A Necessary and Sufficient Condition for Deadlock-Free Routing in Cut-Through and Store-and-Forward Networks

José Duato

Abstract—This paper develops the theoretical background for the design of deadlock-free adaptive routing algorithms for virtual cut-through and store-and-forward switching. This theory is valid for networks using either central buffers or edge buffers. Some basic definitions and three theorems are proposed, developing conditions to verify that an adaptive algorithm is deadlock-free, even when there are cyclic dependencies between routing resources. Moreover, we propose a necessary and sufficient condition for deadlock-free routing. Also, a design methodology is proposed. It supplies fully adaptive, minimal and non-minimal routing algorithms, guaranteeing that they are deadlock-free.

The theory proposed in this paper extends the necessary and sufficient condition for wormhole switching previously proposed by us. The resulting routing algorithms are more flexible than the ones for wormhole switching. Also, the design methodology is much easier to apply because it automatically supplies deadlock-free routing algorithms.

Index Terms—Adaptive routing, deadlock avoidance, design methodologies, interconnection networks, store-and-forward, virtual cut-through.

1 Motivation

Most computer networks use a store-and-forward switching technique to control the flow of information through the network [38]. Also, several commercial multicomputers [3] are based on that switching technique. Each time a packet reaches a node, it is completely buffered in local memory, and the processor interrupted to execute the routing algorithm.

More sophisticated switching techniques have been developed, drastically reducing the average latency (time required to transfer a message from the source node to the destination node). Current multicomputers use a switching technique known as wormhole [9]. It splits messages into smaller units, known as flow control units, or flits [8]. 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. When the header is routed, the router can offer one or more alternative paths, depending on whether the routing algorithm is deterministic or adaptive. If all the channels offered by the router are busy, the header is blocked until one of those channels is freed; the flow control within the network blocks the trailing flits. Currently, wormhole is the switching technique that achieves the lowest message latency. See [31] for a detailed analysis of wormhole switching and [15] for a study of adaptive routing protocols.

• *The author is with the Departamento de Ingenieria de Sistemas, Computadores y Automatica, Universidad Politecnica de Valencia, 46071-Valencia, Spain. Email: jduato@gap.upv.es.*

Manuscript received Dec. 27, 1994; accepted Aug. 24, 1995.
For information on obtaining reprints of this article, please send e-mail to: transpds@computer.org, and reference IEEECS Log Number D95140.

Virtual cut-through [23] was initially proposed for computer networks. It also reduces latency by pipelining packet transmission across successive channels. When the packet header arrives at a node, it is routed without waiting for the rest of the packet. The main difference with respect to wormhole is that packets are buffered in intermediate nodes when the packet header cannot be routed because the output channels are busy. Packets can be stored in buffers allocated in memory or in hardware buffers. These buffers can be shared by all the channels (central buffers). Alternatively, each buffer can be associated with a particular channel (edge buffers). The use of hardware buffers in virtual cut-through networks requires that messages are split into fixed length packets.

Assuming that the switching technique has no impact on clock frequency, virtual cut-through achieves a higher throughput than wormhole [29], [36]. The reason being that when a packet is blocked, it remains in the network if wormhole is used. Virtual cut-through allows blocked packets to be buffered and removed from the network, thus releasing channels that can be used by other packets. However, virtual cut-through requires large buffers. These buffers occupy more silicon area. Also, large buffers usually increase propagation delay, thus slowing down clock frequency.

As VLSI technology progresses, more transistors are available and larger buffers can be integrated. Moreover, as circuits become faster, channel propagation delay is becoming the bottleneck [1]. As a consequence, the impact of node delay on performance will decrease over time, and virtual cut-through may be the choice for the future, provided that buffer design is optimized. This is specially true for distributed shared-memory multiprocessors with hardware cache coherence protocols because messages are very short [32]. Thus, it is not necessary to split messages into packets. Several researchers have recently analyzed the behavior of virtual cut-through routers [4], [2], [30]. Moreover, some experimental multiprocessors under development use this switching technique [32].

2 Introduction

Deadlocks may appear if the routing algorithms are not carefully designed. A deadlock occurs when no message or packet can advance toward its destination because the queues of the message system are full. Obviously, deadlocks arise because the number of resources is finite.

Many deadlock-free routing algorithms have been developed for store-and-forward computer networks. Most of them require the use of central queues, restricting buffer allocation [7], [16], [19], [21], [28], [33]. These algorithms are also applicable to virtual cut-through networks with central queues. Although algorithms that use central queues require less storage than those using edge buffers, central queues can become a bottleneck. So, algorithms that use edge buffers usually achieve a higher performance. Several researchers have proposed the use of edge buffers for multicomputer networks [22], [25].

The restriction of buffer allocation, although it avoids deadlock, can increase traffic jams, especially in heavily loaded networks. In order to avoid congested regions of the network, an adaptive routing algorithm can be used. Adaptive strategies have been shown to outperform deterministic strategies in store-and-forward switching [5], in packet-switched communications [24], [35] and in wormhole switching [11], [12].

When adaptive routing is used, deadlocks can be avoided in virtual cut-through and store-and-forward switching by misrouting packets in the presence of congestion. This technique, known as deflection or hot-potato routing [20], requires the use of non-minimal paths. Examples of deflection routers for virtual cut-through networks can be found in [29], [17], [26], [27]. This mechanism relies on the existence of as many input channels as output channels in each node. Incoming packets will always find a free output channel from the switch, sending packets away from the destination if necessary. If the only free output channel is the one connecting to local memory, the packet is buffered in the node. As mentioned above, this mechanism requires the use of misrouting, thus wasting channel bandwidth when the network is heavily loaded. Additionally, livelock freedom is only guaranteed in a probabilistic manner [27]. However, peak values for packet latency are not excessively high.

Other proposals use an injection limitation strategy based on buffer occupancy. The ring protocol [37] prevents the injection of a new packet into a node of a ring if it fills the local queue. Thus, at least one packet from the previous node is able to advance. This mechanism can be easily generalized for topologies with Hamiltonian paths.

The most common way to prevent deadlock consists of eliminating cyclic dependencies in the use of resources. Preventing cyclic dependencies between resources is a sufficient condition for deadlock avoidance. However, it is too restrictive. More efficient routing algorithms can be obtained by allowing cyclic dependencies between resources. Pifarre et al. [33], [34] proposed a theory for the design of deadlock-free adaptive routing algorithms in packet switched networks with central queues. Starting from an acyclic queue dependency graph, they showed how to develop fully adaptive routing algorithms without producing deadlock.

Simultaneously, a similar theory was proposed by us for wormhole switching [10], [11] and store-and-forward switching with edge buffers [10]. This theory proposes sufficient conditions to verify that an adaptive algorithm is deadlock-free, even when there are cyclic dependencies between channels. We also proposed two design methodologies. The first one supplies fully adaptive routing algorithms and it is very similar to the one proposed in [33], [34].

Cypher and Gravano [6] proposed a necessary condition for deadlock-free adaptive routing in packet switched networks with central queues. They showed that an adaptive deadlock-free packet routing algorithm can always be restricted, obtaining an oblivious deadlock-free routing algorithm. This is not a sufficient condition, as will be seen in Section 4.5.

More recently, we proposed a necessary and sufficient condition for deadlock-free routing in networks using wormhole switching [13]. Although this theory supplies sufficient conditions for deadlock-freedom when it is applied to virtual cut-through and store-and-forward switching, more flexibility can be obtained by considering the behavior of those switching techniques. Additionally, the theory proposed in [13] only considers the current and destination nodes while computing the routing algorithm. This is acceptable for wormhole switching because most routing algorithms using edge buffers can be defined in this way. However, routing algorithms using central queues usually require information about the queue containing the packet to be routed.

In this paper, we propose a necessary and sufficient condition for deadlock-free routing in virtual cut-through and store-and-forward switching. This theory is valid for networks using either central buffers or edge buffers. Also, the buffer containing the packet is considered by the routing algorithm. As we will see, the deadlock freedom properties of the whole network can be derived from the properties of a subset of buffers. Thus, the remaining buffers can be used without any restriction. This flexibility will allow us the definition of a design methodology that automatically supplies deadlock-free fully adaptive routing algorithms. Also, this methodology supports both, minimal and non-minimal routing algorithms. By contrary, the methodology proposed in [11] for wormhole switching requires a verification step. Usually, non-minimal routing algorithms do not pass the verification step. As in [11], this methodology is restricted to routing functions that only consider the current and destination nodes. As will be seen, it makes no sense for other definitions of the routing function.

We introduce the new theory and the new kinds of dependency in an informal way in Section 3. Section 4 proposes the new theory formally. Section 5 proposes a design methodology, giving some application examples in Section 6. Finally, some conclusions are drawn.

3 Informal Description

This section is organized as follows: First, we informally define the concept of dependency between resources. Then, we introduce the difference between deterministic and adaptive routing regarding deadlocks, defining routing subfunction informally. Later, we summarize the basic idea for our previous theory of store-and-forward routing. Then, we present the basic idea for the new theory, using an example. Moreover, we introduce the different kinds of resource dependency that will be defined in the next section. Finally, we briefly consider the existence of deadlocked configurations that cannot be reached.

Channels and buffers are the routing resources of an interconnection network. Deadlocks arise because packets hold resources while requesting other resources that will never be granted. While a packet is moving, resources are dynamically reserved and released. The critical resources are the ones held by blocked packets. When a packet is holding a resource, and it requests the use of another resource, there is a *dependency* between them. At a given node, a packet may request the use of several resources, then selecting one of them (adaptive routing). Every requested resource produces a dependency because it will be selected when the remaining resources are busy.

A network can use either edge buffers associated with channels or central queues. In the first case, a blocked packet occupies a channel and its associated buffer, producing dependencies between channels. In the second case, a blocked packet occupies a buffer in a central queue, producing queue dependencies. In both cases, a blocked packet is stored in one buffer. Thus, it is possible to use the same model to analyze both kinds of networks. So, unless explicitly stated, we will indistinctly refer to the dependencies between edge buffers or central queues as resource dependencies or simply dependencies. Also, we will indistinctly refer to edge buffers and central buffers as buffers. A set of one or more buffers with a FIFO policy will be referred to as a queue.

With deterministic routing, packets have a single routing option at each node. Thus, it is necessary to remove all the cyclic dependencies between resources to prevent deadlocks. Otherwise, packets may indefinitely hold some buffers while waiting for other buffers that are held by other packets. When adaptive routing is considered, packets usually have several choices at each node. Even if some of those routing choices are involved in cyclic dependencies, packets can use alternative resources to break deadlocks. Thus, it is not necessary to eliminate all the cyclic dependencies, provided that every packet can always find a path toward its destination whose buffers are not involved in cyclic dependencies.

For virtual cut-through and store-and-forward switching with edge buffers, it suffices with the existence of a connected channel subset C_1 whose channels are not involved in cyclic dependencies [10]. If cyclic dependencies are not allowed between channels belonging to C_1, then we guarantee that packets will not block indefinitely holding those channels. If the routing function is able to deliver all the packets using only channels belonging to C_1 (connected channel subset), deadlock freedom is guaranteed regardless of how packets are routed across the remaining channels. Similar considerations are also valid for networks using central queues. The channels belonging to C_1 will be referred to as *escape channels*, *escape paths* or, in general, *escape resources*. The routing function restricted to use only the escape resources will be referred to as *routing subfunction*. A routing subfunction is only a mathematical tool to restrict our attention to a subset of resources. Routing is not restricted at all. When a routing subfunction is considered, only the dependencies between the resources supplied by it are considered.

There are two important issues about the theory we proposed in [10] for store-and-forward switching. When a packet uses an escape resource at a given node, it can freely use any of the available resources supplied by the routing function at the next node. Also, when a packet is blocked because all the alternative resources are busy, it is not required to wait until a predetermined resource is available. Instead, it is repeatedly routed until any of the resources supplied by the routing function becomes free. Both issues are important, because they considerably increase routing flexibility, especially when the network is heavily loaded. These considerations also apply to the theory proposed in this paper.

In [10], a given resource was either supplied by the routing subfunction for all the destinations or not supplied at all. Labeling each resource as belonging or not belonging to the set of escape resources is simple, but the result is only a sufficient condition for deadlock-free adaptive routing. In this paper, we go one step further by conditionally labeling each resource as escape resource depending on the destination of the packet. By doing so, we achieve the maximum flexibility in the definition of the routing subfunction, but resource dependencies must be carefully analyzed, as illustrated in the following example.

Consider a unidirectional ring with four nodes denoted n_i, $i = \{0, 1, 2, 3\}$ and two channels connecting each pair of adjacent nodes, except nodes n_3 and n_0 that are linked by a single channel. Let c_{Ai}, $i = \{0, 1, 2, 3\}$ and c_{Hi}, $i = \{0, 1, 2\}$ be the outgoing channels from node n_i. The routing algorithm uses edge buffers and can be stated as follows: If the current node n_i is equal to the destination node n_j, deliver the packet. Otherwise, use either c_{Ai}, $\forall j \neq i$ or c_{Hi}, $\forall j > i$. In other words, c_{Ai} channels can be used to forward packets to all the destinations. However, c_{Hi} channels can only be used if the destination is higher than the current node. Fig. 1 shows the network.

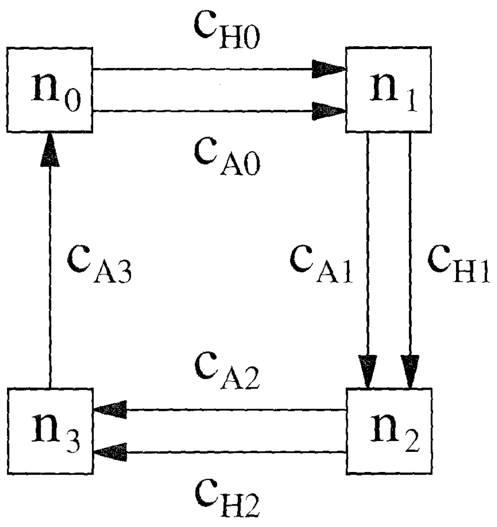


Fig. 1. Network for the example.

In this routing algorithm, we cannot easily identify a subset of channels that do not produce cyclic dependencies and allow packets to reach all the destinations. c_{Hi} channels alone do not allow packets to reach node n_0. c_{Ai} channels have cyclic dependencies between them. Thus, we define the escape paths by using part of the routing capabilities of c_{Ai} channels. More precisely, c_{Ai} channels are used as escape paths when the packet destination is lower than the node currently holding the packet. c_{Hi} channels are also used as escape paths. This definition allows packets to reach any destination because they will either use c_{Hi} or c_{Ai} channels depending on whether the destination node is higher or lower than the current node.

Restricting c_{Ai} channels to be used as escape paths only when the packet destination is lower than the current node, breaks the cyclic dependencies between c_{Ai} channels because c_{A0} is not an escape channel for any destination. Effectively, there is not any node lower than n_0.

When we focus on the resources supplied by the routing subfunction some dependencies are easy to detect. Suppose that a packet destined for node n_2 has reserved channel c_{H0}, and then it requests channel c_{H1} to reach node n_2. Both channels, c_{H0} and c_{H1}, are supplied by the routing subfunction for packets destined for node n_2. Thus, there is a dependency from channel c_{H0} to channel c_{H1}. We will refer to this kind of dependency as *direct* dependency.

However, some dependencies are difficult to detect. Suppose that a packet destined for node n_3 has reserved channel c_{A1}, and then it requests channel c_{H2} to reach node n_3. Obviously, there is a dependency from channel c_{A1} to channel c_{H2}. However, c_{A1} is not supplied by the routing subfunction for packets destined for node n_3. So, the question is: When we restrict our attention to the resources supplied by the routing subfunction, do we consider the dependency from c_{A1} to c_{H2}? The answer must be positive because c_{A1} is supplied by the routing subfunction for other destinations. Thus, it is an escape resource. If we do not consider this dependency, some packets requesting c_{A1} to escape from cycles may block forever, as we will see in Section 4.5 using an example. We will refer to this kind of dependency as *direct cross* dependency. It is important to note that both kinds of dependency are particular cases of the same definition. Although this view slightly differs from the one presented in [13] for wormhole switching, the concept is identical. We believe that the view presented in this paper is easier to understand.

Let us consider the opposite case. Suppose that a packet destined for node n_3 has reserved channel c_{H1}, and then it requests channel c_{A2} to reach node n_3. In this case, there is also a dependency from channel c_{H1} to channel c_{A2}. However, c_{A2} is not requested by the routing subfunction for packets destined for node n_3. In other words, c_{A2} is not required as an escape resource for packets destined for node n_3. Thus, we will not consider this dependency when the behavior of the routing subfunction is analyzed.

Finally, we consider whether a static analysis of the network is enough. A deadlocked configuration is an assignment of a set of packets to each queue such that no packet is able to advance. A configuration describes the state of an interconnection network at a given time. In some cases, a deadlocked configuration cannot be reached by routing packets starting from an empty network. This situation usually arises when two or more packets require the use of the same resource at the same time to reach the configuration. A configuration that can be reached by routing packets starting from an empty network is *reachable* or *routable* [6]. If all the configurations are reachable then a static analysis of the network is enough. Otherwise, the dynamic evolution of the network should be considered. This dynamic analysis is not required to prove that a routing function is deadlock-free. Reachability only affects necessary conditions for deadlock-free routing.

If the routing algorithm only considers the current and destination nodes, all the configurations are reachable, as will be seen later. However, if it also considers the buffer where the packet is stored then some configurations may be unreachable. As far as we know, nobody has proposed unreachable configurations for common topologies. Thus, in practice, we can consider that all the configurations are reachable, and a static analysis of the network is enough. Reachability remains as a theoretical open problem for some definitions of the routing function.

4 Formal Theory

This section develops the theoretical background for the design of deadlock-free adaptive routing algorithms for virtual cut-through and store-and-forward networks with central buffers or edge buffers. Without loss of generality, we will mainly focus on virtual cut-through to avoid mixing different terminology and assumptions. However, the theoretical results are valid for both flow control techniques.

4.1 Router Model

Before proposing the theory, we present two simple router models, highlighting the aspects that may affect deadlock avoidance. More precise assumptions are given in the next subsection.

Fig. 2 shows the router model for networks with edge buffers. Each router consists of a switch, an address decoder that implements the routing algorithm, and several channels. The local processor sends and receives packets through the channels connecting it to the switch.

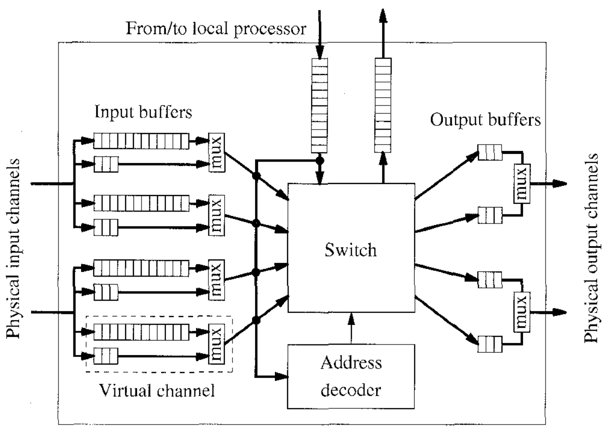


Fig. 2. Router model.

The switch allows multiple packets to traverse a node simultaneously without interference.

The address decoder configures the switch, determining the output channel for each packet as a function of the destination node, the current node and/or the current queue, and the output channel status. The address decoder can only process one packet header at a time. If there is contention for the address decoder, access is round robin. When a packet gets the address decoder, but cannot be routed because all the alternative output channels are busy, it must wait in the corresponding input queue until its next turn. By doing so, the packet will get the first channel that becomes free, thus increasing routing flexibility. Configurations with as many address decoders as input channels are also supported, provided that they are linked by an arbiter with round robin policy.

Physical channels are bidirectional full-duplex. Physical channels may be split into virtual channels. Virtual channels are assigned the physical channel cyclically, only if they are ready to transfer a flit (demand-slotted round robin). Each virtual channel has two buffers at the input side. A large buffer with capacity for one or more packets is required to store packets and remove them from the network whenever no output channel is available. A channel will only accept a new packet if there is enough buffer space to store the whole packet. Also, a small buffer with capacity for a few flits is used to speed up packet pipelining when the packet is not blocked. This small buffer does not affect deadlock avoidance, because blocked packets are always moved to the large buffer. So, in what follows, we will not consider the small buffers. Similarly, small output buffers may be used to increase throughput when virtual channels are used but those buffers will not be considered for deadlock avoidance.

If the network uses central queues instead of edge buffers, the router model is similar to the one described above. In this case, a few central queues deep enough to store one or more packets are used. As above, a channel will only accept a new packet if there is enough buffer space to store the whole packet. Buffer space must be reserved before starting packet transmission, thus preventing other channels from reserving the same buffer space. An arbiter is needed to handle simultaneous requests. As above, a small buffer associated with each channel may be used to speed up packet pipelining when the packet is not blocked. Small buffers do not affect deadlock avoidance and they will not be considered.

4.2 Assumptions

The basic assumptions are very similar to the ones proposed in [11], modified for virtual cut-through. Messages may be split into packets. As packets carry information about their destination, the theory is valid for both, messages and packets. Without loss of generality, we will only refer to packets. The assumptions are the following:

1. A node can generate messages of arbitrary length destined for any other node at any rate. Messages may be split into packets.
2) A packet arriving at its destination node is eventually consumed.
3) Virtual cut-through or store-and-forward switching is used.
4) Each channel has a single queue associated with it, allowing the storage of a finite number of packets. Alternatively, each node has a few central queues with capacity for a finite number of packets. A channel will only accept a new packet if there is enough buffer space in the corresponding queue to store the whole packet. Buffer space must be reserved before starting packet transmission.
5) The address decoder may arbitrate between packets requesting it, but may not choose among waiting packets.
6) The route taken by a packet depends on its destination, the current node and the status of the requested channels or queues (free or busy). The queue containing the packet can also be considered. However, some theoretical results are only valid for routing algorithms that do not consider the current queue. At a given node, an adaptive routing function supplies a set of channels or queues. A selection from this set is made based on the status of those resources. This selection is performed in such a way that a free resource (if any) is supplied.
7) When several packets are waiting because all the resources supplied by the routing function are busy, they are routed following a round-robin strategy, thus preventing starvation.

8) The routing function may allow packets to follow non-minimal paths.
9) All the configurations can be reached by routing packets starting from an empty network.

4.3 Definitions

Before proposing the theorems, some definitions are needed. Most differences between these definitions and the ones proposed in [11] for wormhole switching arise because the routing function used in this paper also considers the queue containing the packet. A similar extension can be done for the theories proposed in [11], [13], [14]. Additionally, definitions and theorems consider both, central and edge queues.

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. The source and destination nodes of channel c_i are denoted s_i and d_i, respectively. Each node has one injection queue and one delivery queue. The sets of injection and delivery queues are denoted Q_I and Q_D, respectively. Let Q_N be the set of standard queues, which includes all the queues in the network except injection and delivery queues. Also, $Q_{IN} = Q_I \cup Q_N$, $Q_{ND} = Q_N \cup Q_D$, and $Q = Q_I \cup Q_N \cup Q_D$. Each queue $q_i \in Q$ has capacity* cap(q_i). *A queue $q_i \in Q_N$ can be a central queue at node n_i if central buffers are used or it can be associated with a channel c_i if edge buffers are used. In this case, we will consider that q_i is at node d_i. For the sake of simplicity, an enumeration of arbitrary queues is denoted $q_1, q_2, \ldots, q_k$ instead of $q_{i_1}, q_{i_2}, \ldots, q_{i_k}$. An enumeration does not imply any queue ordering.*

DEFINITION 2. *Let F be the set of valid* queue status, *$F = \{full, non\text{-}full\}$.*

DEFINITION 3. *An* adaptive routing function *R: $Q_{IN} \times N \rightarrow \mathcal{P}(Q_{ND})$, where $\mathcal{P}(Q_{ND})$ is the power set of Q_{ND}, supplies a set of alternative edge or central queues to send a packet from the current queue q_c to the destination node n_d, $R(q_c, n_d) = \{q_1, q_2, \ldots, q_p\}$. In general, p will be less than the number of queues that can be reached from the current queue to restrict routing and obtain deadlock-free algorithms. As a particular case, $p = 1\ \forall\ (q_c, n_d) \in Q_{IN} \times N$, defines a deterministic routing function. If the current queue q_c is at destination node n_d then the routing function can only supply a delivery queue at current node. Defining the domain of the routing function as $Q_{IN} \times N$ is more general than considering only the current and destination nodes. Thus, all the theoretical results for routing functions defined on $Q_{IN} \times N$ are also valid for routing functions defined on $N \times N$. Some specific results for these functions will be presented in Section* 4.5.

DEFINITION 4. *A* selection function *S: $\mathcal{P}(Q_{ND} \times F) \rightarrow Q_{ND}$ selects a non-full queue (if any) from the set supplied by the routing function. From the definition, S takes into account the status of all the queues belonging to the set supplied by the routing function. The selection can be random or based on static or dynamic priorities. It must be noticed that starvation is prevented using a round-robin strategy when several packets are waiting for the router, according to Assumption 7. The selection function will only affect performance.*

DEFINITION 5. *A* configuration *is an assignment of a set of packets to each queue. The number of packets in the queue q_i is denoted* size(q_i). *The destination node for a packet p_j will be denoted* dest(p_j). *If the first packet in the queue q_i* is *destined for node n_d, then* head(q_i) = n_d. *A configuration is* legal *iff*

$$\forall\ q_i \in Q,\ size(q_i) \leq cap(q_i) \text{ and}$$

$$\forall\ q_i \in Q_{ND}, \forall p_j \in q_i, \exists\ q_0 \in Q_I, \exists q_1, q_2, \ldots, q_{i-1} \in Q_N$$

$$\text{such that } q_m \in R(q_{m-1}, dest(p_j)),\ m = 1, \ldots, i$$

For each queue, the capacity is not exceeded and all the packets stored in the queue (if any) have been routed from some injection queue using the routing function.

DEFINITION 6. *A* deadlocked configuration *for a given interconnection network I and routing function R is a nonempty legal configuration verifying the following condition*:

$$\forall q_i \in Q_{IN} \text{ such that } size(q_j) > 0 \begin{cases} head(q_i) \neq n_i \\ size(q_i) = cap(q_j)\ \ \forall q_j \in R(q_i, head(q_i)) \end{cases}$$

In a deadlocked configuration no packet has already arrived at its destination node. Packets cannot advance because all the alternative queues supplied by the routing function are full. No condition is imposed on empty queues.

DEFINITION 7. *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.*

DEFINITION 8. *A routing function R for a given interconnection network I is* connected *iff for any legal configuration*

$$\forall\ q_i \in Q_{IN} \text{ such that } size(q_i) > 0, \exists q_{i+1}, \ldots, q_{i+k-1} \in Q_N,$$

$$\exists q_{i+k} \in Q_D \text{ such that}$$

$$q_m \in R(q_{m-1}, head(q_i)),\ m = i + 1, \ldots, i + k$$

In other words, it is always possible to establish a path for every packet from its current queue to its destination node (delivery queue). Notice that the configuration must be legal. Otherwise, the routing function may not supply any queue.

As will be seen, it is possible to prove deadlock freedom by focusing on the behavior of a restricted routing function. This concept is formalized by the following definition:

DEFINITION 9. *A* routing subfunction *R_1 for a given routing function R is a routing function defined on the same domain as R that supplies a subset of the queues supplied by R*

$$R_1(q_i, x) \subseteq R(q_i, x) \ \forall\, q_i \in Q_{IN}, \forall\, x \in N \qquad (1)$$

The set of all the queues supplied by R_1 is

$$Q_{ND1} = \bigcup_{\forall q_i \in Q_{IN},\ \forall x \in N} R_1(q_i, x)$$

As a particular case, given a queue subset $Q_{ND1} \subseteq Q_{ND}$ one can always construct a routing subfunction R_1 by applying the following relationship:

$$R_1(q_i, x) = R(q_i, x) \cap Q_{ND1} \ \forall q_i \in Q_{IN}, \forall x \in N \qquad (2)$$

Expression (1) is more general than expression (2) because it allows us to remove a queue from R_1 only for some destinations.

DEFINITION 10. *Given an interconnection network I, a routing function R and a pair of queues $q_i, q_j \in Q$, there is a* direct dependency *from q_i to q_j iff there exists a legal configuration with packets stored in q_i and*

$$q_j \in R(q_i, head(q_i))$$

That is, q_j can be requested by packets stored in q_i for some legal configuration. If routing subfunctions are not considered, this is the only kind of dependency between resources.

When a routing subfunction is considered, we only consider the queues supplied by it and the dependencies between them, as indicated in the next definition.

DEFINITION 11. *Given an interconnection network I, a routing function R, a routing subfunction R_1 and a pair of queues $q_i, q_j \in Q_{ND1}$, there is a* dependency *from q_i to q_j iff there exists a legal configuration with packets stored in q_i and*

$$q_j \in R_1(q_i, head(q_i))$$

This definition is identical to definition 10 except that it is restricted to queues supplied by R_1. It considers all the possible dependencies between resources supplied by a routing subfunction. However, the definition of routing subfunction produces two particular cases with subtle differences between them. As the configuration with packets stored in q_i is legal then

$$\exists q_k \in Q_{IN} \text{ such that } q_i \in R(q_k, head(q_i))$$

If $q_i \in R_1(q_k, head(q_i))$ for some legal configuration then the dependency is referred to as direct dependency. *If for all the legal configurations $q_i \notin R_1(q_k, head(q_i))$ then the dependency is referred to as* direct cross dependency. *Notice that $q_i \in Q_{ND1}$. Thus, q_i is supplied by R_1 for some other destinations.*

Direct and direct cross dependencies are particular cases of the same definition. The difference between them can be informally viewed as follows: If the packet stored in q_i has been routed there by using queues supplied by R_1 (the configuration is legal for R_1), the dependency is referred to as direct dependency. If the packet stored in q_i cannot be routed there by using queues supplied by R_1 (the configuration is legal for R but not for R_1), the dependency is referred to as direct cross dependency. This view complements the one presented in Section 3. When a routing subfunction is defined by using expression (2) there is not any direct cross dependency because this kind of dependency requires that a queue is supplied by the routing subfunction only for some destinations.

Direct dependencies are the only ones that would exist if R_1 were a routing function instead of being a routing subfunction of R. Thus, these dependencies are identical to the ones proposed in Definition 10 for routing functions. So, we use the same name.

The additional kinds of dependency defined in [13], [14] only exist in wormhole networks. Resource dependencies can be analyzed by using a graph to represent them. We define two graphs.

DEFINITION 12. *A* resource dependency graph *D for a given interconnection network I and routing function R, is a directed graph, D = G(Q, E). The vertices of D are the queues of I. The arcs of D are the pairs of queues (q_i, q_j) such that there is a direct dependency from q_i to q_j.*

DEFINITION 13. *An* extended resource dependency graph D_E *for a given interconnection network I and routing subfunction R_1 of a routing function R, is a directed graph, $D_E = G(Q_{ND1}, E_E)$. The vertices of D_E are the queues supplied by the routing subfunction R_1 for some destinations. The arcs of D_E are the pairs of queues (q_i, q_j) such that there is either a direct or direct cross dependency from q_i* to *q_j. It must be noticed that the extended resource dependency graph has been redefined with respect to both,* [11] *and* [13], [14].

4.4 Necessary and Sufficient Condition

Two theorems are proposed. The first one simply states that techniques based on directed acyclic graphs can also be applied to adaptive routing functions. The second theorem allows us the design of adaptive routing functions with cyclic dependencies in their resource dependency graph. It supplies a necessary and sufficient condition for deadlock avoidance. 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 resource dependency graph D.*

PROOF SKETCH. As the resource dependency graph for R is acyclic, it is possible to establish an order between the queues of Q. As R is connected, the minimal queues in that order are also delivery queues. Suppose that there is a deadlocked configuration for R. Let $q_i \in Q$ be a full queue such that there is not any full queue less than q_i. If q_i is a delivery queue, then the packet at the queue head has reached its destination and there is no deadlock. Otherwise, using the queues less than q_i, the packet at the queue head of q_i can advance and there is not any deadlock. □

PROOF. Suppose that there are no cycles in D. Then, one can assign an order to the queues of Q so that if $(q_i, q_j) \in E$ then $q_i > q_j$. Consider the queue(s) q_i such that

$$\forall q_j \in Q, (q_i, q_j) \notin E$$

Such a queue q_i is minimal in the order. Let us prove that it is a delivery queue. If it were not a delivery queue, as the routing function is connected, for any legal configuration such that $size(q_i) > 0$

$$\exists q_k \in Q_{ND} \text{ such that } q_k \in R(q_i, head(q_i))$$

As the configuration is legal then $(q_i, q_k) \in E$, contrary to the assumption that q_i is minimal. Thus, q_i is a delivery queue.

Suppose that there is a deadlocked configuration for R. Let $q_i \in Q$ be a full queue such that there is not any full queue less than q_i. If q_i is minimal, it is also a delivery queue and the packet at the head of q_i is not blocked. If q_i is not minimal then

$$size(q_j) < cap(q_j) \; \forall q_j \in Q \text{ such that } q_i > q_j$$

Thus, the packet at the head of queue q_i is not blocked, and there is no deadlock. □

THEOREM 2. *A connected and adaptive routing function R for an interconnection network I is deadlock-free iff there exists a routing subfunction R_1 that is connected and has no cycles in its extended resource dependency graph D_E.*

PROOF SKETCH. $\Leftarrow$ The case $R_1 = R$ is trivial. Otherwise, R_1 will supply a subset of the queues supplied by R. As the extended resource dependency graph for R_1 is acyclic, it is possible to establish an order between the queues of Q_{ND1} (queues supplied by R_1 for some destination). As R_1 is connected, the minimal queues in that order are also delivery queues. Suppose that there is a deadlocked configuration for R. There are two possible cases:

a) No queue belonging to Q_{ND1} is full. As R_1 is connected, packets stored at queue heads can be routed using queues belonging to Q_{ND1} and there is no deadlock.

b) Some queues belonging to Q_{ND1} are full. Let $q_i \in Q_{ND1}$ be a full queue such that there is not any full queue less than q_i. Again, there are two possible cases:

b1) If q_i is minimal (delivery queue) then the packet at the queue head is not blocked and there is no deadlock.

b2) If q_i is not minimal, no queue of Q_{ND1} less than q_i will have a full queue. Thus, the packet at the queue head of q_i can be routed because R_1 is connected and there is no deadlock.

$\Rightarrow$ Suppose that R is deadlock-free. Thus, there is not any deadlocked configuration. We have to construct a connected routing subfunction R_1 without cycles in its extended resource dependency graph D_E.

The basic idea for the proof is the following: First, we define a connected routing subfunction R_1 that uses as few queues belonging to cycles of D as possible. Using queues belonging to cycles of D does not imply that the resulting routing subfunction will have cyclic dependencies between queues, provided that it only uses a subset of the queues belonging to each cycle of D. Then, we proceed by contradiction. We assume that there is a cycle in the extended resource dependency graph for R_1, showing that it is possible to redefine R_1 in such a way that the cycle is broken.

Let us describe this idea in more detail. The definition of R_1 is iterative. We start by including in R_1 all the queues that do not belong to any cycle of D. If R_1 is not connected yet, we iteratively add queues to R_1 until it is connected. In each iteration, we consider a single queue q_j belonging to a cycle of D. q_j is added to R_1 to route packets from a queue q_i to a node x if the previous definition for R_1 did not offer any path from q_i to x and q_j is supplied by R to route packets from q_i to x following a shortest (possibly minimal) path.

Now, let us assume that there is a cycle in the extended resource dependency graph for R_1. We fill the queues of the cycle trying to build a deadlocked configuration, in such a way that the next queue requested by each packet is supplied by R_1 (see Fig. 3). The resulting configuration is legal because the definition of dependency requires legal configurations. As R is deadlock-free, it will offer an alternative queue q_k for the packet stored in some queue q_i. Thus, we redefine R_1 by eliminating q_j and by adding q_k for the packet destination. By repeating this process for all the legal packet destinations it is possible to break the dependency from q_i to q_j, contrary to the assumption that there is a cycle in the extended resource dependency graph for R_1. □

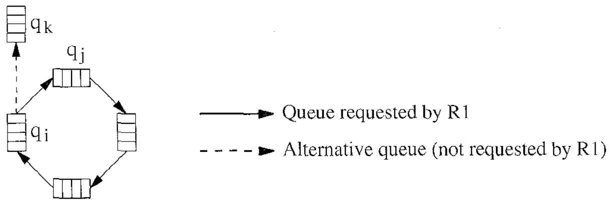


Fig.3. Filling the cycle.

PROOF. $\Leftarrow$ Suppose that there exists a routing subfunction R_1 of the routing function R and that R_1 is connected and there are no cycles in D_E. If $R_1(q, x) = R(q, x) \; \forall q \in Q_{IN}, \forall x \in N$, then $D_E = D$. Thus, there is not any cycle in D and R is deadlock-free by Theorem 1. Otherwise, $R_1(q, x) \subseteq R(q, x) \; \forall q \in Q_{IN}, \forall x \in N$. Let $Q_{ND1} = \bigcup_{\forall q \in Q_{IN}, \forall x \in N} R_1(q, x)$. As there are no cycles in D_E, one can assign an order to the queues of Q_{ND1} so that if $(q_i, q_j) \in E_E$ then $q_i > q_j$. Consider the queue(s) $q_i \in Q_{ND1}$ such that

$$\forall q_j \in Q_{ND1}, (q_i, q_j) \notin E_E$$

Such a queue q_i is minimal in the order. Let us prove that it is a delivery queue. If it were not a delivery queue, as the routing subfunction R_1 is connected, for any legal configuration such that $size(q_i) > 0$

$$\exists q_k \in Q_{ND1} \text{ such that } q_k \in R_1(q_i, head(q_i))$$

As the configuration is legal then $(q_i, q_k) \in E_E$, contrary to the assumption that q_i is minimal. Thus, q_i is a delivery queue.

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

a) No queue belonging to Q_{ND1} is full. Let $q_i \in Q_{IN}$ be a nonempty queue. As R_1 is connected then

$$\exists q_j \in Q_{ND1} \text{ such that } q_j \in R_1(q_i, head(q_i)) \Rightarrow$$
$$\exists q_j \in Q_{ND} \text{ such that } q_j \in R(q_i, head(q_i))$$

Also $size(q_j) < cap(q_j)$ and R does not have a deadlock.

b) Some queues belonging to Q_{ND1} are full. Let $q_i \in Q_{ND1}$ be a full queue such that there is not any full queue less than q_i. Again, there are two possible cases:

b1) q_i is minimal. As shown above, it is also a delivery queue and thus, the packet at the head of the queue is not blocked.

b2) q_i is not minimal. Thus

$$size(q_j) < cap(q_j) \; \forall q_j \in Q_{ND1} \text{ such that } q_i > q_j$$

Taking into account that R_1 is connected

$$\exists q_j \in Q_{ND1} \text{ such that } q_j \in R_1(q_i, head(q_i)) \Rightarrow$$
$$\exists q_j \in Q_{ND} \text{ such that } q_j \in R(q_i, head(q_i))$$

As every deadlocked configuration is legal, then

$$q_i > q_j \Rightarrow size(q_j) < cap(q_j)$$

and R does not have any deadlock.

$\Rightarrow$ Suppose that R is deadlock-free. Thus, there is not any deadlocked configuration. Now, we have to construct a routing subfunction, showing that it is connected and it has no cycles in its extended resource dependency graph D_E. There are two possible cases:

a) If the resource dependency graph D for R is acyclic, then we define

$$R_1(q, x) = R(q, x) \; \forall q \in Q_{IN}, \forall x \in N$$

Obviously, R_1 is connected and it has no cycles in D_E.

b) If D is not acyclic, let $Q_a \subset Q$ be the subset of queues that do not belong to any cycle in D. We define

$$R_1(q, x) = R(q, x) \cap Q_a \; \forall q \in Q_{IN}, \forall x \in N$$

Now, there are two possible cases:

b.1) If R_1 is connected, it is obvious that it has no cycles in D_E.

b.2) If R_1 is not connected, we start from the current definition for R_1, adding queues to it from $Q_{ND} - Q_a$ until R_1 is connected. This is an iterative process. Let $SP(q, x)$ be the set of queues supplied by $R(q, x)$ belonging to a shortest path from q to x. A shortest path will only be minimal if R supplies at least one minimal path from q to x. In each iteration, a single queue $q_j \in Q_{ND} - Q_a$ is considered, redefining R_1 as follows:

$$\forall q_i \in Q_{IN}, \forall x \in N,$$
$$R_1^{(p+1)}(q_i, x) = \begin{cases} R_1^{(p)}(q_i, x) & \text{if } R_1^{(p)}(q_i, x) \neq \phi \\ & \quad \text{or } q_j \notin SP(q_i, x) \\ \{q_j\} & \text{if } R_1^{(p)}(q_i, x) = \phi \\ & \quad \text{and } q_j \in SP(q_i, x) \end{cases}$$

where $R_1^{(p)}$ is the routing subfunction R_1 in the iteration p. In other words, q_j is added to R_1 to route packets from q_i to x if the previous definition for R_1 did not offer any path from q_i to x and q_j is supplied by R to route packets from q_i to x following a shortest path. Otherwise, R_1 remains unaltered. The iterative process will finish because R is connected. Thus, after the iterative process, R_1 is also connected. Let Q_{ND1} be the set of queues supplied by R_1 for some destination.

Now, we have to prove that there are no cycles in the extended resource dependency graph D_E for R_1, possibly redefining it again. Restricting routing cannot produce new cyclic dependencies. Thus, only a cycle in the resource dependency graph D for R may induce a cycle in D_E.

We proceed by contradiction. We assume that R_1 has a cycle in its extended resource dependency graph, showing that it is possible to redefine it so that the new routing subfunction is connected and it has no cycles in its extended resource dependency graph. Suppose that there is a cycle in D_E. Let $Q_c = \{q_1, q_2, \ldots, q_k\}$ be the subset of queues that form the cycle in D_E. Without loss of generality, q_{i+1} will always denote a queue requested by packets stored in q_i. Let us consider the dependency from q_1 to q_2. We fill the queue q_1 with packets destined for a legal destination requiring the use of q_2. We proceed in the same way with the remaining dependencies. Let Q_d be the set formed by all the queues $q_i \in Q_{ND1}$ such that

$$\exists\, q_j \in Q_c, \exists q_{i+1}, \ldots, q_{j-1} \in Q_{ND1} \text{ such that}$$
$$(q_m, q_{m+1}) \in E_E, m = i, \ldots, j-1$$

There is a sequence of dependencies (a path in D_E) from every queue in Q_d to some queue in Q_c. We also fill every queue $q_i \in Q_d$ with packets destined for a legal destination requiring the use of the next queue in the sequence of dependencies.

We have filled the queues in $Q_c \cup Q_d$ in such a way that

$$size(q_m) = cap(q_m) \text{ and } head(q_m) \neq n_m \ \forall q_m \in Q_c \cup Q_d$$

As R is deadlock-free, all the packets in the above described configuration must be able to advance. If R only supplies paths for packets stored in queues belonging to Q_d, the packets in the cycle (stored in queues belonging to Q_c) will never advance. Thus, R must supply paths for packets stored in queues belonging to Q_c

$$\exists q_i \in Q_c, \exists q_j \in Q_{ND} \text{ such that}$$
$$q_j \in R(q_i, head(q_i)) \text{ and } size(q_j) < cap(q_j)$$

Obviously, $q_j \notin Q_c \cup Q_d$ because all these queues are full. Taking into account the definition for R_1

$$q_j \notin R_1(q_i, head(q_i))$$

Otherwise, R_1 would offer two different paths to reach $head(q_i)$. Additionally, there is no dependency from q_j to any other queue belonging to $Q_c \cup Q_d$. Otherwise, $q_j \in Q_d$ and it would be full. We can redefine R_1 by including q_j and eliminating q_{i+1} to route packets from q_i toward $head(q_i)$. Obviously, R_1 is still connected. According to the new definition for R_1

$$q_{i+1} \notin R_1(q_i, head(q_i))$$

thus breaking the dependency from q_i to q_{i+1} when the packet destination is $head(q_i)$. Similarly, we can fill the cycle with other configurations identical to the one described above, except that the packet stored in q_i is now destined for other nodes. Once we have tried all the legal destinations that require crossing q_{i+1}, we have broken the dependency from q_i to q_{i+1}, contrary to the assumption that there was a cyclic dependency in D_E. Thus, the extended resource dependency graph D_E for R_1 is acyclic and the theorem holds. □

COROLLARY 1. *A connected and adaptive routing function R for an interconnection network I is deadlock-free if there exists a subset of queues $Q_{ND1} \subseteq Q_{ND}$ such that the routing subfunction $R_1(q, x) = R(q, x) \cap Q_{ND1}$ $\forall q \in Q_{IN}$, $\forall x \in N$, is connected and has no cycles in its resource dependency graph D_1.*

PROOF. Taking into account the definition of R_1, there is no direct cross dependency between the queues belonging to Q_{ND1}. Thus, the resource dependency graph for R_1 is identical to the extended resource dependency graph. By Theorem 2, R is deadlock-free. □

There are some interesting considerations:

1. Theorem 2 is identical to the necessary and sufficient condition proposed in [13], [14] for wormhole switching, except that the routing function is not required to supply at least one minimal path and be coherent. A routing function R is coherent if for every path P that can be established by R, all subpaths of P are also paths of R. Coherence is required in wormhole switching because a blocked packet may occupy several channels. Thus, a packet may visit a node twice and request a channel it is still occupying. Also, it must be noticed that the extended resource dependency graph defined in this paper only contains direct and direct cross dependencies.

2. If some legal configurations are unreachable then assumption 9 does not hold, and Definition 7 becomes only a sufficient condition. As a consequence, Theorem 2 also becomes a sufficient condition. However, if the routing function only considers the current and destination nodes to compute the path then every legal configuration is also reachable. Effectively, as the routing function has no memory of the path followed by each packet, we can consider that a packet stored in a channel queue was generated by the source node of that channel. Thus, there is not any other packet requiring the use of that channel at the same time to reach the configuration. Similar considerations are applicable to routing functions using central queues.

3. Corollary 1 gives a very flexible way to define deadlock-free adaptive routing functions. If a routing function satisfies the conditions proposed by it, it does not matter how packets are routed through queues not belonging to Q_{ND1}. Those packets are allowed to use all the minimal and non-minimal paths. However, when non-minimal paths are used, livelock-freedom should be guaranteed. This result is much more flexible than the one previously obtained for wormhole switching [13], [14]. The reason being that packets occupy a single buffer when they are blocked. As a consequence, dependencies only exist between queues in the same node or in adjacent nodes.

4. The routing subfunction R_1 can be adaptive.

4.5 Routing Functions Defined on $N \times N$

In this section we consider routing functions defined as follows:

DEFINITION 14. *An* adaptive routing function R: $N \times N \rightarrow \mathcal{P}(Q_{ND})$, *supplies a set of alternative edge or central queues to send a packet from the current node n_c to the destination node n_d, $R(n_c, n_d) = \{q_1, q_2, \ldots, q_p\}$.*

Routing functions defined on $N \times N$ have no memory of the path followed by the packet. The remaining definitions proposed in Section 4.3 can be modified accordingly. In particular, legal configurations and routing subfunctions are defined as follows:

DEFINITION 15. *A configuration is* legal *iff*

$$\forall q_i \in Q,\ size(q_i) \le cap(q_i) \text{ and}$$
$$\forall q_i \in Q_{ND},\ \forall p_j \in q_i,\ \exists n_k \in N$$
$$\text{such that } q_i \in R(n_k, dest(p_j))$$

DEFINITION 16. *A routing subfunction R_1 for a given routing function R is a routing function defined on the same domain as R that supplies a subset of the queues supplied by R*

$$R_1(x, y) \subseteq R(x, y)\ \forall x, y \in N \tag{3}$$

The following theorem is specific for routing functions defined on $N \times N$. This theorem will allow us the definition of a very flexible design methodology for adaptive routing algorithms.

THEOREM 3. *A connected and adaptive routing function R for an interconnection network I is deadlock-free if there exists a subset of queues $Q_{ND1} \subseteq Q_{ND}$ such that the routing subfunction $R_1(x, y) = R(x, y) \cap Q_{ND1}\ \forall x, y \in N$, is connected and deadlock-free.*

PROOF. Suppose that there exists a queue subset $Q_{ND1} \subseteq Q_{ND}$ that defines a routing subfunction R_1 that is connected and deadlock-free. If $Q_{ND1} = Q_{ND}$ the proof is trivial. Otherwise $Q_{ND1} \subset Q_{ND}$.

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

a) No queue belonging to Q_{ND1} is full. Let $q_i \in Q_{IN}$ be a nonempty queue. As R_1 is connected then

$$\exists q_j \in Q_{ND1} \text{ such that } q_j \in R_1(n_i, head(q_i)) \Rightarrow$$
$$\exists q_j \in Q_{ND} \text{ such that } q_j \in R(n_i, head(q_i))$$

Also $size(q_j) < cap(q_j)$ and R do not have a deadlock.

b) Some queues belonging to Q_{ND1} are full. As there is a deadlocked configuration for R

$$\forall q_i \in Q_{IN} \text{ such that } size(q_i) > 0 \begin{cases} head(q_i) \ne n_i \\ size(q_j) = cap(q_j)\ \forall q_j \in R(n_i, head(q_i)) \end{cases}$$

A queue belonging to Q_{ND1} is supplied by R_1 for the same destinations as it is supplied by R. Thus, if we restrict our attention to queues belonging to Q_{ND1}, the configuration is also legal for R_1. Let $Q_{N1} = Q_{ND1} \cap Q_N$. Taking into account that $Q_{N1} \subset Q_{IN}$ and $R_1(x, y) = R(x, y) \cap Q_{ND1}\ \forall x, y \in N$

$$\forall q_i \in Q_{N1} \text{ such that } size(q_i) > 0 \begin{cases} head(q_i) \ne n_i \\ size(q_j) = cap(q_j)\ \forall q_j \in R_1(n_i, head(q_i)) \end{cases}$$

Thus, considering the queues belonging to Q_{N1}, there is a deadlocked configuration for R_1, contrary to the initial assumption. Thus, R must be deadlock-free. □

If the routing function were defined on $Q_{IN} \times N$ then it would not be possible to guarantee that a legal configuration for R is also legal for R_1. The definition of legal configuration given by definition 5 requires the existence of a sequence of queues $q_1, q_2, \ldots, q_{i-1} \in Q_N$ supplied by the routing function. If some of those queues are supplied by R but not for R_1, then the configuration is legal for R but not for R_1.

Also, if we used the definition of routing subfunction given by expression (3) for theorem 3, the resulting theorem would not be valid. Again, the reason is that a legal configuration for R is not necessarily a legal configuration for every routing subfunction defined as $R_1(x, y) \subseteq R(x, y)\ \forall x, y \in N$. The following example explains this issue more clearly.

Consider a 2D-mesh with edge buffers and a single physical channel connecting each pair of nodes. The routing function R is defined on $N \times N$. It forwards packets following any minimal path. This routing function is not deadlock-free. Fig. 4 shows a deadlocked configuration. Dotted incomplete boxes represent nodes of a 3×3 mesh. Dashed boxes represent switches. Solid boxes represent packet buffers (queues). The number inside each buffer indicates the packet destination. Solid arrows indicate the channel requested by the packet at the queue head. As packets are allowed to follow all the minimal paths, there are cyclic dependencies between channels. Additionally, there is no alternative path for the packets in the figure because packets are only allowed to follow minimal paths.

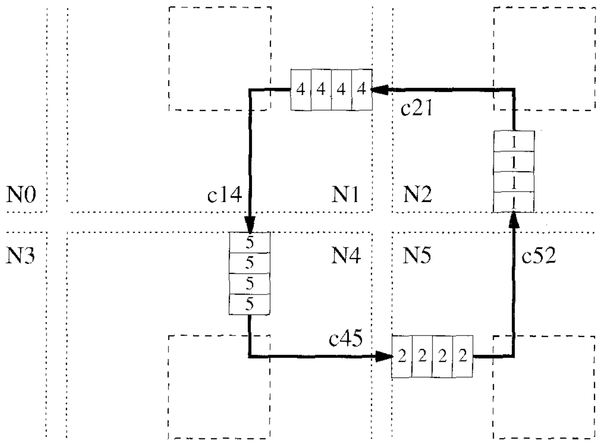


Fig. 4 Deadlocked configuration for *R*.

Consider a routing function R_1 that uses dimension-order routing (XY-routing). Obviously, $R_1(x, y) \subseteq R(x, y)\ \forall x, y \in N$. Thus, R_1 is a routing subfunction of R. It is well known that R_1 is connected and deadlock-free. However, R is not deadlock-free. Theorem 3 cannot be applied because there is no queue subset $Q_{ND1} \subseteq Q_{ND}$ such that $R_1(x, y) = R(x, y) \cap Q_{ND1}\ \forall x, y \in N$. According to this definition of routing subfunction, channels must be used in the same way by R and R_1 if they belong to Q_{ND1}. However, Y channels can be used by R to forward packets whose destination node is not in the same column as the current node, and R_1 can only use Y channels to forward packets whose destination node is in the same column as the current node.

This example also shows that the existence of an oblivious deadlock-free routing subfunction according to expression (3) does not imply that the routing function is deadlock-free. Thus, the necessary condition for deadlock-free adaptive routing proposed in [6] is not a sufficient one.

The above described example does not invalidate Theorem 2. Fig. 5 shows the extended resource dependency graph for R_1 on a 3×3 mesh. Black circles represent channels. The channel from node i to node j is denoted cij. Solid arrows represent direct dependencies. Dashed arrows represent direct cross dependencies. The graph contains cycles. It can be seen that there are direct cross dependencies from Y channels to X channels that close the cycles.

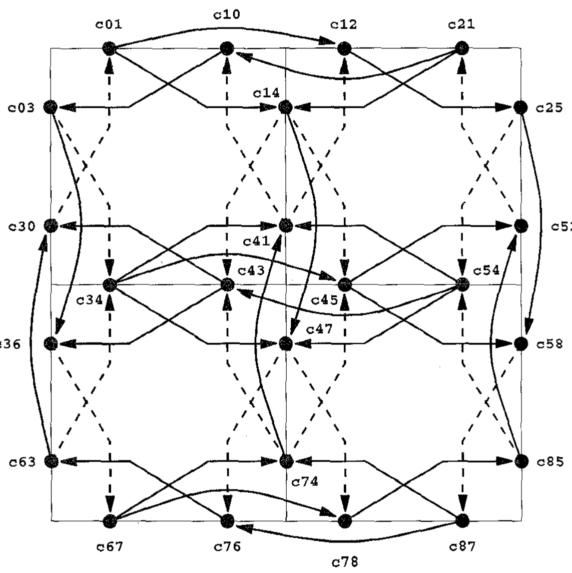


Fig. 5. Extended resource dependency graph for R_1.

5 Design Methodology

In this section we propose a methodology for the design of deadlock-free fully adaptive routing algorithms. It is restricted to routing functions defined on $N \times N$. As we will see, this methodology makes no sense for routing functions that consider the current queue. It starts from a previously proposed deterministic or partially adaptive routing algorithm. This methodology indicates a way to add channels or central queues to an existing network, also deriving the new routing function from the old one. It is similar to the first methodology proposed in [11] for wormhole switching, the main difference being that the methodology we propose here automatically supplies deadlock-free routing algorithms.

Methodology 1. *This methodology supplies fully adaptive minimal and non-minimal routing algorithms, starting from a previously proposed routing algorithm. The steps are the following:*

1. *Given an interconnection network I_1, define a connected deadlock-free routing function R_1 for it. Let Q_1 be the set of queues at this point.*

2. *Split each physical channel into a set of (additional) virtual channels or add a set of central queues. Let Q be the set of all the queues in the network. Let Q_{xy} be the set of queues belonging to a path (minimal or not) from node x to node y that can be requested at node x. Define the new routing function R as follows:*

$$R(x, y) = R_1(x, y) \cup (Q_{xy} \cap (Q - Q_1)) \ \forall x, y \in N$$

That is, the new routing function can use any of the new queues or, alternatively, the queues supplied by R_1. The selection function can be defined in any way. However, it is recommended to give a higher priority to the new queues belonging to minimal paths.

Step 1 establishes the starting point. We can use either a deterministic or a partially adaptive routing function as the basic one.

Step 2 indicates how to add edge or central queues to the network and how to define a new fully adaptive routing function from the basic one.

Only one additional central queue per node or virtual channel per physical channel are required for fully adaptive routing. As defined, this methodology supplies non-minimal routing functions. It is possible to define minimal routing functions by restricting Q_{xy} to contain only the queues belonging to minimal paths from x to y.

Lemma 1. *Methodology 1 supplies deadlock-free routing algorithms.*

Proof. According to the definition for R

$$R_1(x, y) = R(x, y) \cap Q_1 \ \forall \ x, y \in N$$

Thus, there exists a subset of queues $Q_1 \subset Q$ that defines a routing subfunction R_1 that is connected and deadlock-free. Taking into account Theorem 3, we can conclude that R is deadlock-free. □

The proposed methodology is very simple to apply. Some examples can be found in [11] for a binary n-cube with wormhole switching. For the sake of completeness, we will briefly present an example in Section 6. The resulting routing algorithms are the same for virtual cut-through and wormhole switching. The only difference being that, for virtual cut-through switching, the routing function can be defined in such a way that it also uses non-minimal paths. As shown above, the resulting routing algorithms are always deadlock-free. It is not necessary to check that the extended methodology resource dependency graph is acyclic. Thus, this design is more flexible and much easier to apply than the one we proposed for wormhole switching [11].

As defined, methodology 1 makes no sense for routing functions defined on $Q_{IN} \times N$. This methodology extends the range of the routing function but not its domain. If the domain is not extended, packets stored in the new queues cannot be routed. However, if the domain of R is extended, the initial routing function R_1 is modified. As an example, let us consider the positive hop algorithm [19]. This fully adaptive minimal routing algorithm uses central queues grouped into classes. Every time a packet stored in a queue of class i crosses a channel, it moves to a queue of class $i + 1$. This routing function requires the class of the queue containing the packet to compute the next class to be used. However, it has no memory of the previous path followed by the packet. Let Q_1 be the set of queues at this point. Suppose that the positive hop algorithm is extended for non-minimal routing by adding a new queue to each node so that the new queues are used for fully adaptive non-minimal routing. Once the new queues have been added, the positive hop algorithm does not work unless additional information is carried in the packet header. Effectively, when a packet moves from a queue of class i belonging to Q_1 to a new queue, and then it returns to another queue of Q_1, the routing function does not remember the previous class for this packet. In summary, extending the domain of the routing function supplied by methodology 1 modifies the initial routing function. As a consequence, the methodology makes no sense for routing functions defined on $Q_{IN} \times N$.

6 Design Examples

In this section, we present some examples of deadlock-free routing algorithms for virtual cut-through and store-and-forward switching using edge buffers. Those algorithms will be proved to be deadlock-free by using the theory proposed in previous sections. First, we present a simple example using the design methodology proposed in Section 5. Then, we present a routing algorithm that is not deadlock-free if wormhole switching is used.

Let us apply the design methodology proposed in Section 5 to n-dimensional meshes. For step 1 we use dimension-order routing. It is well known that this routing function is connected and deadlock-free. For step 2, consider that each physical channel c_i has been split into two virtual channels, namely, a_i and b_i. Let C_1 be the set of b channels. The algorithm obtained applying step 2 can be stated as follows: Route using any of the a channels. If all of them are busy, route using dimension-order routing on b channels. This routing function allows packets to follow any minimal or non-minimal path. Of course, it is also possible to restrict the use of a channels to minimal paths.

Let us consider a 2D-mesh without virtual channels using the north-last routing function [18]. For the sake of simplicity, consider that only minimal paths are allowed. Channels corresponding to north, east, south and west directions will be denoted as N, E, S and W. The north-last routing function allows packets to follow any minimal path with one exception: Packets are not allowed to turn after using N channels. Thus, it is not fully adaptive. As shown in Section 4.5, fully adaptive routing without virtual channels may produce deadlocks. We are going to add the minimum number of virtual channels to obtain a fully adaptive deadlock-free routing function. Also, we will show that it is not deadlock-free if wormhole switching is used.

Consider that N channels are split into two virtual channels, namely, N_1 and N_2. After using N_1 channels, no turns are allowed. However, 90-degree turns are allowed after using N_2 channels. The new routing function R is fully adaptive and deadlock-free. Effectively, consider that C_1 is the subset containing N_1, E, S and W channels. C_1 defines a routing subfunction R_1 identical to the north-last routing function. It is easy to see that R_1 can be defined on $N \times N$. As shown in [18], R_1 is connected and deadlock-free. Thus, according to Theorem 3, R is deadlock-free for virtual cut-through and store-and-forward switching.

However, R is not deadlock-free in a wormhole network. Fig. 6 shows a deadlocked configuration on a 3 × 3 mesh. Solid lines represent the channel(s) already reserved by each packet. Dashed arrows represent the next channel requested by each packet. Dashed arrows also point at the destination node for the packet. We have chosen a configuration such that a single routing option is available for each packet. As can be seen, no packet is able to advance. One of the packets has reserved one E channel and two N_2 channels, then requesting another E channel. Thus, such a deadlocked configuration is only valid for wormhole switching.

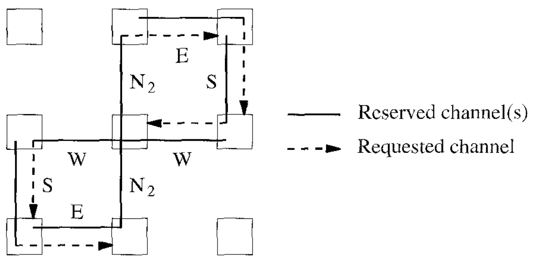


Fig. 6. Deadlocked configuration for R on a 3 × 3 mesh using wormhole switching.

7 Conclusions

A theory of deadlock-free adaptive routing has been proposed for virtual cut-through and store-and-forward networks with edge buffers and/or central buffers. This theory extends our previous results for wormhole networks. Theorem 2 proposes a necessary and sufficient condition for deadlock-free adaptive routing. This condition establishes that a routing function is deadlock-free if and only if there exists a restricted routing function that is able to deliver all the packets and it has no cyclic dependencies between the queues supplied by it. This condition is very similar to the one we proposed for wormhole switching. However, the restrictions required for wormhole switching have been removed.

Theorem 2 and Corollary 1 give a very flexible condition for the design of adaptive routing algorithms, by allowing the existence of cyclic dependencies between resources. For routing functions that only consider the current and destination nodes, theorem 3 adds more flexibility, allowing the addition of edge or central buffers to a deadlock-free routing function. The new buffers can be used in any way, even following non-minimal paths. To simplify the application of Theorem 3, a design methodology has been proposed. It supplies fully adaptive routing algorithms which are guaranteed to be deadlock-free. Finally, we have presented some design examples, showing the application of the theory.

Acknowledgments

This work was supported in part by the Spanish CICYT under Grant TIC94-0510-C02-01.

I would like to thank the anonymous referees for their helpful comments and suggestions.

References

[1] A. Agarwal, "Limits on Interconnection Network Performance," *IEEE Trans. Parallel Distributed Systems*, vol. 2, no. 4, pp. 398–412, Oct. 1991.

[2] A. Arruabarrena, R. Beivide, C. Izu and J. Miguel, "A Performance Evaluation of Adaptive Routing in Bidimensional Cut-Through Networks," *Parallel Processing Letters*, vol. 3, no. 4, pp. 469–489, 1993.

[3] W.C. Athas and C.L. Seitz, "Multicomputers: Message-Passing Concurrent Computers," *IEEE Computer*, vol. 21, no. 8, pp. 9–24, Aug. 1988.

[4] K.W. Bolding and L. Snyder, "Mesh and Torus Chaotic Routing," *Proc. MIT/Brown Conf. Advanced Research in VLSI*, 1992.

[5] 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.

[6] R. Cypher and L. Gravano, "Requirements for Deadlock-Free, Adaptive Packet Routing," *Proc. 11th ACM Symp. Principles Distributed Computing*, 1992.

[7] R. Cypher and L. Gravano, "Adaptive, Deadlock-Free Packet Routing in Torus Networks with Minimal Storage," *Proc. Int'l Conf. Parallel Processing*, Aug. 1992.

[8] W.J. Dally and C.L. Seitz, "The Torus Routing Chip," *Distributed Computing*, vol. 1, no. 3, pp. 187–196, Oct. 1986.

[9] W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans Computers*, vol. 36, no. 5, pp. 547–553, May 1987.

[10] J. Duato, "On the Design of Deadlock-Free Adaptive Routing Algorithms for Multicomputers: Design Methodologies," *Proc. Parallel Architectures and Languages Europe 91*, June 1991.

[11] J. Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Trans. Parallel Distributed Systems*, vol. 4, no. 12, pp. 1,320–1,331, Dec. 1993.

[12] J. Duato and P. López, "Performance Evaluation of Adaptive Routing Algorithms for k-ary n-cubes," *Proc. Workshop Parallel Computer Routing and Communication*, May 1994.

[13] J. Duato, "A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks," *Proc. Int'l Conf. Parallel Processing*, Aug. 1994.

[14] J. Duato, "A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Trans. Parallel Distributed Systems*, vol. 6, no. 10, pp. 1,055-1067, 1995.

[15] P.T. Gaughan and S. Yalamanchili, "Adaptive Routing Protocols for Hypercube Interconnection Networks," *IEEE Trans. Computers*, vol. 26, no. 5, pp. 12–23, May 1993.

[16] D. Gelernter, "A DAG-Based Algorithm for Prevention of Store-and-Forward Deadlock in Packet Networks," *IEEE Trans Computers*, vol. 30, pp. 709–715, Oct. 1981.

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

[18] C.J. Glass and L.M. Ni, "The Turn Model for Adaptive Routing," *Proc. 19th Ann. Int'l Symp. Computer Architecture*, May 1992.

[19] I.S. Gopal, "Prevention of Store-and-Forward Deadlock in Computer Networks," *IEEE Trans. Commun.*, vol. 33, no. 12, pp. 1,258–1,264, Dec. 1985.

[20] A.G. Greenberg and B. Hajek, "Deflection Routing in Hypercube Networks," *IEEE Trans. Commun.*, vol. 40, pp. 1,070–1,081, 1992.

[21] K.D. Gunther, "Prevention of Deadlocks in Packet-Switched Data Transport Systems," *IEEE Trans. Commun.*, vol. 29, pp. 512–524, Apr. 1981.

[22] P.A.J. Hilbers and J.J. Lukkien, "Deadlock-Free Message Routing in Multicomputer Networks," *Distributed Computing*, vol. 3, pp. 178-186, 1989.

[23] P. Kermani and L. Kleinrock, "Virtual Cut-Through: A New Computer Communication Switching Technique," *Computer Networks*, vol. 3, pp. 267–286, 1979.

[24] C.K. Kim and D.A. Reed, "Adaptive Packet Routing in a Hypercube," *Proc. 3rd Conf. on Hypercube Concurrent Computers & Applications*, Jan. 1988.

[25] S. Konstantinidou and L. Snyder, "The Chaos Router: A Practical Application of Randomization in Network Routing," *Proc. 2nd ACM Symp. Parallel Algorithms Architectures*, 1990.

[26] S. Konstantinidou and L. Snyder, "Chaos Router: Architecture and Performance," *Proc. 18th Ann. Int'l Symp. Computer Architecture*, 1991.

[27] S. Konstantinidou and L. Snyder, "The Chaos Router," *IEEE Trans. Computers*, vol. 43, no. 12, pp. 1,386–1,397, Dec. 1994.

[28] P.M. Merlin and P.J. Schweitzer, "Deadlock Avoidance in Store-and-Forward Networks–I: Store-and-Forward Deadlock," *IEEE Trans. Commun.*, vol. 28, no. 3, pp. 345–354, Mar. 1980.

[29] J.Y. Ngai and C.L. Seitz, "A Framework For Adaptive Routing in Multicomputer Networks," *Proc. ACM Symp. Parallel Algorithms Architectures*, 1989.

[30] T. Nguyen and L. Snyder, "Performance of Minimal Adaptive Routers," *Proc. Parallel Computer Routing and Communication Workshop*, May 1994.

[31] L.M. Ni and P.K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *IEEE Trans. Computers*, vol. 26, no. 2, pp. 62–76, Feb. 1993.

[32] A.G. Nowatzyk, M.C. Browne, E.J. Kelly and M. Parkin, "S-Connect: From Networks of Workstations to Supercomputer Performance," *Proc. 22nd Int'l Symp. Computer Architecture*, June 1995.

[33] G.D. Pifarré, L. Gravano, S.A. Felperin, and J.L.C. Sanz, "Fully-Adaptive Minimal Deadlock-Free Packet Routing in Hypercubes, Meshes and Other Networks," *Proc. 3rd ACM Symp. Parallel Algorithms Architectures*, June 1991.

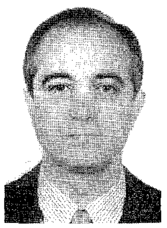
[34] G.D. Pifarré, L. Gravano, S.A. Felperin, and J.L.C. Sanz, "Fully Adaptive Minimal Deadlock-Free Packet Routing in Hypercubes, Meshes and Other Networks: Algorithms and Simulations," *IEEE Trans. Parallel Distributed Systems*, vol. 5, no. 3, pp. 247–263, Mar. 1994.

[35] S. Ragupathy, M.R. Leutze, and S.R. Schach, "Message Routing Schemes in a Hypercube Machine," *Proc. 3rd Conf. Hypercube Concurrent Computers & Applications*, Jan. 1988.

[36] J. Rexford and K.G. Shin, "Support for Multiple Classes of Traffic in Multicomputer Routers," *Proc. Parallel Computer Routing and Communication Workshop*, May 1994.

[37] A.W. Roscoe, "Routing Messages Through Networks: An Exercise in Deadlock Avoidance," Oxford Univ. Computing Laboratory Report, 1987.

[38] A.S. Tanenbaum, "*Computer Networks*," 2nd ed., Prentice-Hall, 1988.



José Duato received the electrical engineering and PhD degrees from the Polytechnical University of Valencia, Valencia, Spain, in 1981 and 1985, respectively. In 1981 he joined the Department of Systems Engineering, Computers and Automation at the Polytechnical University of Valencia, where he is currently a professor of computer architecture and technology, and dean of the School of Computer Science. He has developed several courses on computer organization, peripheral devices, computer architecture and multicomputer design. His current research interests include parallel architectures, interconnection networks, and simulation. Dr. Duato is serving as a member of the editorial board of *IEEE Transactions on Parallel and Distributed Systems.* He received the Best Curriculum in Electrical Engineering Nation-Wide Award in 1982. He served as vice-dean of the School of Computer Science from 1986 to 1992. He is a member of the Committee of Professors at the Polytechnical University of Valencia.