R(s_i, n) \; and \; c_j \in R(d_i, n) \; for \; some \; n \in N$$

that is, c_j can be used immediately after c_i by messages destined to some node n.

Definition 8 Given an interconnection network I, a routing function R, a channel subset $C_1 \subset C$ which defines a routing subfunction R_1 and a pair of channels $c_i, c_j \in C_1$, there is an *indirect dependency* from c_i to c_j iff

$$\exists c_1, c_2, \ldots, c_k \in C - C_1 / \begin{cases} c_i \in R_1(s_i, n) \\ c_1 \in R(d_i, n) \\ c_{m+1} \in R(d_m, n), & m = 1, \ldots, k-1 \\ c_j \in R_1(d_k, n) & for \; some \; n \in N \end{cases}$$

that is, it is possible to establish a path from s_i to d_j for messages destined to some node n. c_i and c_j are the first and last channels in that path and the only ones belonging to C_1. Then, c_j can be used after c_i by some messages. As c_i and c_j are not adjacent, some other channels belonging to $C - C_1$ are used between them. It must be noticed that, given three channels $c_i, c_k \in C_1$ and $c_j \in C - C_1$, the existence of direct dependencies between c_i, c_j and c_j, c_k, respectively, does not imply the existence of an indirect dependency between c_i, c_k.

Definition 9 A *channel dependency graph* D for a given interconnection network I and routing function R, is a directed graph, $D = G(C, E)$. The vertices of D are the channels of I. The arcs of D are the pairs of channels (c_i, c_j) such that there is a direct dependency from c_i to c_j. Notice that there are no 1-cycles in D, because channels are unidirectional.

Definition 10 An *extended channel dependency graph* D_E for a given interconnection network I and routing subfunction R_1 of a routing function R, is a directed graph, $D_E = G(C_1, E_E)$. The vertices of D_E are the channels that define the routing subfunction R_1. The arcs of D_E are the pairs of channels (c_i, c_j) such that there is either a direct or an indirect dependency from c_i to c_j.

Definition 11 A *sink* channel for a given interconnection network I and routing function R is a channel c_i such that

$$x \in N, c_i \in R(s_i, x) \Rightarrow x = d_i$$

In other words, all the flits that enter a sink channel reach their destination in a single hop. As a result, there are no outgoing arcs from a sink channel in any channel dependency graph, as can be easily seen from the definitions.

Definition 12 A *configuration* is an assignment of a set of flits to each queue, all of them belonging to the same message (assumption 6). The number of flits in the queue for channel c_i will be denoted $size(c_i)$. If the first flit in the queue for channel c_i is destined for node n_d, then $head(c_i) = n_d$. If the first flit is not a header and the next channel reserved by its header is c_j, then $next(c_i) = c_j$. Let $C_h \subseteq C$ be the set of channels containing a header flit at their queue head. Let $C_d \subseteq C$ be the set of channels containing a data or tail flit at their queue head. A configuration is *legal* iff

$$\forall c_i \in C \begin{cases} size(c_i) \leq cap(c_i) \\ size(c_i) > 0 \Rightarrow c_i \in R(s_i, head(c_i)) \end{cases}$$

For each channel, the queue capacity is not exceeded and all the flits stored in the queue (if any), which have the same destination, can reach the channel from the previous node using the routing function.

Definition 13 A *deadlocked configuration* for a given interconnection network I and routing function R is a nonempty legal configuration verifying the following conditions:

$$1)\ \forall c_i \in C_h \begin{cases} head(c_i) \neq d_i \\ size(c_j) > 0 \quad \forall c_j \in R(d_i, head(c_i)) \end{cases}$$

$$2)\ \forall c_i \in C_d \begin{cases} head(c_i) \neq d_i \\ size(next(c_i)) = cap(next(c_i)) \end{cases}$$

In a deadlocked configuration there is not any flit one hop from its destination. Header flits cannot advance because the queues for all the alternative output channels supplied by the routing function are not empty (see assumption 6). As a particular case (for disconnected routing functions), the routing function may not supply any output channel. Data and tail flits cannot advance because the next channel reserved by their message header has a full queue. No condition is imposed to empty channels. It must be noticed that a data flit can be blocked at a node even if there are free output channels to reach its destination. Also, in a deadlocked configuration, there is no message whose header flit has already arrived to its destination.

Definition 14 A routing function R for an interconnection network I is *deadlock-free* iff there is not any deadlocked configuration for that routing function on that network.

Two theorems are proposed. The first one is a straightforward extension of Dally's theorem for adaptive routing functions. The second one allows the design of adaptive routing functions with cyclic dependencies in their channel dependency graph. For each theorem, a sketch of the proof as well as the full proof are given.

Theorem 1 *A connected and adaptive routing function R for an interconnection network I is deadlock-free if there are no cycles in its channel dependency graph D.*

Proof sketch:

$\Leftarrow$ As the channel dependency graph for R is acyclic, it is possible to establish an order between the channels of C. As R is connected, the minimals of that order are also sinks. Suppose that there is a deadlocked configuration for R. Let c_i be a channel of C with a nonempty queue such that there are no channels less than c_i with a nonempty queue. If c_i is a minimal (that is, a sink) then the flit at the queue head can reach its destination in a single hop and there is no deadlock. Otherwise, using the channels less than c_i, the flit at the queue head of c_i can advance and there is not a deadlock.$\Box$

Proof:

$\Leftarrow$ Suppose that there are no cycles in D. Then, one can assign an order to the channels of C so that if $(c_i, c_j) \in E$ then $c_i > c_j$. Consider the channel(s) c_i such that

$$\forall c_j \in C, (c_i, c_j) \notin E$$

Such a channel c_i is a minimal of the order. Let us prove that it is a sink. If it were not a sink, as the routing function is connected, for any legal configuration with a header flit stored in the queue head of c_i

$$d_i \neq head(c_i) \Rightarrow \exists c_k \in C / c_k \in R(d_i, head(c_i))$$

As the configuration is legal then

$$c_i \in R(s_i, head(c_i)) \Rightarrow (c_i, c_k) \in E$$

contrary to the assumption that c_i is a minimal. So, $d_i = head(c_i)$ and c_i is a sink of D.

Suppose that there is a deadlocked configuration for R. Let c_i be a channel of C with a nonempty queue such that there is not any channel less than c_i with a nonempty queue. If c_i is a minimal, it is also a sink and then, all the flits stored in its queue will be destined to d_i and the flit at the head of the queue for c_i is not blocked. If c_i is not a minimal then

$$size(c_j) = 0 \quad \forall c_j \in C / c_i > c_j$$

Thus, the flit at the head of the queue for c_i is not blocked, regardless it is a header or a data flit, and there is no deadlock.□

There are some interesting considerations:

1. The theorem gives a sufficient but not necessary condition for an adaptive routing function to be deadlock-free. As will be seen later, the existence of cycles in the channel dependency graph does not imply the existence of a deadlocked configuration.

2. For most networks and routing functions, even for static ones, only a partial order between channels can be defined, based on the set E. In general, there will be more than a single sink in D.

3. As indicated above, in a legal configuration all the flits stored in a given queue have reached it using the routing function. Otherwise, the theorem cannot be proved. Consider, for instance, a configuration in which the queues of all the sink channels in D are full of flits destined to nodes not directly connected to those channels.

Theorem 2 *A connected and adaptive routing function R for an interconnection network I is deadlock-free if there exists a subset of channels $C_1 \subseteq C$ that*

defines a routing subfunction R_1 which is connected and has no cycles in its extended channel dependency graph D_E.

Proof sketch:

$\Leftarrow$ The case $C_1 = C$ is trivial. Otherwise $C_1 \subset C$. As the extended channel dependency graph for R_1 is acyclic, it is possible to establish an order between the channels of C_1. As R_1 is connected, the minimals of that order are also sinks. Suppose that there is a deadlocked configuration for R. There are two possible cases:

a) The queues for channels belonging to C_1 are empty. As R_1 is connected and the header flits are at queue heads, each header can be routed using channels belonging to C_1 and there is no deadlock.

b) The queues for channels belonging to C_1 are not empty. Let c_i be a channel of C_1 with a nonempty queue such that there are no channels less than c_i with a nonempty queue. Again, there are two possible cases:

b1) If c_i is a minimal (sink) then the flit at the queue head is not blocked and there is no deadlock.

b2) If c_i is not a minimal, all the channels of C_1 less than c_i will have empty queues, existing three possible cases:

b2.1) If c_i has a header at the queue head, it can be routed because R_1 is connected and there is no deadlock.

b2.2) If there is a data flit at the queue head of c_i and $next(c_i)$ belongs to C_1, that flit can also advance.

b2.3) If $next(c_i)$ belongs to $C - C_1$, we have to use the indirect dependencies in the extended channel dependency graph. Let c_k be the channel containing the header of the data flits contained in c_i. Then, it is possible to find a channel c_j belonging to C_1 to route that header, because R_1 is connected. In that case,

there is an indirect dependency from c_i to c_j ($c_i > c_j$), implying that c_j is empty and there is no deadlock.□

Proof:

$\Leftarrow$ Suppose that there exists a channel subset $C_1 \subseteq C$ which defines a routing subfunction R_1 and that R_1 is connected and there are no cycles in D_E. If $C_1 = C$ then $D_E = D$, because $C - C_1 = \emptyset$. Thus, there is not any cycle in D and R is deadlock-free by theorem 1. Otherwise $C_1 \subset C$. As there are no cycles in D_E, one can assign an order to the channels of C_1 so that if $(c_i, c_j) \in E_E$ then $c_i > c_j$. Similarly to theorem 1, it can be proved that the minimals of that order are also sinks.

Suppose that there is a deadlocked configuration for R. There are two possible cases:

a) The queues for channels belonging to C_1 are empty. Then, there will be channels belonging to $C - C_1$ with header flits at their queue heads. Let c_i be one of those channels. As R_1 is connected then

$$head(c_i) \neq d_i \Rightarrow \exists c_j \in C_1 / c_j \in R_1(d_i, head(c_i)) \Rightarrow$$
$$\exists c_j \in C / c_j \in R(d_i, head(c_i))$$

Also $size(c_j) = 0$ and R does not have a deadlock.

b) The queues for channels belonging to C_1 are not empty. Let c_i be a channel belonging to C_1 with a nonempty queue such that there are no channels less than c_i with a nonempty queue. Again, there are two possible cases:

b1) c_i is a minimal. As shown above, it is also a sink and then, all the flits stored in its queue will be destined to d_i and the flit at the head of the queue for c_i is not blocked.

b2) c_i is not a minimal. Then

$$size(c_j) = 0 \quad \forall c_j \in C_1 / c_i > c_j$$

existing three possible cases:

b2.1) c_i has a header at the queue head. Taking into account that R_1 is connected

$$head(c_i) \neq d_i \Rightarrow \exists c_j \in C_1 / c_j \in R_1(d_i, head(c_i)) \Rightarrow$$
$$\exists c_j \in C / c_j \in R(d_i, head(c_i))$$

Also

$$c_i > c_j \Rightarrow size(c_j) = 0$$

and R does not have a deadlock.

b2.2) c_i has a data flit at the queue head, not destined to d_i, and $next(c_i)$ belongs to C_1. Then

$$c_i > next(c_i) \Rightarrow size(next(c_i)) = 0$$

and R does not have a deadlock.

b2.3) c_i has a data flit at the queue head, not destined to d_i, and $next(c_i)$ belongs to $C - C_1$. Let $c_1, c_2, \ldots, c_k \in C - C_1$ be the set of channels reserved by the message after reserving c_i, c_k containing the message header. Those channels belong to $C - C_1$, because in C_1 there are no channels less than c_i with a nonempty queue.

$$\exists c_1, c_2, \ldots, c_k \in C - C_1 / \begin{cases} c_i \in R_1(s_i, head(c_k)) \\ c_1 \in R(d_i, head(c_k)) \\ c_{m+1} \in R(d_m, head(c_k)), \quad m = 1, \ldots, k-1 \end{cases}$$

As R_1 is connected

$$head(c_k) \neq d_k \Rightarrow \exists c_j \in C_1 / c_j \in R_1(d_k, head(c_k)) \Rightarrow$$
$$\exists c_j \in C / c_j \in R(d_k, head(c_k))$$

Thus, there is an indirect dependency from c_i to c_j ($c_i > c_j$), implying that $size(c_j) = 0$. Then, the header at the queue head for c_k is not blocked and R does not have a deadlock.□

Again, there are some interesting considerations:

1. The basic idea behind theorem 2 is that one can have an adaptive routing function with cyclic dependencies between channels, provided that there are alternative paths without cyclic dependencies to send a given flit towards its destination. As messages are several flits long, the extended channel dependency graph must be used to take into account the indirect dependencies.

2. If it were not necessary to empty a queue before accepting the header of another message, then there would be no guarantee that header flits occupy the queue heads and the theorem would not be valid. Consider, for instance, a set of four or more channels with cyclic dependencies between them and a configuration in which the queues of those channels are full, each one containing the tail of a message followed by a fragment of another message destined two nodes away. The rest of that message occupies part of the next channel queue and so on. That configuration is deadlocked because the header flits do not occupy the queue heads and cannot be routed using the alternative paths offered by the routing function.

3. If the routing function were defined as $R : C \times N \rightarrow \mathcal{P}(C)$, then the theorem would not be valid. Consider, for instance, two subsets of C, namely,

C_1 and $C - C_1$, and a routing function defined in such a way that all the messages arriving to a given node through a channel belonging to $C - C_1$ are routed through a channel belonging to the same subset. Suppose that there are cyclic dependencies between the channels belonging to $C - C_1$ and that C_1 defines a routing subfunction which is connected and has no cycles in its extended channel dependency graph. That routing function is not guaranteed to be deadlock-free.

4. The routing subfunction R_1 is not necessarily static. It can be adaptive.

3 Design methodologies

In this section we propose two methodologies for the design of deadlock-free adaptive routing algorithms. The generation of static deadlock-free routing algorithms requires restricting routing by removing arcs from the channel dependency graph D to make it acyclic. If it is not possible to make D acyclic without disconnecting the routing function, arcs can be added to D by splitting physical channels into a set of virtual channels, each one requiring its own buffer. This technique was introduced by Dally [10] to remove cycles from the channel dependency graph.

However, a physical channel can be split into more virtual channels than the ones strictly necessary to avoid deadlock [6, 11]. In such a case, the router can choose among several channels to send a message, reducing channel contention and message delay. Alternatively, more physical channels can be added to each node, increasing the network bandwidth and allowing the design of fault-tolerant adaptive routing algorithms.

A design methodology must supply a way to add channels following a regular pattern, also deriving the new routing function from the old one. A design

methodology based on theorem 1 has been presented in [11]. Although the algorithms designed with it behave better than the static ones, a higher degree of freedom can be obtained basing the design on theorem 2. Here we will present some more general methodologies for the design of deadlock-free adaptive routing algorithms.

Methodology 1 This methodology is intended to increase the number of valid alternative paths to send a message towards its destination without increasing the number of physical channels. In general, it will reduce channel contention and message delay. The steps are the following:

1. Given an interconnection network I_1, define a minimal path connected static routing function R_1 for it, following Dally's methodology and splitting physical channels into virtual ones, if necessary, to guarantee that R_1 is deadlock-free. Alternatively, define a minimal path connected adaptive routing function R_1 and selection function S_1, verifying that R_1 is deadlock-free using theorem 2. Let C_1 be the set of channels at this point.

2. Split each physical channel into a set of additional virtual channels. Let C be the set of all the (virtual) channels in the network. Let C_{xy} be the set of output channels from node x belonging to a minimal path from x to y. Define the new routing function R as follows:

$$R(x, y) = R_1(x, y) \cup (C_{xy} \cap (C - C_1)) \quad \forall x, y \in N$$

   that is, the new routing function can use any of the new channels belonging to a minimal path or, alternatively, the channels supplied by R_1. The selection function can be defined in any way.

3. Verify that the extended channel dependency graph for R_1 is acyclic. If it is, the routing algorithm is valid. Otherwise, it must be discarded, returning to step 1.

Step 1 establishes the starting point. We can use either a static or adaptive routing function as the basic one. Dally's theorem and theorem 2 can be used to verify that the basic function is deadlock-free. Step 2 indicates how to add more (virtual) channels to the network and how to define a new adaptive routing function from the basic one. Step 3 verifies whether the new routing function is deadlock-free or not. If the verification fails, the above proposed methodology may lead to an endless cycle. Then, it does not supply a totally mechanical way to design adaptive routing algorithms. Fortunately, all the algorithms we have designed up to now have passed the verification step. So, we wonder whether the step 3 is really necessary. This is still an open question.

The implementation of adaptive routing algorithms based on this methodology requires that physical channels are split into virtual ones. Dally [8] has described the implementation of virtual channels. That description is not restricted to the case of static routing algorithms. Then, all the implementation details described in [8] are applicable to this case.

It must be noticed that the methodology can also be applied by adding physical channels instead of virtual ones. The resulting network will be faster and more expensive, but the effective fault-tolerance will not increase. The reason is that the new routing function relies on the set of channels C_1 to guarantee that it is deadlock-free.

Methodology 2 This methodology is intended to increase fault-tolerance in a network. It will add physical channels, instead of splitting channels into virtual ones. Of course, it will also reduce channel contention and message delay. The steps are the following:

1. Given an interconnection network I_1, define a static or adaptive connected routing function R_1 for it, following Dally's methodology, the

above proposed methodology or verifying that R_1 is deadlock-free using theorem 2. Let C_1 be the set of channels at this point.

2. Duplicate each physical channel. If the original channel was split into several virtual channels, the duplicated channel will also be split into the same number of virtual channels. Let C_2 be the set of duplicated channels and C the set of all the channels. Let R_2 be a routing function identical to R_1, but defined using C_2 instead of C_1. Define the new routing function R as follows:

$$R(x,y) = R_1(x,y) \cup R_2(x,y) \quad \forall x,y \in N$$

that is, the new routing function can use any of the channels supplied by both, R_1 and R_2. Define the selection function giving to the channels belonging to C_1 and C_2 the same probability of use.

Again, step 1 supplies the basic routing function and step 2 adds alternative paths. As can be easily seen, R_2 does not add any cycle to the extended channel dependency graph for R_1. Then, R is deadlock-free.

The duplication of channels defines an interconnection network $I_2 = G(N, C_2)$, which is identical to I_1 and shares the same set of nodes N. However, C_1 and C_2 are disjointed sets. R_2 has the same properties as R_1. Also,

$$R_1(x,y) = R(x,y) \cap C_1 \quad \forall x,y \in N$$
$$R_2(x,y) = R(x,y) \cap C_2 \quad \forall x,y \in N$$

So, one can find, at least, two subfunctions of R, which allow us the application of theorem 2 to guarantee that R is deadlock-free. Then, the theorem can be applied even if we remove some channels either from C_1 or from C_2.

However, the set of nodes is the same for I_1 and I_2. It seems that the proposed methodology is not tolerant to node faults. But, provided that R_1

and R_2 are adaptive routing functions, in general there will be alternative paths to reach the destination node (assuming that it is not the faulty one). Of course, some mechanism is needed to identify faulty channels, marking them as busy, and faulty nodes, marking all the channels connected to them as busy and avoiding to send messages to them. It must be noticed that if there is not any faulty node, the information about faulty channels can be recorded locally.

Finally, step 2 can be applied several times, duplicating each channel as many times as desired.

The proposed methodologies are very simple to apply. They illustrate the power of the theorems. More complex design methodologies can be defined based on the same theoretical background.

As an example, we will present a design based on the above proposed methodologies. Consider a binary n-cube. We will study three cases: a) applying methodology 1; b) applying methodology 2; c) applying methodologies 1 and 2.

a) For the step 1 we can use the conventional static routing algorithm for the binary n-cube. It forwards messages crossing the channels in order of decreasing dimensions. It is well known that this routing function is connected and deadlock-free.

For the step 2, consider that each physical channel c_i has been split into k virtual channels, namely, $a_{i,1}, a_{i,2}, \ldots, a_{i,k-1}, b_i$. Let C_1 be the set of b channels. The algorithm obtained applying the step 2 can be stated as follows: Route over any useful dimension using any of the a channels. If all of them are busy, route over the highest useful dimension using the corresponding b channel. A useful dimension is one that forwards a message nearer to its destination.

Fig. 1 shows the extended channel dependency graph for R_1 on a 3-cube.

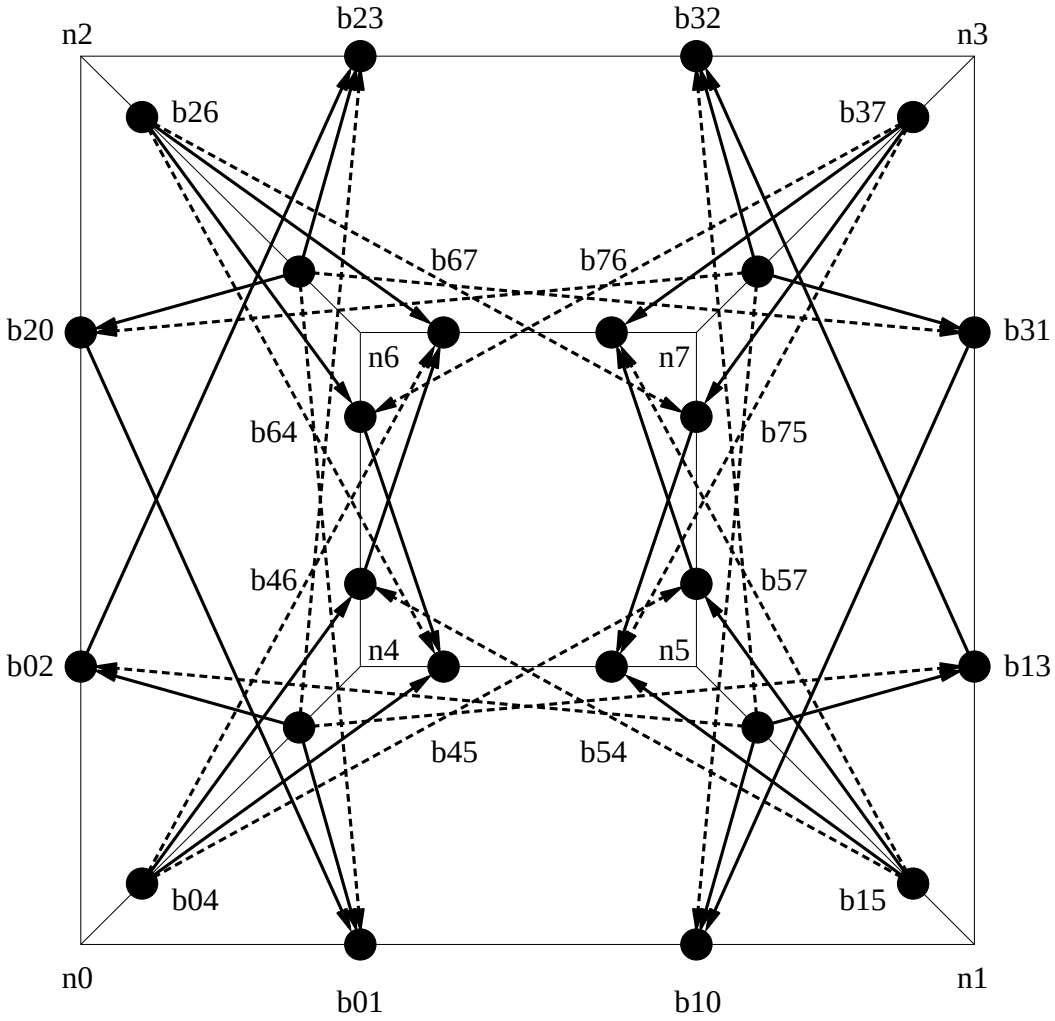

Figure 1: Extended channel dependency graph for R_1

Black circles represent the unidirectional channels belonging to C_1 and are labeled as bij, where i and j are the source and destination nodes, respectively. As a reference, channels are also represented by thin lines, horizontal and vertical ones corresponding to dimensions 0 and 1, respectively. Also, the nodes of the 3-cube have been labeled as nk, where k is the node number. Thick lines represent channel dependencies, dashed arrows corresponding to indirect dependencies. It can be seen that the graph is acyclic. Then, R is deadlock-free. It must be noticed that the channel dependency graph for R (not shown) has cycles.

As virtual channels share a single physical channel, the former algorithm effectively allows messages to cross the physical channels corresponding to the n-cube dimensions in any order, increasing the number of alternative paths and decreasing network contention. The performance of that algorithm has been evaluated by simulation. The results are presented in the next sections.

b) Assume that step 1 is applied as in case a), obtaining the conventional static routing algorithm.

For the step 2, consider that each physical channel c_i has been duplicated, obtaining a new channel g_i. The algorithm obtained applying the step 2 can be stated as follows: Route over the highest useful dimension using the corresponding c or g channels.

That algorithm increases the tolerance to faulty channels, but it does not take advantage of alternative minimal paths.

c) Assume that we apply the methodology 1, obtaining the algorithm proposed in case a). That algorithm constitutes the step 1.

For the step 2, consider that each physical channel c_i has been duplicated, obtaining a new channel g_i, which is also split into k virtual channels, namely,

$e_{i,1}, e_{i,2}, \ldots, e_{i,k-1}, f_i$. The algorithm obtained applying the step 2 can be stated as follows: Route over any useful dimension using any of the a or e channels. Alternatively, route over the highest useful dimension using the corresponding b or f channels.

That algorithm has the advantages of the previous ones at the cost of a slightly more complicated circuitry.

The number of virtual channels per physical channel only affects the performance. We will comment on this in the next sections. Also, as stated in section 2, the selection function only affects the performance. It is not necessary to give a higher priority to the channels in the acyclic dependency subgraph, because when the remaining channels are busy, those ones will be used. In general, a higher performance is achieved when the channels in the cyclic dependency subgraph are given a higher priority, because they usually offer a larger number of alternative paths. Moreover, when several routing options are available, selecting a virtual channel in such a way that channel multiplexing is minimized usually reduces the average message delay.

Finally, the selection function can be extended by including additional information in its domain. For instance, for the algorithm obtained in case a), it is possible to favor the a channels connecting to the neighbor with a higher number of free channels in useful dimensions. This selection function is inspired in an algorithm proposed in [25], the main difference being that our algorithm does not require a complex mechanism to abort messages because it is deadlock-free.

4 Evaluation methodology

The routing algorithm obtained in case a) supplies a large number of routing options to forward a message toward its destination. Also, the multiplexing of physical channels reduces channel contention by allowing several messages to share a channel. However, each virtual channel only has a fraction of the total bandwidth, increasing the term L/B. So, there are opposite effects and the behavior of the new routing algorithm is not intuitive, being necessary to evaluate its performance under different load conditions. In what follows, this algorithm will be referred to as adaptive algorithm. The static routing algorithm for the binary n-cube (e-cube) has been evaluated under the same conditions for comparison purposes. For this algorithm we have also analyzed the effect of the virtual channel flow control mechanism proposed in [6], showing the effect of channel multiplexing.

Once the topology has been selected, the algorithm performance depends on several parameters, including channel bandwidth, network size, message traffic, message length and the number of virtual channels per physical channel. Let us analyze that dependency in order to select the conditions for the evaluation.

Firstly, we are interested in comparing different routing algorithms rather than obtaining absolute performance measures. Then, channel bandwidth is not important, provided a uniform criterion is used. The criterion we have used consists of using a constant channel bandwidth equal to one flit per clock cycle.

In general, network size has a considerable influence on network performance. Then, taking into account the maximum sizes of current multicomputers, we have mainly evaluated networks with 4096 nodes, also obtaining some results for networks ranging from 64 to 4096 nodes. It is especially important

to see whether the routing algorithms scale well with network size.

Message traffic and message length are application and operating system dependent. As these parameters are very difficult to model, we have used a uniform distribution for message destination, trying to model the worst case. For each routing algorithm, the flit generation rate ranges from a small value to saturation. For message length, a constant value equal to 16 flits has been used.

The number of virtual channels per physical channel is a design parameter. We have analyzed the effect of this parameter, holding the total queue size per physical channel constant, as proposed in [6].

Instead of analytic modeling, simulation has been used to evaluate the routing algorithms, because the model can more faithfully represent a hardware implementation, taking into account details like channel multiplexing, partial buffering and delays in blocked messages. The simulation methodology we have used is summarized in the following sections.

4.1 Multicomputer model

Our simulator models different topologies and network sizes up to 16K nodes. Each node consists of a processor, a crossbar, a router and several channels. Processors can generate messages at any rate. Message format will be described below. Message reception is buffered, allowing the storage of messages independently of the processes which have to receive them. The simulator takes into account memory contention, limiting the number of messages that can be sent or received simultaneously. Binary n-cubes have a large number of channels in the network bisection, limiting channel bandwidth. Because of that, we have assumed that up to four messages can be sent or received simultaneously.

The crossbar allows multiple messages to traverse a node simultaneously without interference. It takes one clock cycle to transfer a flit from an input queue to an output queue.

The router decides when to transfer a message, determining the output channel as a function of the destination node, the current node and the output channel status. Wormhole routing is used. The router can only process a message header at a time. If there is no contention for the router, it takes one clock cycle to compute the output channel. Otherwise, access is round robin. When a message gets the router, but cannot be routed because all the alternative output channels are busy, it must wait until its next turn.

Physical channels are split into up to four virtual channels. Each virtual channel has queues of equal size at both ends. The total queue size associated to each physical channel is held constant. Virtual channels are assigned the physical channel cyclically, only if they can transfer a flit. So, channel bandwidth is shared among the virtual channels requesting it. It must be noticed that blocked messages and messages waiting for the router do not consume any channel bandwidth.

4.2 Message generation

A message consists of a header, some data and a tail. For the sake of simplicity, we have assumed that the header occupies a single flit, thus a message may be routed as soon as the first flit arrives to a node.

The message generation rate is constant and the same for all the nodes. After generating a message, each node waits a random number of clock cycles before generating the next message. The number of cycles is uniformly distributed between two simulation parameters. The inverse of the mean value of

these parameters will be referred to as message generation rate. The product of the message generation rate by the message length is the flit generation rate and can be measured in flits per node and clock cycle. As indicated above, up to four messages are allowed to leave each node simultaneously. If there are four outgoing messages and a new message is generated, it is queued.

Once the network has reached a steady state, the flit generation rate is equal to the flit reception rate, unless the network is saturated. In this case, both rates differ and source queues grow. We will refer to the flit reception rate as traffic, because it is the actual traffic through the network. It must be noticed that traffic is not an independent variable.

4.3 Performance measures

The most important performance measures are delay and throughput. Delay is the additional latency required to transfer a message with respect to an idle network. It is measured in clock cycles. The message latency lasts since the message is introduced in the network until the last flit is received at the destination node. An idle network means a network without message traffic and, thus, without channel multiplexing.

Throughput is usually defined as the maximum amount of information delivered per time unit. However, the network may be unstable when traffic reaches the maximum value. In this case, increasing the flit generation rate may reduce the traffic, as will be seen in the next section. Then, throughput is defined as the saturation traffic. As we have normalized channel bandwidth, throughput is measured in flits per node and clock cycle, making it independent of network size.

4.4 Validation

Each simulation has collected data from 100000 messages, discarding the messages corresponding to a transient period required to reach a steady state. The number of discarded messages ranged from 50000 to 240000, depending on the traffic. After the transient period, the fluctuation of the performance measures for 4096-node networks was lower than 2.4%. Several simulations were rerun to test the reproducibility. The performance measures did not change by more than 1%.

5 Simulation results

The first simulation results for the adaptive algorithm proposed in section 3 (case a) were presented in [12], showing an important improvement over the static algorithm for 16-flit and 256-flit messages. However, those results did not consider partial buffering nor memory contention. Also, a single message was allowed to leave a node at a time, preventing network saturation. Then, the throughput gain achieved by the adaptive algorithm was not properly displayed. The simulation results presented in this paper overcome the former limitations, also analyzing the effect of the number of virtual channels per physical channel.

Firstly, we will compare the static and adaptive routing algorithms on a binary 12-cube under the same conditions. The adaptive algorithm has been simulated with three virtual channels per physical channel, while the static one has been simulated with one and three virtual channels. Fig. 2 shows the average message delay versus traffic for 4096 nodes. The independent variable is the flit generation rate, traffic and average message delay being measures.

The use of virtual channels drastically increases the throughput achieved by the static algorithm. This result was already pointed out by Dally [6]. Nevertheless, the adaptive algorithm still improves the throughput by 35%, using the same number of virtual channels. Moreover, the adaptive algorithm achieves an important reduction in message delay with respect to the static algorithm. For the same number of virtual channels, the average delay for the adaptive algorithm is as much as 35% of the delay obtained for the static algorithm. Also, the standard deviation of message delay (not shown) is smaller for the adaptive algorithm.

It is interesting to see the effect of the number of virtual channels per physical channel. Fig. 3 also shows the average message delay versus traffic for 4096 nodes. The curves correspond to the static and adaptive algorithms, both with two, three and four virtual channels per physical channel. Again, traffic and average message delay are measures. Then, the curves do not represent functions. As can be seen, there are two distinct average delay values corresponding to a range of traffic values for one of the curves.

For the static algorithm, increasing the number of virtual channels also increases throughput. However, this increment is small (less than 4%) when the fourth virtual channel is added. This addition also increases the average message delay by as much as 22%. So, it seems that three virtual channels per physical channel is the best choice. This behavior agrees with the one presented by Dally in [8] for a 16-ary 2-cube network. This similarity is interesting, taking into account the very different number of dimensions of both networks.

For the adaptive algorithm, the influence of the number of virtual channels is similar. The average message delay also increases by as much as 22% when the fourth virtual channel is added. However, there are two important differences. The most noticeable one consists of a drastic increase in message

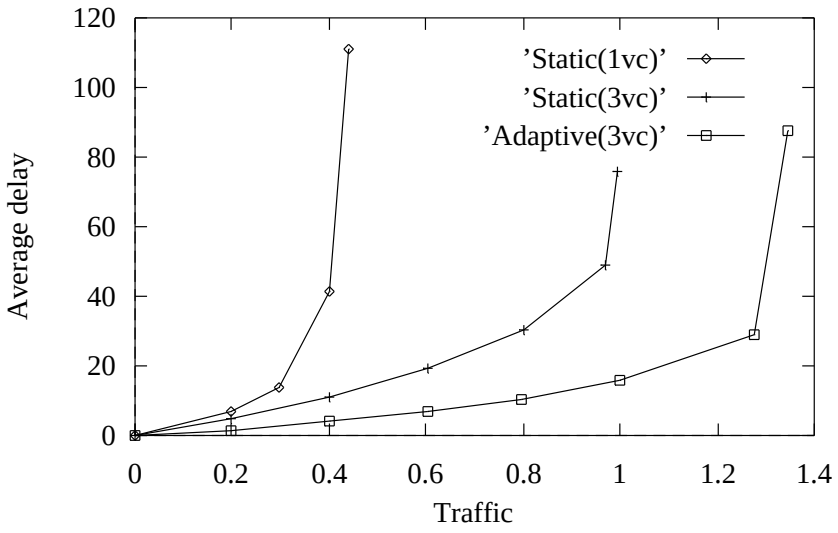

Figure 2: Average message delay versus traffic for 4096 nodes

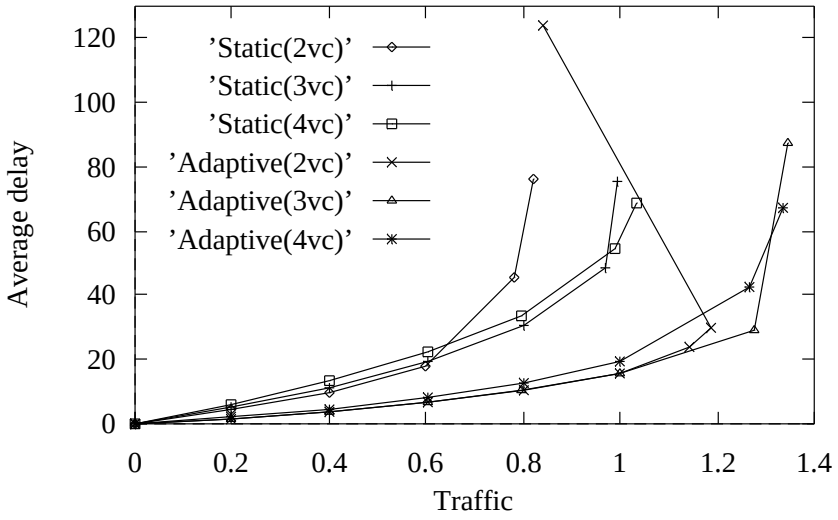

Figure 3: Effect of the number of virtual channels per physical channel on the average message delay for 4096 nodes

delay and a reduction in traffic when the adaptive algorithm with two virtual channels reaches saturation. The second difference refers to throughput. It does not increase when the fourth virtual channel is added. Again, three virtual channels per physical channel seems the best choice. These results will be explained later.

It is also interesting to know whether the routing algorithms scale well with network size. Fig. 4 shows the throughput as a function of network size. For the range of sizes we have analyzed, throughput decreases by 32% for the static algorithm. When virtual channels are used, throughput only decreases by 14%. The adaptive algorithm scales even better, throughput decreasing by as much as 6%. Additionally, the use of virtual channels in the static algorithm increases throughput by a factor ranging from 1.8 to 2.2. Also, the adaptive algorithm increases throughput over the static algorithm with a single virtual channel by a factor ranging from 2.2 to 3.

Finally, Fig. 5 shows the average message delay as a function of network size, holding traffic constant. This figure clearly shows that the static algorithm with a single virtual channel does not scale well with network size. However, when virtual channels are used, message delay increases almost linearly with the logarithm of the network size. Message delay increases by 14% each time the network size is doubled. This result is improved by the adaptive algorithm, delay increasing linearly by 3.3% each time the network size is doubled. It must be noticed that these results have been obtained holding channel bandwidth constant while network size increases.

Let us analyze the former results. The static algorithm with a single virtual channel has no choice, because the routing function supplies a single channel. As network traffic increases, the probability for a message header of finding a free output channel decreases, increasing channel contention and message

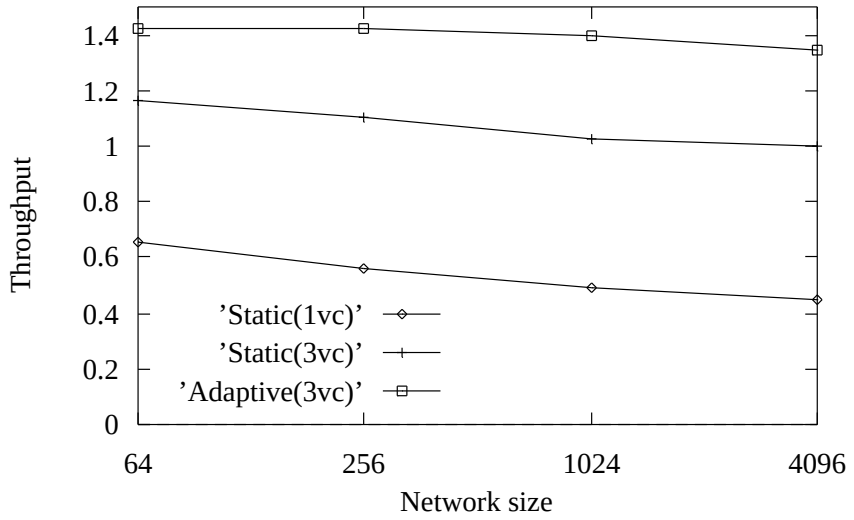

Figure 4: Throughput per node as a function of network size

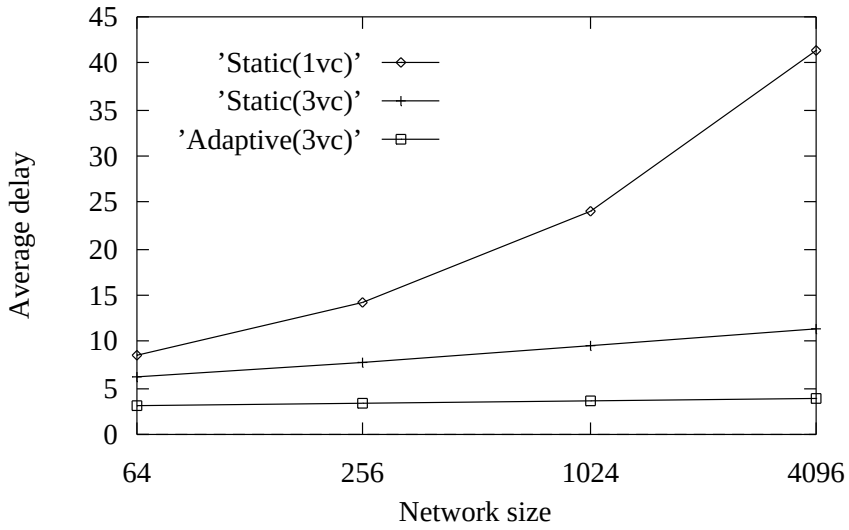

Figure 5: Average message delay as a function of network size for flit generation rate equal to 0.4 flits/cycle/node

delay. Moreover, when a message is blocked, the bandwidth of all the channels it has reserved is wasted. This is the main reason why throughput is relatively small. Also, as network size increases, there are more messages flowing through the network, increasing channel contention and message delay.

As stated above, the use of virtual channels has two opposite effects: it reduces channel contention by supplying more routing options, but it can increase message delay because channel bandwidth is shared among the requesting messages. When traffic is intense, the reduction in channel contention has a larger effect than bandwidth sharing. Moreover, the virtual channel mechanism has an important advantage. When a message is blocked, the virtual channels reserved by it do not consume any channel bandwidth. As a consequence, the use of virtual channels increases throughput considerably and message delay scales well with network size. However, adding more virtual channels yields diminishing returns. As pointed out by Dally [8], adding virtual channels while holding storage constant increases the usage of virtual channels. For instance, when the static algorithm with four virtual channels reaches saturation, messages in transit occupy 100000 virtual channels. The same algorithm with three virtual channels only has 78000 busy channels.

On the other hand, as soon as a single channel occupied by a message is multiplexed, the whole message is slowed-down. An uneven distribution for channel multiplexing produces bottlenecks. As a consequence, the bandwidth of the less multiplexed channels is partly wasted. Then, for low flit generation rates, adding virtual channels increases message delay. This increment is small because messages are short and only a few flits are affected by bottlenecks. Also, message headers must be routed, introducing another bottleneck. Routing operations, together with partial buffering, partly hide the effect of uneven channel multiplexing until the message header reaches its destination. For long

messages, the negative effects of channel multiplexing are more noticeable, as shown in [12].

The adaptive algorithm also uses virtual channels. Thus, it inherits the basic properties of this flow control mechanism. However, it permits a message to cross the n-cube 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 from 70% (static algorithm with four virtual channels) to 90% (adaptive algorithm with three virtual channels). As a consequence, throughput also increases. Moreover, the selection function helps in reducing channel multiplexing. First, it looks for a virtual channel belonging to a free physical channel. Also, it assigns the a channels a higher priority. Then, channels are only multiplexed when network traffic is intense, considerably reducing the negative effects of virtual channels.

As pointed out before, there is an important performance degradation when the adaptive algorithm with two virtual channels reaches saturation. We have carefully analyzed this case. It must be noticed that theorem 2 allows the existence of cyclic dependencies between channels. Deadlocks are avoided by relying on a channel subset to drain messages involved in cyclic dependencies. If the number of virtual channels per physical channel is insufficient, channel contention increases the number of blocked messages when traffic is intense. It is possible to reach a situation such that messages block cyclically faster than they are drained. In this situation, some messages remain blocked for long periods, increasing message delay drastically. Also, most channels are occupied by blocked messages, reducing the effective network bandwidth. Throughput is reduced accordingly. This is the price to pay for allowing the existence of cyclic dependencies between channels. Fortunately, this problem disappears

when more virtual channels are added.

We also pointed out before that adding a fourth virtual channel to the adaptive algorithm does not increase throughput, as opposed to the static algorithm. The adaptive routing function supplies several routing options, reducing channel contention with respect to the static algorithm. Then, adding more options does not reduce contention significantly. It seems that three virtual channels per physical channel is the best option. The existence of an optimal number of virtual channels per physical channel has already been suggested in [11]. However, that optimal value may change for another distribution of message destination. Currently, we are working on this subject.

Finally, the number of choices at each routing step increases with network size, almost nullifying the negative effect produced by a higher traffic. Because of that, the adaptive algorithm scales very well with network size. Unfortunately, it is not feasible to implement binary n-cubes holding channel bandwidth constant as network size increases.

6 Conclusions

The theoretical background for the development of deadlock-free adaptive routing algorithms has been proposed for wormhole networks. Firstly, a straightforward extension of Dally's theorem has been presented, allowing the design of adaptive algorithms. However, the absence of cycles in the channel dependency graph is too restrictive.

Theorem 2 gives a more flexible condition for the development of adaptive algorithms, by allowing the existence of cycles in the channel dependency graph. The only requirement is the existence of a channel subset which defines a connected routing subfunction with no cycles in its extended channel

dependency graph.

To simplify the application of the theorems, two design methodologies have been proposed. The first one supplies adaptive algorithms with a high degree of freedom. The second one gives a way to design fault-tolerant routing algorithms. Both methodologies can be easily combined. Also, an example showing three alternative ways to apply the proposed design methodologies is presented. This example derives new adaptive routing algorithms for the binary n-cube.

Finally, one of those routing algorithms has been evaluated by simulation, showing an important reduction in message delay with respect to the static algorithm, even with the same number of virtual channels. Throughput also increases considerably. Moreover, the new algorithm scales very well with network size, showing an almost null increment in message delay as size increases. Finally, the effect of the number of virtual channels has been analyzed.

Acknowledgment

Most assumptions, some definitions and theorem 1 are inspired in the theory developed by W. J. Dally for static routing [10]. I also thank him for his encouraging comments and suggestions to improve the paper. Finally, I would like to thank the anonymous referees for their helpful comments and suggestions.

References

[1] W.C. Athas and C.L. Seitz, "Multicomputers: Message-passing concurrent computers," *IEEE Comput. Mag.*, vol. 21, no. 8, pp. 9–24, Aug. 1988.

[2] S. Borkar et al., "iWarp: An integrated solution to high-speed parallel computing," in *Proc. Supercomputing'88*, Nov. 1988.

[3] W. Chou, A.W. Bragg, and A.A. Nilsson, "The need for adaptive routing in the chaotic and unbalanced traffic environment," *IEEE Trans. Commun.*, vol. COM-29, no. 4, pp. 481–490, Apr. 1981.

[4] E. Chow, H. Madan, J. Peterson, D. Grunwald, and D.A. Reed, "Hyperswitch network for the hypercube computer," in *Proc. 15th Annu. Int. Symp. Comput. Architecture*, May–June 1988.

[5] W.J. Dally, *A VLSI Architecture for Concurrent Data Structures*. Boston, MA: Kluwer Academic, 1987.

[6] W.J. Dally, "Virtual-channel flow control," in *Proc. 17th Annu. Int. Symp. Comput. Architecture*, May 1990.

[7] W.J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Trans. Comput.*, vol. C-39, no. 6, pp. 775–785, June 1990.

[8] W.J. Dally, "Virtual-channel flow control," *IEEE Trans. Parallel Distributed Syst.*, vol. 3, no. 2, pp. 194–205, Mar. 1992.

[9] W.J. Dally and C.L. Seitz, "The torus routing chip," *Distributed Comput.*, vol. 1, no. 3, pp. 187–196, Oct. 1986.

[10] W.J. Dally and C.L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. C-36, no. 5, pp. 547–553, May 1987.

[11] J. Duato, "Deadlock-free adaptive routing algorithms for multicomputers," *Tech. et Sci. Informatiques*, vol. 10, no. 4, 1991.

[12] J. Duato, "Deadlock-free adaptive routing algorithms for multicomputers: evaluation of a new algorithm," in *Proc. 3rd IEEE Int. Symp. Parallel Distributed Processing*, Dec. 1991.

[13] D. Gelernter, "A DAG-based algorithm for prevention of store-and-forward deadlock in packet networks," *IEEE Trans. Comput.*, vol. C-30, pp. 709–715, Oct. 1981.

[14] C. Germain-Renaud, "Etude des mécanismes de communication pour une machine massivement parallèle: MEGA," Ph.D. dissertation, Université de Paris-Sud, Centre d'Orsay, 1989.

[15] K.D. Gunther, "Prevention of deadlocks in packet-switched data transport systems," *IEEE Trans. Commun.*, vol. COM-29, pp. 512–524, Apr. 1981.

[16] W.D. Hillis, *The Connection Machine.* Cambridge, MA: MIT Press, 1985.

[17] H. Hofestädt, A. Klein, and E. Reyzl, "Performance benefits from locally adaptive interval routing in dynamically switched interconnection networks," in *Proc. 2nd European Distributed Memory Comput. Conf.*, Apr. 1991.

[18] C.R. Jesshope, P.R. Miller, and J.T. Yantchev, "High performance communications in processor networks," in *Proc. 16th Annu. Int. Symp. Comput. Architecture*, May–June 1989.

[19] P. Kermani and L. Kleinrock, "Virtual cut-through: a new computer communication switching technique," *Comput. Networks*, vol. 3, pp. 267–286, 1979.

[20] C.K. Kim and D.A. Reed, "Adaptive packet routing in a hypercube," in *Proc. 3rd Conf. Hypercube Concurrent Comput. Appl.*, Jan. 1988.

[21] H.T. Kung, "Deadlock avoidance for systolic communication," in *Proc. 15th Annu. Int. Symp. Comput. Architecture*, May–June 1988.

[22] D.H. Linder and J.C. Harden, "An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes," *IEEE Trans. Comput.*, vol. C-40, no. 1, pp. 2–12, Jan. 1991.

[23] P.M. Merlin and P.J. Schweitzer, "Deadlock avoidance in store-and-forward networks – I: Store-and-forward deadlock," *IEEE Trans. Commun.*, vol. COM-28, pp. 345–354, Mar. 1980.

[24] S. Ragupathy, M.R. Leutze, and S.R. Schach, "Message routing schemes in a hypercube machine," in *Proc. 3rd Conf. Hypercube Concurrent Comput. Appl.*, Jan. 1988.

[25] D.S. Reeves, E.F. Gehringer, and A. Chandiramani, "Adaptive routing and deadlock recovery: a simulation study," in *Proc. 4th Conf. Hypercube Concurrent Comput. Appl.*, Mar. 1989.