A General Theory for Deadlock-Free Adaptive Routing Using a Mixed Set of Resources

José Duato, *Member*, *IEEE*, and Timothy Mark Pinkston, *Senior Member*, *IEEE*

Abstract—This paper presents a theoretical framework for the design of deadlock-free fully adaptive routing algorithms for a general class of network topologies and switching techniques in a single, unified theory. A general theory is proposed that allows the design of deadlock avoidance-based as well as deadlock recovery-based wormhole and virtual cut-through adaptive routing algorithms that use a homogeneous or a heterogeneous (mixed) set of resources. The theory also allows channel queues to be allocated nonatomically, utilizing resources efficiently. A general methodology for the design of fully adaptive routing algorithms applicable to arbitrary network topologies is also proposed. The proposed theory and methodology allow the design of efficient network routers that require minimal resources for handling infrequent deadlocks.

Index Terms—General theory for deadlock-free fully adaptive routing, regular networks, irregular networks, nonatomic queue allocation.

1 Introduction

The performance of parallel computer systems hinges on the efficient utilization of system resources, especially the interconnection network. Networks consist of switching components, called routers, and communication links. The network topology is the interconnection pattern between routers. Network topologies are described as being direct or indirect, depending upon whether nodes connect to other nodes or a subset of other nodes directly (because a router is integrated inside each node) or indirectly through a set of routers. Networks are also described as being regular or irregular, depending upon whether the router degree is invariant or variant and whether the connections between routers follow regular or irregular (arbitrary) patterns.

Traditionally, direct and indirect regular networks have been prevalent in massively parallel processor (MPP) systems, such as the IBM SP2 [1], Cray T3E [2], and SGI Origin [3]. Recently, however, direct and indirect irregular networks, such as Autonet [4], Myrinet [5], and ServerNet II [6], have gained popularity for network-based computing systems such as a network of workstations (NOWs). Irrespective of network type—whether direct or indirect, regular or irregular—communication performance can be optimized by incorporating switching, flow control, and routing techniques that pipeline, multiplex, and increase flexibility in packet transmission over the network while guarding against deadlock. This maximizes the utilization of scarce network resources to facilitate low latency, high throughput communication.

- *J. Duato is with Facultad de Informática, Universidad Politécnica de Valencia, PO Box 22012, 46071 Valencia, Spain. E-mail: jduato@gap.upv.es.*
- *T.M. Pinkston is with SMART Interconnects Group, EE-Systems Department, University of Southern California, 3740 McClintock Ave., EEB 208, Los Angeles, CA 90089-2562. E-mail: tpink@charity.usc.edu.*

Manuscript received 10 Jan. 2000; revised 21 Feb. 2001; accepted 11 June 2001.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 111198.

Pipelined transmission across multiple routers over network links at the physical unit (phit) level can be accomplished using wormhole switching [7] or virtual cut-through switching [8]. Both techniques and variants of them, such as buffered wormhole switching [1], are interesting for MPP as well as NOW network implementations [2], [5], [9], [10], [11]. However, since both techniques require channel queues at least the size of flow control units (flits), efficient use of queues is critical to maximizing resource utilization. The size of flits (and, therefore, queues) varies widely between the two techniques. Wormhole requires flits the size of one or more phits, whereas virtual cut-through requires flits the size of one packet. The increased queue requirement of virtual cut-through is offset by the benefit of reduced network contention when packets block; all phits of a blocked packet are removed from network links instead of blocking in place across multiple routers. Pipelined transmission of multiple phits in flight simultaneously over each network link [12] of long or dissimilar length (such as in NOWs) may further increase queue requirements. Queues with capacity proportional to link length are required in order to avoid queue overflow during the overlapped propagation of flow control signals (i.e., acknowledgments or credits) [5]. Although larger queues are required, these pipelining techniques allow network bandwidth to be used efficiently at the phit level so long as the issue of *when* queues can be allocated is optimized, i.e., nonatomic as opposed to atomic use of queues.

Multiplexed transmission over network links at the flit level can be accomplished by decoupling channel queues from physical links. Channel multiplexing can be used not only to increase the flow capabilities of a network but also to guard against network deadlock [7]. However, the allocation of queue resources for handling deadlock takes away resources for increasing flow. One technique that is particularly useful for wormhole networks is virtual channel flow control [13]. Virtual channels implemented as edge queues allow flits of nonblocked packets to pass flits of blocked packets sharing the same physical link. This technique has widespread use in many MPP and NOW networks [2], [10], [11], [14], [15]. Alternatively, decoupling can be achieved through the use of central queues and is most applicable to buffered wormhole and virtual cut-through networks [1], [16]. Although more queues (and arbitration for queues) are required, these multiplexing techniques allow network bandwidth to be utilized efficiently at the flit level so long as the issue of *how* queues are allocated is optimized, i.e., queues are used for flow control as opposed to deadlock handling.

Flexible transmission over the network at the packet level can be accomplished using adaptive routing. This technique reduces contention by allowing packets to route around network congestion using alternative paths. However, deadlocks in routing can occur as a result of cyclic waits for network resources by packets holding onto resources while continuously and indefinitely waiting for other resources to become free. Avoidance has been the traditional approach for handling deadlocks in which strict routing rules are enforced to determine which of the available queue resources can be allocated in order to altogether prevent deadlocks from forming [17], [18], [19]. Alternatively, deadlocks can be handled by detecting and recovering from cyclic-wait dependencies arising from unrestricted routing on resources [20], [21], [22]. Currently, several routers implement fully adaptive routing based on either deadlock avoidance or deadlock recovery [2], [16], [23], [24]. Although deadlock handling is more complex, flexible multipath routing techniques such as adaptive routing allow network bandwidth to be utilized efficiently at the packet level so long as the issue of *which* queues can be allocated is optimized, i.e., unrestricted as opposed to restricted adaptive routing on queues.

The issues of when to allocate queues, how to allocate queues, and which queues to allocate are determined primarily by the routing algorithm and, secondarily, by the selection function that chooses among possibly many alternatives presented by the routing function. The development of a general theory and methodology for the design of routing algorithms that allows for the optimization of all these issues is important. In the past, theories and methodologies for designing deadlock-free adaptive routing algorithms implicitly or explicitly have been defined for and applicable only to particular cases, such as direct or indirect networks, regular or irregular topologies, wormhole or virtual cut-through switching, edge or central queues, and avoidance or recovery from potential deadlock (see Section 6 for a description of related work and the scope of previously proposed theories). Moreover, some theories for adaptive wormhole routing assume atomic queue allocation, preventing the sharing of queues by multiple packets [19], [25]. A general formal theory or methodology that encompasses all these cases for the design of fully adaptive routing algorithms had not yet been developed until now.

The theory proposed in this paper is generally applicable to direct and indirect networks as well as regular and irregular networks. It allows the design of deadlock avoidance-based as well as deadlock recovery-based wormhole and virtual cut-through adaptive routing algorithms that use a homogeneous or a heterogeneous (mixed) set of resources. It glues together several previously proposed theories [7], [19], [25], [26], [27] into a single theoretical framework. Moreover, the theory relaxes the atomic queue allocation requirement of previous theories [19], [25], allowing the allocation of queue resources to packets even when queues partially contain flits from other packets. This issue becomes more important as VLSI technology advances, allowing wormhole networks to be implemented with routers having large queue capacity which should be utilized efficiently. In addition, advances in VLSI technology allow faster clock frequencies and, thus, higher transmission rates. As a consequence, when routers and links are pipelined [28], [29], deeper buffers are required to tolerate flow control round-trip delay. Therefore, as buffer size increases, buffer utilization decreases unless the constraint on atomic queue allocation is removed. Hence, the additional generality provided by this theory allows adaptive routing algorithms to be designed that achieve maximum flexibility in routing and maximum utilization of queue resources.

The remainder of this paper is organized as follows: Section 2 provides background and an informal description of the general theory. Section 3 formally presents the proposed general theory for deadlock-free adaptive routing based on the notion of queues. Section 4 gives a general methodology for designing deadlock-free adaptive routing algorithms by applying the proposed theory. Section 5 provides a few examples of the application of the general methodology and gives a brief discussion. Section 6 discusses related work and, finally, conclusions are given in Section 7.

2 Informal Description

2.1 Network Model

Before proposing the general theory, we present the network model assumed that supports both direct and indirect network topologies. Following the unified view proposed in [30], we consider networks consisting of a set of interconnected routers,[1] each one being connected to zero, one, or more processing nodes through point-to-point links. Direct networks correspond to the case where every router is connected to a single node. Crossbar networks correspond to the case where there is a single router connected to all the nodes. Indirect networks with irregular topology correspond to the general case: Each router has zero, one, or more nodes directly attached to it and the interconnection scheme between routers does not follow any regular pattern. Fig. 1 illustrates a network with arbitrary topology.

2.2 Router Model

Fig. 2 shows the simple router model assumed for networks which use a mixed set of edge and central queue resources. In the figure, there are four physical network input/output

1. We use the word *router* instead of *switch* to avoid confusion with the internal switch inside the router.

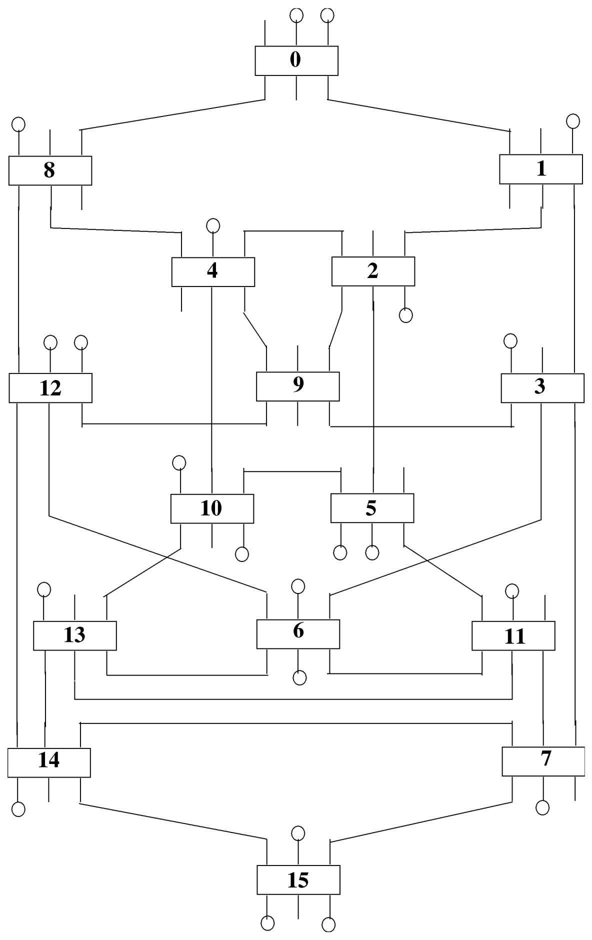


Fig. 1. An irregular network composed of 16 routers and 22 processor nodes. Small circles represent processor nodes and numbered rectangles represent router nodes.

channels attached to the router and one physical processor injection/delivery channel attached to the router. Each router consists of an internal switch (crossbar), a routing and arbitration module that implements the routing function, link input and output flow controllers (LCs), one (or more) edge queues and output buffers associated with each physical channel, an injection queue and delivery buffer, and one (or more) central queues.

The switch allows multiple packets to traverse a router simultaneously without blocking. The routing and arbitration module configures the switch, determining the output channel for each packet as a function of the destination node, the current router and/or current queue, and the output channel queue status. We assume that the routing module can process only one packet header at a time. If there is contention for the module, access is round robin. When a packet gets the module but cannot be routed because all the alternative output channels are busy, it must wait in the corresponding input channel queue until its next turn. Configurations that are able to process more than one packet header at a time can also be supported, provided that additional routing modules are linked by an arbiter with round-robin policy for fairness.

Physical channels are bidirectional full-duplex. However, they will be considered as two independent unidirectional

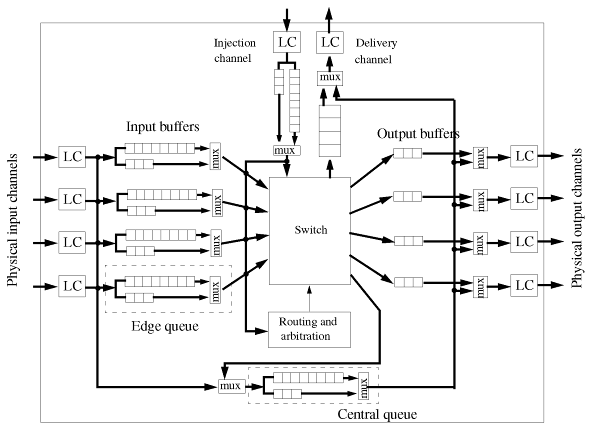


Fig. 2. Model for a router with a mixed set of resources.

channels. Each unidirectional physical channel may have a single edge queue associated with it (as shown in the figure) or may be split into multiple virtual channels implemented as edge queues. Virtual channels are assigned physical channel bandwidth cyclically only if a flit is ready to be transferred (demand-slotted round robin). Central queues can also accept flits and be assigned physical channel bandwidth. However, as central queues are shared resources, arbitration is needed when several concurrent requests arrive through different input links. A positive/negative acknowledgment is returned through the corresponding links to the requesting routers to notify that a central buffer has been granted/denied access. This arbitration process is required only once per packet. Finally, small output buffers are used to increase throughput of packets using edge queues and can be thought of as extensions of the queues. We will not consider them in the formal theory.

In order to deal with deadlocks, we are only interested in how packets are stored in the queues. Therefore, in what follows, virtual channels will be simply referred to as channels. If a physical channel is not split into virtual channels, we will also refer to it as a channel. Note that, in both cases, a channel is associated with a single edge queue.

2.3 Motivation for a General Theory

Physical links and queues are the primary network resources allocated to packets by the routing algorithm's *routing function*. A queue is assumed to be a set of one or more buffers with a FIFO policy. To decouple queue resources from physical links, multiple *edge queues* (virtual channels) can be associated with each link and one or more *central queues* can be associated with all the links of a router. Our attention can thus be restricted to the relationship among queue resources, also referred to simply as resources.

When a packet holds a resource and requests the use of another as provided by the routing function, a *dependency* exists between those resources. It does not matter which type of resource is involved in the dependency but, rather, only that the dependency exists. Hence, we will indistinctively refer to the dependencies among a mixed set of edge and central queues as resource dependencies or, simply, dependencies. An adaptive routing function could allow a packet to request several resources—producing several dependencies. In that case, a *selection function* is used to choose one of the possible resources supplied by the routing function. Since all packets being routed can produce such dependencies, deadlocks can arise in the network due to the formation of cyclic dependencies by packets occupying and holding some resources while requesting and waiting to acquire others.

Deadlock can be avoided simply by not allowing cyclic dependencies to form [7]. Alternatively, deadlock can be *avoided* or *progressively resolved* by providing a way of escape for packets involved in cyclic dependencies using alternative resources. The resources belonging to the set used to escape from deadlock are referred to as *escape resources* or *escape queues* and can consist of edge queues, central queues, or a mixture of the two. The routing function restricted to using only the escape resources is referred to as the *routing subfunction*. A routing subfunction is only a mathematical tool used to restrict our attention to a subset of resources that could be supplied by an overall routing function.

Routing algorithms that avoid deadlock are referred to as being *avoidance-based* and those that recover from potential deadlock are referred to as being *recovery-based*. Both routing approaches allow fully adaptive routing. However, two salient differences between the two routing approaches are apparent: 1) the rules governing how resources are allocated to handling deadlock versus increasing flow and 2) the rules governing which (and how many) resources are allocated by the routing function. Deadlocks have been shown to occur infrequently when the network and routing algorithm provide sufficient routing freedom [31], [32]. It therefore suffices to dedicate minimal resources for handling these exceptional cases through recovery—fewer resources are required with recovery than with avoidance. In avoidance-based routing, escape resources are allocated immediately when other resources are busy, provided that escape resources are free. However, escape resources and associated physical link bandwidth are allocated less frequently with progressive recovery in which escape resources are used only when possible deadlock is detected [33]. As a consequence, progressive recovery routing relaxes routing restrictions, optimizing the allocation of resources for the more common case of no deadlock. For example, it is possible to use all the edge virtual channels for fully adaptive routing and only a few central flit buffers for deadlock recovery instead of dedicating some edge virtual channels for avoiding rare deadlocks [27].

Previous theories for avoidance-based adaptive wormhole routing have another basic limitation: Resources are assumed to be only of one type—edge *or* central queues, but not both. The more general case of mixed set of resources is not considered. Moreover, for both avoidance and recovery-based theories, resources are assumed to be allocated *atomically*; that is, flits of only one packet can occupy any queue at any given time. This assumption guarantees that all blocked packets can reach the head of a queue to gain access to the escape path at every router. While this may be necessary for adaptive wormhole routing algorithms that allow packets to return to nonescape resources from escape

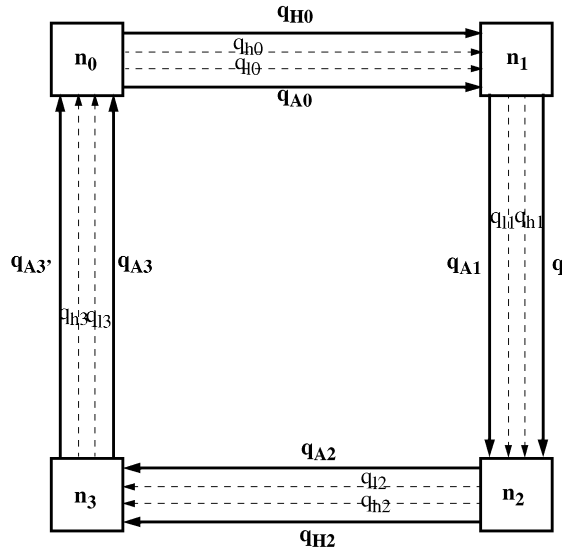


Fig. 3. Unidirectional ring network with two edge queues (solid line) per physical channel and two central queues (dashed lines) per router for the case of progressive recovery-based routing only.

resources, the general theory presented in this paper shows that it is not required for algorithms that keep packets on escape resources until being delivered. In this case of *nonatomic* allocation, it suffices to guarantee that at least one packet in a deadlocked configuration can gain access to the escape path, not all packets. Relaxation of these assumptions as allowed by our general theory provides for the design of avoidance-based or progressive recovery-based routing algorithms that strike a proper balance in maximizing both routing adaptivity and resource utilization while minimizing the number of escape resources required. This is illustrated by the following example:

Consider a wormhole-switched unidirectional ring network with four router nodes, denoted n_i, indexed by $i = \{0, 1, 2, 3\}$, as shown in Fig. 3. Let us examine an avoidance-based wormhole routing algorithm provided by previous theories ([19], [25]) which atomically uses a homogeneous set of resources (only edge queues). We compare this with a progressive recovery-based routing algorithm provided by the proposed general theory which nonatomically uses a mixed set of resources (both edge and central queues). Both routing algorithms make use of the same amount of buffer space configured differently. In one configuration, each node uses two edge queues each consisting of B flit-sized buffers per physical link between adjacent node pairs. In the other configuration, each node uses two edge queues, each consisting of $B-1$ flit-sized buffers per physical channel, and two central queues, each consisting of one flit-sized buffer per router node. Let q_{Ai} and q_{Hi} denote the edge queues (solid arcs in the figure) and q_{hi} and q_{li} denote the central queues (dashed arcs in the figure) used for routing at node n_i. Note that those queues are physically located at node $n_{(i+1) \bmod 4}$ (next node in the ring). Also, outgoing channels from node n_3 have been labeled as q_{A3} and $q_{A3'}$ because both of them can be used for routing interchangeably.

The avoidance-based adaptive routing algorithm can be stated as follows: If the index i of the node to which the

TABLE 1
Edge Queues Allocated by the Routing Algorithms

Node n_i (Current Queue)	Destination Node n_j			
	n_0	n_1	n_2	n_3
n_0	—	q_{A0}, q_{H0}	q_{A0}, q_{H0}	q_{A0}, q_{H0}
n_1	q_{A1}, $\mathbf{q_{H1}}$	—	q_{A1}, q_{H1}	q_{A1}, q_{H1}
n_2	q_{A2}, $\mathbf{q_{H2}}$	q_{A2}, $\mathbf{q_{H2}}$	—	q_{A2}, q_{H2}
n_3	q_{A3}, $q_{A3'}$	q_{A3}, $q_{A3'}$	q_{A3}, $q_{A3'}$	—

The additional edge queues allocated by progressive recovery-based routing (but not provided by avoidance-based routing) are shown in bold.

current queue belongs is equal to the index j of the destination node n_j, deliver the packet. Otherwise, use q_{Ai} $\forall j \neq i$ or use q_{Hi} $\forall j > i$. In other words, q_{Ai} queues (including $q_{A3'}$) can be used to forward packets to all destinations, but q_{Hi} queues can be used only if the destination node index is higher than the index of the node to which the current queue belongs. The selection function chooses any edge queue presented by the routing function that is unoccupied.

The progressive recovery-based adaptive routing algorithm can be stated as follows: If the index i of the node to which the current queue belongs is equal to the index j of the destination node n_j, deliver the packet. Otherwise, use either q_{Ai} or q_{Hi} $\forall j \neq i$ if the packet currently occupies an edge queue or use q_{hi} $\forall j > i$ or q_{li} $\forall j < i$ if the packet currently occupies an edge or central queue. In other words, both q_{Ai} and q_{Hi} edge queues (including $q_{A3'}$) can be used to forward packets currently in edge queues to all destinations, but q_{hi} or q_{li} central queues can be used only if the destination node index is higher or lower, respectively, than the index of the node to which the current queue (edge or central) belongs. When packets are in edge queues, the selection function chooses the central queue only when possible deadlock is suspected (i.e., detected); otherwise, any edge queue presented by the routing function that has space for at least one flit is chosen. The resources supplied by the routing functions of the two approaches are summarized in Table 1.

The avoidance-based routing algorithm requires that all queues be allocated atomically and restricts routing on one set of edge queues to escape deadlock. The progressive recovery-based routing algorithm relaxes both these requirements, allowing sharing of queues and unrestricted routing of packets on all edge queues. The central queues comprise the escape resource subset as packets use only those queues in a cycle-free manner once allocated. Although each edge queue is larger by one buffer in the avoidance-based scheme, the small increase in size compared to the recovery-based scheme is of little consequence. This is because edge queues of the avoidance-based scheme cannot be used by more than one packet at a time. In fact, depending on the ratio of queue size to packet size, the *effective* size of the queue for the recovery scheme could be much greater, on average, owing to its higher utilization. This added to the increased flexibility in routing packets over edge queues, as shown in Table 1 (the bold queues are not allowed by the avoidance-based scheme), can offset the possible marginal increase in cost of implementing separate flit-sized central queues.

Minimal resources are allocated to handling deadlocks in the recovery-based scheme. Only two central queues, each of only one flit-sized buffer shared by all links of the node, are used to escape from deadlock. The use of these escape resources is likely to be very infrequent since prior studies have shown deadlocks occur rarely [32], [34]. Although having larger sized central queues would allow for pipelined transmission of packets over those queues, the rarity of these events likely would not warrant the resource increase. Hence, the optimistic organization of buffer space in the recovery-based scheme provides the same number of edge queues as in the avoidance-based scheme, but allows packets more routing freedom (allocation of resources) in the form of true fully adaptive routing over all nonescape resources. This flexibility improves for multidimensional wrap-around networks, such as n-D tori, $n > 1$. Although increased routing freedom usually results in increased throughput, performance does not necessarily increase linearly with routing freedom. Initially, alternative paths increase throughput significantly, but additional routing flexibility beyond a certain point yields diminishing returns [32], [34], very much like adding virtual channels [13].

Since the resources supplied to packets occupying edge queues may be different from those supplied to packets occupying central queues, the general theory defines routing functions such that the type of queue where the packet header is stored must also be considered. The general theory, however, makes no distinction between deadlock avoidance and deadlock recovery; routing algorithms for either approach can be derived from the general theory. It is also possible to derive routing algorithms such that the selection function determines the approach used, whether avoidance-based or progressive recovery-based. The selection function should allocate escape resources in such a way that they do not prematurely become saturated, causing a performance bottleneck on those resources. Therefore, the domain of the selection function could include information about possible deadlock detection and eligibility of packets to undergo recovery, such as link inactivity [35], [33] and/or routing turn status [24]. This is the case for the routing algorithm derived from the general theory described in the example in Fig. 3. The recovery-based scheme could be converted into an avoidance-based scheme if the restriction on when the selection function chooses central queues is lifted. Note that the opposite is not true for the scheme derived from previous theories: If the escape queues are allocated only when impending deadlock is suspected, the routing function would be disconnected for all packets with current-destination node pairs such that $j > i$ and no impending deadlock is detected.

3 Formal General Theory

The theorems derived in [19], [25] cannot be applied to schemes such as *Disha* [21], [26] which use both edge queues and central queues. As the routing function for edge queues may be different from that for central queues, the routing function must consider the type of queue in which the packet is stored at the current router. This section develops the theoretical background for the design of deadlock-free adaptive routing algorithms that use a mixed set of resources (central queues and edge queues) under wormhole switching without requiring atomic use of queue resources. This is a generalized version of the theories presented in [19] and [25] for wormhole networks. Later, the theory is extended to virtual cut-through switching. Additionally, definitions have been rewritten in order to support both direct and indirect network topologies. The main theoretical difference between this general theory and previous theories is in the way concepts are defined and used, not in the theorem statements and proofs. Once the concepts are defined and their use shown, the proofs follow along similar previous lines. With this generalization, the theory is applicable to both deadlock avoidance and progressive deadlock recovery routing schemes for direct and indirect networks with regular and irregular topologies using atomic or nonatomic queue allocation. This is shown in the sections to follow.

3.1 Assumptions

The basic assumptions are similar to those proposed in [19], modified to support a mixed set of resources. 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 main assumptions on which the theory is based are the following:

Assumption 1. *A node can generate messages of arbitrary length destined for any other node at any rate. Messages may be split into packets.*

Assumption 2. *A packet arriving at its destination node is eventually consumed.*

Assumption 3. *Each channel has at most a single edge queue (or multiqueue, e.g., DAMQ [36]) associated with it, allowing the storage of a finite number of flits. In case a given channel has no edge queue associated with it, it must have access to one or more central queues where incoming flits will be buffered. If each channel at a router has an edge queue associated with it, it may also have access to one or more central queues with capacity for a finite number of flits.*

Assumption 4. *Once a queue accepts the first flit of a packet, it must accept the remainder of the packet before accepting any flits from another packet. Also, a packet may occupy several queues simultaneously. Central queues must be reserved before starting packet transmission. Arbitration for central queues is assumed to follow a round-robin strategy, thus preventing starvation.*

Assumption 5. *Depending on the deadlock handling strategy, atomic queue allocation may be required. When atomic allocation is required, a queue must be completely emptied before accepting any flits from another packet. When it is not required, the header flit of a packet may immediately follow the tail flit of another packet in a queue.*

Assumption 6. *The routing and arbitration module may arbitrate between packets requesting it, but it may not choose among waiting packets.*

Assumption 7. *The route taken by a packet depends on its destination node, the current router and/or queue containing the packet header, and the status of the requested channels or queues (free or busy). At a given router, an adaptive routing function supplies a set of channels and/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 according to some selection criteria.*

Assumption 8. *When several packets are waiting because all the resources supplied by the routing function are busy, they are routed following a round-robin strategy to prevent starvation.*

Assumption 9. *The routing function may allow packets to follow nonminimal paths.*

3.2 Definitions

Before proposing the theorems, some definitions are needed. These definitions are very similar to the ones proposed in [37] for virtual cut-through and store-and-forward switching because those definitions, as well as the ones in this section, consider the input queue containing the packet header in the domain of the routing function. However, some modifications have been introduced to support wormhole switching. Additionally, definitions and theorems consider both edge and central queues as well as direct and indirect networks.

Also, without loss of generality, a destination node will be specified by indicating the router attached to it and the output port at that router. This strategy supports both direct and indirect networks. In the case of direct networks, the router and the node have the same identifier and the output port is the delivery port at that router. In this case, it is not necessary to specify the output port because it is implicitly assumed. In the case of indirect networks, several processing nodes may be connected to a single router. From a formal point of view, the destination node will be specified by indicating the delivery queue. Note that a node may be connected to a router through several links to increase performance. Also, a node in an indirect network may be connected to several routers through different links for increased reliability. The proposed theory supports all these cases.

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 routers. The arcs of the multigraph C represent the set of channels. Each channel has a single edge queue associated with it. More than a single channel is allowed to connect a given pair of routers.*[2] *The source and destination routers of channel $c_i \in C$ are denoted as s_i and d_i, respectively. Each processing node has one or more injection channels and one or more delivery channels directly*

2. This includes the case where a physical channel is split into multiple virtual channels as well as the case where there exist multiple physical channels between a given pair of routers.

connected to a router. These channels have their corresponding queues denoted as injection and delivery queues, respectively. The sets of injection and delivery queues are denoted as 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 router $n_j \in N$, where $j = \lfloor i/q \rfloor$ and q is the number of central queues per router, or it can be associated with a channel c_i (edge queue). In the latter case, we will consider that q_i is at router d_i. For consistency, we assume that delivery queues are not at any router. Instead, they are in the network interface controller at the destination node. 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, *$F = \{free, busy\}$. When atomic queue allocation is required, a queue is free when it is empty and it is busy otherwise. When atomic queue allocation is not required, a queue is free when it is not full and it is busy otherwise.*

Definition 3. *An* adaptive routing function

$$R : Q_{IN} \times Q_D \rightarrow \mathcal{P}(Q_{ND}),$$

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 a delivery queue q_d,

$$R(q_c, q_d) = \{q_1, q_2, \ldots, q_p\}.$$

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,

$$p = 1 \quad \forall (q_c, q_d) \in Q_{IN} \times Q_D$$

defines a deterministic routing function. If the current queue q_c is at destination router n_d then the routing function can only supply a delivery queue at the current router. Defining the domain of the routing function as $Q_{IN} \times Q_D$ is more general than considering only the current and destination routers. Thus, all the theoretical results for routing functions defined on $Q_{IN} \times Q_D$ are also valid for routing functions defined on $N \times Q_D$, $Q_{IN} \times N$, and $N \times N$.

Definition 4. *A* selection function *$S : \mathcal{P}(Q_{ND} \times F) \rightarrow Q_{ND}$ selects a free queue (if any) from the set supplied by the routing function according to some selection criteria. 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 routing module, according to Assumption 8. The selection function cannot induce starvation. It affects performance and can also reflect the routing approach (deadlock avoidance-based or progressive recovery-based).*

Definition 5. *A* configuration *is an assignment of a set of flits to each queue. The number of flits in queue q_i is denoted $size(q_i)$. A packet is in a queue if some flits of that packet are stored in that queue. The destination (delivery queue) for a packet p_j will be denoted $dest(p_j)$. If the first flit in queue q_i is destined for a delivery queue $q_{d'}$ at router n_d, then $rhead(q_i) = n_d$ and $head(q_i) = q_{d'}$. The functions $rhead()$ and $head()$ return the destination router and the destination delivery queue, respectively. If the first flit is not a header and the next queue reserved by its header is q_j, then $next(q_i) = q_j$. Let $Q_h \subseteq Q_{IN}$ be the set of queues containing a header flit at their queue head. Let $Q_d \subseteq Q_{IN}$ be the set of queues containing a data or tail flit at their queue head. A configuration is legal iff*

$$\forall q_i \in Q,\ size(q_i) \leq cap(q_i),\ and$$
$$\forall q_i \in Q_{ND},\ \forall p_j \in q_i,\ \exists q_0 \in Q_I,\ \exists q_1, q_2, \ldots, q_{i-1} \in Q_N$$
$$such\ that\ q_m \in R(q_{m-1}, dest(p_j)),\ m = 1, \ldots, i.$$

That is, for each queue, the capacity is not exceeded and all the packets with flits 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 satisfying the following conditions:*

1. $Q_h \neq \emptyset$.
2. $$\forall q_i \in Q_h \begin{cases} rhead(q_i) \neq d_i & \text{(for edge queues)} \\ rhead(q_i) \neq n_{\lfloor i/q \rfloor} & \text{(for central queues)} \\ q_j\ isbusy & \forall q_j \in R(q_i, head(q_i)). \end{cases}$$
3. $$\forall q_i \in Q_d$$ $$\begin{cases} rhead(q_i) \neq d_i & \text{(for edge queues)} \\ rhead(q_i) \neq n_{\lfloor i/q \rfloor} & \text{(for central queues)} \\ size(next(q_i)) = cap(next(q_i)). \end{cases}$$

In a deadlocked configuration there is no packet whose header flit has already arrived at its destination node. Header flits cannot advance because all the alternative queues supplied by the routing function are busy. Consequently, there is at least one set of packets waiting cyclically for resources occupied by other packets in the cycle. Data and tail flits cannot advance because the next queue reserved by their packet header is full. Also, data flits can be blocked at a node even if there are free queues to reach their destination because data flits must follow the path reserved by their header. No condition is imposed on empty queues.

The above definition assumes that packet injection has been stopped. After delivering all the packets that are not deadlocked, the packets blocked at the network satisfy the above-mentioned conditions. This configuration is also referred to as a *canonical deadlocked configuration*. It should be noted that packets involved in a deadlocked configuration will remain blocked forever, regardless of whether there are other packets traveling across the network. Also, deadlocked configurations can be complex—there may exist several nested cycles as well as several disjoint cycles or disjoint sets of nested cycles. In what follows, two or more disjoint cycles (or sets of nested cycles) will be considered as separate and distinct deadlocked configurations, as in [32], [38].

Definition 7. *A routing function R for an interconnection network I is* deadlock-free *iff there is no deadlocked configuration for that routing function on that network.*

Note that we consider the network statically. We do not consider whether a deadlocked configuration is reachable or not by routing packets across the network. In other words, it may happen that a legal configuration is not reachable because two packets must cross the same channel at the same time in order to reach that configuration. While considering reachability in the definition of a routing function's deadlock properties is useful for a few theoretical cases, it appears to have no practical interest and considerably complicates the application of the theory.

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

$$\begin{gathered}\forall q_i \in Q_{IN} \text{ such that } size(q_i) > 0,\\ \exists q_{i+1}, \ldots, q_{i+k-1} \in Q_N,\ \exists q_{i+k} \in Q_D \text{ such that}\\ q_m \in R(q_{m-1}, head(q_i)),\ m = i+1, \ldots, i+k.\end{gathered}$$

In other words, it is always possible to establish a path for every packet from its current queue to its destination node (delivery queue). Note 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:*

$$R_1(q_i, q_d) \subseteq R(q_i, q_d) \quad \forall q_i \in Q_{IN}, \quad \forall q_d \in Q_D. \tag{1}$$

The set of all the queues supplied by R_1 is

$$Q_{ND1} = \bigcup_{\forall q_i \in Q_{IN},\ \forall q_d \in Q_D} R_1(q_i, q_d).$$

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:

$$R_1(q_i, q_d) = R(q_i, q_d) \cap Q_{ND1} \quad \forall q_i \in Q_{IN}, \quad \forall q_d \in Q_D \tag{2}$$

Expression (1) is more general than (2) because it allows us to remove a queue from R_1 only for some destinations. This generalization was useful to define necessary and sufficient conditions for deadlock-free adaptive routing [25], [37], [39], but its additional routing flexibility is quite limited in practice.

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 flits stored in q_i and*

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

That is, q_j can be requested by a packet header 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 flits stored in q_i and there exists a possibly empty set of queues $q_1, q_2, \ldots, q_k \in Q_N$ such that*

$$\begin{aligned}&q_1 \in R(q_i, head(q_i))\\ &q_{m+1} \in R(q_m, head(q_i)), m = 1, \ldots, k-1\\ &q_j \in R_1(q_k, head(q_i)).\end{aligned}$$

This definition is identical to Definition 10 except that it is restricted to queues supplied by R_1 and it considers that a packet may occupy several queues simultaneously. This definition considers all the possible dependencies between resources supplied by a routing subfunction. When the set of queues $\{q_1, q_2, \ldots, q_k\}$ is empty, there is a dependency between adjacent resources. Otherwise, there is a dependency between nonadjacent resources. In general, dependencies exist between both adjacent and nonadjacent resources. The different kinds of dependency defined in [39], [25] are particular cases of this definition. 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 if there is a directed edge 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 a dependency from q_i to q_j.*

3.3 Sufficient Condition

The simplest sufficient condition for deadlock-free routing consists of preventing cyclic dependencies among the mixed set of resources. The following theorem states this condition formally.

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 in Q. As R is connected, the minimals in that order are delivery queues. We construct a proof by contradiction. Assume there is a deadlocked configuration for R. Let q_i be a nonempty queue in Q such that all the queues less than q_i are empty. If q_i is minimal in the order (that is, it is a delivery queue), then the flit at the queue head can reach its destination in a single hop and there is no deadlock. Otherwise, using the queues less than q_i, the flit at the queue head of q_i can advance. As the flits in q_i can advance, there is no deadlock which contradicts the original assumption. □

The following theorem allows us to design adaptive routing functions with cyclic dependencies in their resource dependency graph. It supplies a sufficient condition for deadlock-free routing, avoidance-based or recovery-based. In order to prove this theorem, atomic queue allocation is required. This assumption was also made in [19] and [25]. With this assumption, the header flit of a blocked packet will always occupy the head of a queue. This constraint will be removed later in proving a more general theorem.

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

Proof Sketch. The case $R_1 = R$ is addressed in Theorem 1. 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 minimals in that order will necessarily be delivery queues. Again, we prove by contradiction. Suppose that there is a deadlocked configuration for R. There are two possible cases:

1. The queues belonging to Q_{ND1} are empty. As R_1 is connected and the header flits are at queue heads, each header can be routed using queues belonging to Q_{ND1}. Thus, there is no deadlock.
2. The queues belonging to Q_{ND1} are not empty. Let q_i be a nonempty queue of Q_{ND1} such that all the queues less than q_i in the order are empty. Again, there are two possible cases:

   2.1. If q_i is minimal, then it is a delivery queue. Thus, the flit at the queue head is not blocked and there is no deadlock.

   2.2. If q_i is not minimal, all the queues of Q_{ND1} less than q_i will be empty, allowing three possible cases:

   a. If q_i has a header at the queue head, it can be routed because R_1 is connected. There is no deadlock.

   b. If there is a data flit at the queue head of q_i and $next(q_i)$ belongs to Q_{ND1}, that flit can also advance.

   c. If $next(q_i)$ belongs to $Q_{ND} - Q_{ND1}$, consider the queue q_k containing the header flit of the data flits contained in q_i. It is possible to find a queue q_j belonging to Q_{ND1} to route that header, because R_1 is connected. In that case, there is a dependency from q_i to q_j (Definition 11). This implies that q_j is less than q_i in the order. Therefore, q_j is empty and there is no deadlock. □

The resources supplied by R_1 can be considered escape resources. These resources are either used to avoid deadlock (e.g., using edge queues) or to progressively recover from or avoid potential deadlock (e.g., using central queues), depending on the selection function. The main differences between deadlock avoidance and progressive deadlock recovery are the following:

- In avoidance-based routing, the selection function allows escape resources to be used immediately when normal resources are busy, whereas, in progressive recovery-based routing, the use of escape resources is delayed until a potential deadlock is detected.
- Resources (i.e., channel bandwidth) are not deallocated from normal packets in avoidance-based routing. However, some resources can be deallocated from normal packets and allocated to potentially deadlocked packets in progressive recovery-based routing.

Consider the case of progressively recovering from potential deadlock. In order to break a deadlocked configuration, it is only necessary to supply escape resources for one packet in that configuration [32], [38]. At first glance, it may seem that escape resources are not necessary at every router in the network. However, in practice, escape resources are usually required at every router. The reason is that it may happen that an escape resource cannot be used at a given router simply because the packet reserved several resources and the header of the packet is not at that router. If the conditions imposed by Theorem 2 are satisfied, it can be guaranteed that an escape resource will be available sooner or later for every packet involved in a cyclic dependency. However, if the assumption on atomic queue allocation is not satisfied, it may happen that the only packets in a deadlocked configuration that could escape from deadlock are permanently blocked because their headers are not at a queue head. The following example shows this situation on a 2D mesh using edge queues.

Consider that each physical channel c_i has been split into two virtual channels, namely a_i and b_i. These virtual channels are implemented using edge queues. The routing algorithm can be stated as follows: Route over any useful dimension (horizontal or vertical) using the a channels. If they are busy, route over the highest useful dimension using the corresponding b channel. A useful dimension is one that forwards a packet nearer to its destination. If channel queues are allocated atomically, the routing subfunction defined by b channels according to (2) is connected and has no cycles in its extended resource dependency graph [19]. Therefore, this routing algorithm is deadlock-free according to Theorem 2.

Deadlock is possible if queues are not allocated atomically, as is illustrated by Fig. 4. Shown is a configuration for a 2D mesh and its corresponding resource wait-for graph using the aforementioned routing algorithm. Resource wait-for graphs are graphical depictions of dynamic resource allocations and requests occuring in a network at a particular instant in time [38]. In the leftmost figure, arrows indicate the path followed by packet headers. Data flits located at queue heads have to follow these paths. Dashed

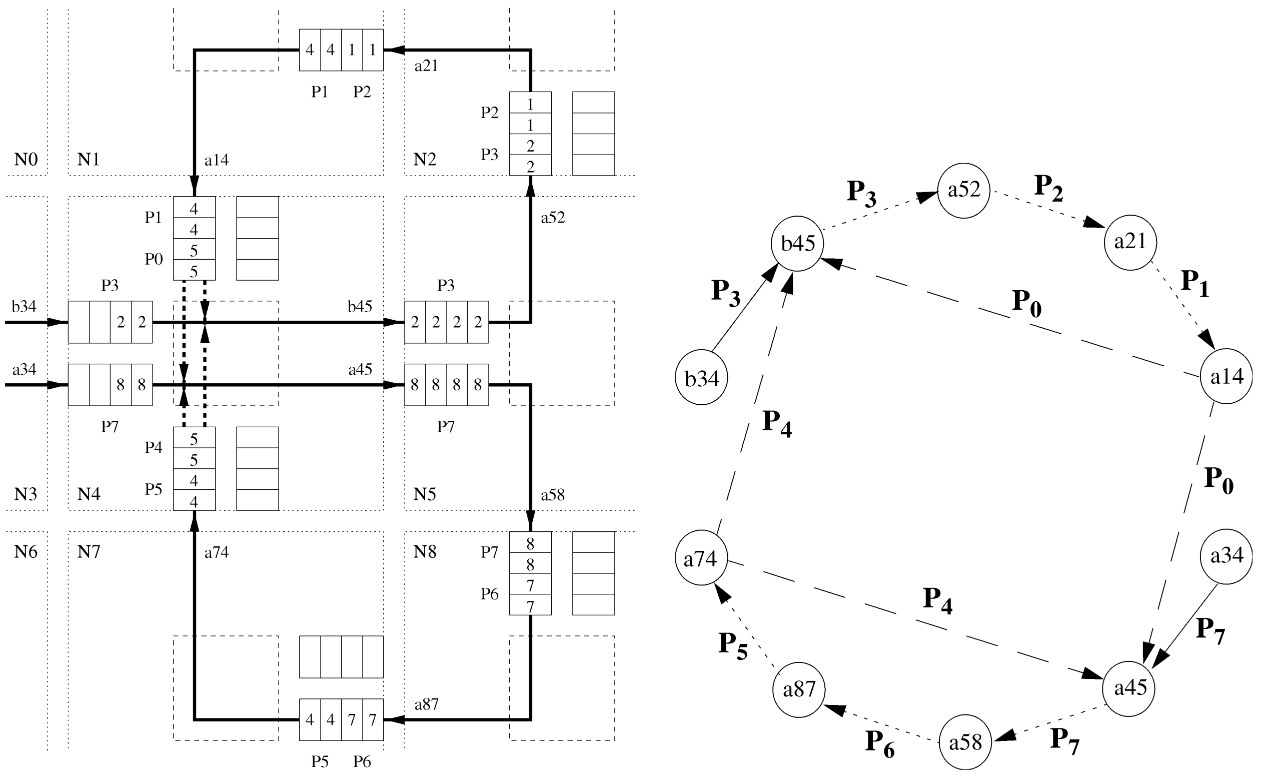


Fig. 4. Deadlocked configuration for a 9-node 2D-mesh and corresponding resource wait-for graph.

arrows indicate routing options for packets whose headers are located at queue heads. Queues contain numbers, indicating the destination node for the corresponding flits. Close to each queue are one or two packet identifiers indicating the packet(s) to which the flits stored in the queue belong. Nodes are delimited by thin dotted lines and are labeled as Ni. Also, channels have been labeled as aij or bij. So, for instance, flits of packet P_1 have been routed through channel a21 but are currently blocked trying to reach the head of the queue associated with channel a14 so that they can be routed to (and sunk at) destination node N4. For the configuration depicted in the figure, packets at the queue heads cannot be routed because all the valid output channels are busy. The remaining packets also cannot be routed since they are blocked behind the tail flits of previous packets and must first reach the queue head in order to be routed. Hence, the configuration is deadlocked.

In the resource wait-for graph shown on the rightmost side of Fig. 4, solid and dashed arcs also indicate the path followed by packet headers and the routing options for packets whose headers are located at queue heads, respectively. Additionally, a dotted arc sinking at a queue indicates that a packet has been allocated queue buffer(s), but has not yet reached the head of the queue. Cycles form in the resource wait-for graph, as shown, indicating that the configuration is effectively deadlocked. Therefore, nonatomic queue allocation disallows the use of Theorem 2 to prove deadlock freedom.

It may be thought that resource wait-for graphs are a better tool than extended resource dependency graphs for the analysis of deadlocked configurations. However, resource wait-for graphs are dynamic in nature, requiring the analysis of all possible configurations that could form in a network in order to find out if any of them is deadlocked. On the other hand, resource dependency graphs are static. They consider all the options offered by the routing function at each router and, therefore, these graphs do not require the analysis of all possible configurations. As a consequence, resource dependency graph analysis should be preferred for practical applications.

When packets are short relative to the size of the queue, removing the constraint on atomic queue allocation may improve throughput considerably. Effectively, packets can start advancing as soon as a single flit buffer is freed instead of waiting until the entire buffer is free. This issue becomes more important as VLSI technology advances. Advances in VLSI technology allow faster clock frequencies and, thus, higher transmission rates. As a consequence, when links are pipelined, deeper buffers are required to tolerate flow control round-trip delay. Therefore, as buffer size increases, buffer utilization decreases unless the constraint on atomic queue allocation is removed. One way to remove this constraint consists of using queues deep enough to store two or more packets and allowing packets to reserve a given queue only when there is enough free buffer space to store the whole packet (effectively, virtual cut-through switching is used). This approach is followed in the Cray T3E router [2]. Although the threat of deadlock is eliminated, queue space is not maximally utilized.

Alternatively, the constraint on atomic queue allocation can be relaxed to allow maximal utilization by guaranteeing that at least one packet in each possible deadlocked configuration is able to use an escape resource. This can be achieved by requiring packets using escape resources to be restricted to continue using escape resources until delivered and by providing access to escape resources from all network queues. This approach is formalized below in a new theorem. But first, the following lemma guarantees that at least one packet in each deadlocked configuration is able to use escape resources, even if the assumption on atomic queue allocation (Assumption 5) is removed.

Lemma 1. *In a deadlocked configuration that can be reached by routing packets starting from an empty network, there is at least one packet whose header flit is at the head of the queue containing it.*

Proof. A deadlock can be due to faulty components, lack of connectivity in the routing function, or the use of a routing function that is not deadlock-free. Deadlocks due to faulty components occur when a packet header is routed and all the output links that can be used at that router to reach the packet destination have failed (alternatively, the routers at the other side of those links have failed). In this case, the packet header must be at the head of the queue containing it. Otherwise, the router cannot route it and discover that all the requested links failed.

Deadlocks due to lack of connectivity in the routing function occur when a packet header is routed and the routing function does not provide any output link for the packet destination. Similarly to the previous case, the packet header must be at the head of the queue containing it. Otherwise, it cannot be routed.

When deadlocks occur due to a routing function that is not deadlock-free, a deadlocked configuration always contains at least one set of packets that are cyclically waiting for resources (queue buffers) occupied by other packets in the set. Let us consider a deadlocked configuration that can be reached by routing packets starting from an empty network. We will proceed by contradiction. *Assume that no packet has its header flit at the head of the queue containing it.* As the configuration is deadlocked, a given header flit in that configuration cannot advance and reach the head of the queue containing it. This implies that it is blocked by flits from another packet. As flits from different packets cannot be interleaved in the same FIFO queue, that packet header must follow the tail flit of another packet, thus being blocked by it. Therefore, the deadlocked configuration consists of a set of packets such that every packet header immediately follows the tail of another packet in the set and both flits (tail from one packet and header from the next one) are stored in the same queue.

As the deadlocked configuration is reachable by routing packets starting from an empty network, let us consider how this deadlocked configuration was reached. When a packet P_1 blocks, the last flit that makes some progress is its tail flit. However, as indicated above, that tail flit is immediately followed by the header flit of another packet P_2. Therefore, that header flit must have arrived at the FIFO queue after the tail flit of P_1. As a consequence, the header flit of P_2 blocks after the tail flit of P_1 blocked. Also, as mentioned above, the tail flit of P_2 will block after the header of P_2 blocked. Following this reasoning, if t_i is the blocking time of the header flit of packet P_i and there are k packets in a cycle of the deadlocked configuration, we have $t_1 < t_2 < ... < t_k$. However, as packets form a cycle, the header of packet P_i blocks after the tail of packet P_k. We conclude that $t_k < t_1$, which is a contradiction. Therefore, the assumption that no packet has its header flit at the head of the queue containing it does not hold, thus proving the lemma.

Alternatively, the lemma is proven as follows: Initially, the network is empty and nodes start injecting packets. When a deadlocked configuration is being formed, the first packet header that blocks must be at the head of a queue. Otherwise, it would have been blocked by the tail flits of a previously blocked packet. □

Here, we restrict our attention to deadlocked configurations that can be reached by routing packets starting from an empty network because we are only interested in real situations. For some network topologies and routing algorithms, there may exist deadlocked configurations that cannot be reached because two packets require the use of the same channel at the same time and this is not possible. Those configurations are not *reachable* [30] and are not considered in this paper.

The following theorem supplies a sufficient condition for deadlock-free routing (avoidance-based or recovery-based) when the assumption on atomic queue allocation (Assumption 5) is removed.

Theorem 3. *A connected and adaptive routing function R for an interconnection network I is deadlock-free if there exists a routing subfunction R_1 that is connected, has no cycles in its resource dependency graph D, and does not allow packets using resources supplied by R_1 to use other resources not supplied by R_1.*

Proof. As R_1 is connected, it is possible to establish a path from any queue in the network to the delivery queue for the packet using only resources supplied by R_1. As R_1 has no cycles in its resource dependency graph D, it is deadlock-free by Theorem 1. Hence, no deadlock can arise from packets using only resources supplied by R_1.

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

1. Some packets occupy resources supplied by R_1. These packets can advance because R_1 is connected and deadlock-free and packets using resources supplied by R_1 are not allowed to use other resources not supplied by R_1. Hence, some packets can advance, contrary to the assumption that there is a deadlocked configuration.
2. No packet occupies resources supplied by R_1. Taking into account Lemma 1, there is at least one packet whose header flit is at the head of the queue containing it. As R_1 is connected and the resources supplied by it are empty, that packet can be routed using resources supplied by R_1 until delivered. So, this packet can advance, contrary to the assumption that there is a deadlocked configuration. □

Theorem 3 is especially interesting for progressive deadlock recovery. Resources can be split into two disjoint sets: adaptive and escape resources. Adaptive resources can be used for fully adaptive routing. Escape resources allow for progressive deadlock recovery of eligible packets. The routing subfunction R_1 can be defined by restricting R to use only escape resources. In other words, routing in escape resources should be done in such a way that it is possible to deliver deadlocked packets from any queue to any destination node. Once a packet is detected as being potentially deadlocked and made eligible to route on the escape resources, it does not require using adaptive resources again. Lemma 1 guarantees that every deadlocked configuration can be recovered even when the assumption on atomic queue allocation is removed.

3.4 Cut-Through Switching

In this section, we informally describe the main differences between virtual cut-through and wormhole switching with respect to deadlock handling. Also, we indicate how the theoretical results presented in Section 3.3 apply to virtual cut-through switching.

The main difference between virtual cut-through and wormhole switching is that flow control is performed at the packet level in virtual cut-through, thus requiring buffer queues deep enough to store one or more packets. However, flits are usually small in wormhole switching and available queue space can be smaller than the packet size. As a consequence, a blocked packet occupies a portion of a single queue in virtual cut-through, but it may span multiple queues in wormhole switching.

From a theoretical point of view, there are two consequences. First, dependencies (see Definition 11) can only exist between adjacent resources, that is, resources located at adjacent routers. Effectively, as blocked packets fit into a single queue and dependencies only arise when a packet is holding one resource and requesting the use of another, it follows that resource dependencies can exist only between adjacent resources. However, the resource dependency graph and the extended resource dependency graph still represent all the direct dependencies for a routing function and all the dependencies for a routing subfunction, respectively. Therefore, Theorems 1 and 2 are also valid for virtual cut-through switching. The only difference with respect to wormhole switching is that some dependencies that exist in a wormhole network may not exist when virtual cut-through switching is used.

The second consequence is that queues can be allocated nonatomically when virtual cut-through switching is used, provided that each queue has capacity for more than one packet. Effectively, buffer space is reserved before starting packet transmission. As a consequence, when a packet is transmitted to another queue, it completely releases the current queue. Thus, the next packet header will occupy the queue head. Therefore, Theorems 1 and 2 are valid for virtual cut-through switching even when queues are not allocated atomically. Obviously, Theorem 3 is also valid for virtual cut-through switching because it imposes more constraints than Theorem 2.

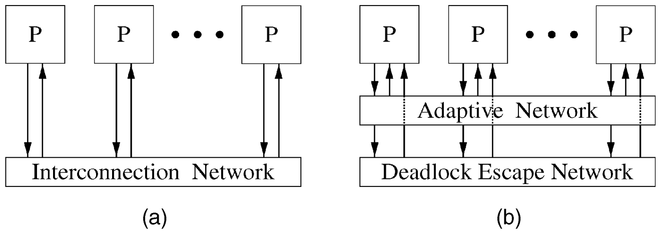


Fig. 5. Informal view for the design methodology: (a) original network and (b) final network.

4 Design Methodology

In this section, we propose a methodology derived from Theorem 3 for the design of deadlock-free fully adaptive routing algorithms. This methodology indicates a way to add resources to an existing network and to derive a new routing function from an initial one. It can be applied to both routing approaches, but is especially useful for the design of progressive recovery-based routing algorithms. It is applicable to routing functions defined on $Q_{IN} \times Q_D$ for which packets remain on escape resources once placed on them. First, we describe the methodology in an informal way. Then, we propose it formally. It should be noted that we could propose a methodology derived from Theorem 2, applicable to both avoidance- and recovery-based routing on a mixed set of resources. In that case, packets would be allowed to come back onto adaptive resources from escape resources, but atomic queue allocation would be required. As this strategy is very similar to that proposed in [19], we will not analyze it in this paper.

The basic idea for the methodology can be seen in Fig. 5. Fig. 5a shows the original deadlock-free network, including the injection and delivery channels for each processor. This network will be used as a deadlock escape network (e.g., progressive deadlock recovery network) in the final network (Fig. 5b). Then, another network is superimposed onto the original. This second network will be used to route packets adaptively. Packets are injected into the adaptive network only, being routed in this network until delivered. In case of potential deadlock, one or more packets are transferred from the adaptive network to the escape (recovery) network, being routed in this network until delivered. It should be noted that the routing function for the original network is not modified at all when used as a deadlock escape network. The only exception is that its domain is modified in such a way that, instead of accepting packets from the injection channels, it now accepts packets traveling on the adaptive network. Also, the adaptive network does not require additional physical links. It can be a virtual network requiring only additional queues. In what follows, we formally present this design methodology.

Methodology 1. *This methodology supplies fully adaptive minimal (alternatively, nonminimal) routing algorithms starting from an initial routing algorithm.*

1. *Given an interconnection network I_1, let Q_I, Q_{N1}, and Q_D be the injection, network, and delivery queues, respectively. Also, $Q_{IN1} = Q_I \cup Q_{N1}$ and $Q_{ND1} = Q_{N1} \cup Q_D$. Define a connected routing*

function $R_1 : Q_{IN1} \times Q_D \rightarrow \mathcal{P}(Q_{ND1})$ for I_1 without cycles in its resource dependency graph. R_1 defines the set of escape resources and will be used to recover from deadlock. It is deadlock-free by Theorem 1.

2. *Add additional edge queues to the network that will serve as fully adaptive routing resources. Let Q_{N2} be the set of adaptive queues in the network. Let Q_{cd} be the set of all the queues in $Q_{ND2} = Q_{N2} \cup Q_D$ belonging to a minimal (alternatively, nonminimal) path from current queue q_c to delivery queue q_d. Define a new routing function $R_2 : Q_{IN2} \times Q_D \rightarrow \mathcal{P}(Q_{ND2})$ as follows:*

$$R_2(q_c, q_d) = Q_{cd} \; \forall q_c \in Q_{IN2}, \; \forall q_d \in Q_D.$$

In general, R_2 is not deadlock-free. It should be noted that R_1 and R_2 are independent. The following step links the two.

3. *Change the domain of R_1 to replace the injection queues with the new queues in Q_{N2} while keeping the range fixed. The expanded routing function $R_1^* : (Q_{N1} \cup Q_{N2}) \times Q_D \rightarrow \mathcal{P}(Q_{ND1})$ is defined as follows:*

$$R_1^*(q_c, q_d) = \begin{cases} R_1(q_c, q_d) & \forall q_c \in Q_{N1} \quad \forall q_d \in Q_D \\ R_1(q_i, q_d) & \forall q_c \in Q_{N2} \quad \forall q_d \in Q_D \\ & \exists q_i \in Q_I \mid \quad d_c = d_i \end{cases}$$

That is, routing is performed as in the original routing function when packet headers are stored at queues in Q_{N1}. When packet headers come from the adaptive network (stored at queues in Q_{N2}), routing is performed as if the packet was injected at the current router. This allows the use of the escape queues in Q_{N1} in case of deadlock.

4. *Once R_1 and R_2 have been linked together, the resulting routing function R is the combination of R_1^* and R_2. That is, the new routing function is minimal (alternatively, nonminimal) fully adaptive. It can use any of the new queues or, alternatively, the queues supplied by R_1^*. If the queues are supplied by R_1^*, the packet will remain on Q_{N1} resources until delivered since R_1^* is restricted in range to supplying (and, therefore, using after the first time) only escape resources. The selection function can be defined in any way. However, to reflect progressive recovery-based routing, it is recommended that the new (adaptive) resources in Q_{N2} be used by normal packets and that the escape resources in Q_{N1} be used only by potentially deadlocked packets.*

Step 1 establishes the escape resources. We can use either a deterministic or a partially adaptive routing function as the basic one for progressively recovering from (or avoiding) deadlock through the escape resources. Although any deadlock-free routing function can be chosen, it is convenient to select a routing function that requires a small amount of resources as these resources will be used only in the rare case of deadlock (for progressive recovery). Routing functions requiring only a few central buffers are especially suitable for this purpose. Step 2 indicates how to add additional edge and/or central queues to the network to provide fully adaptive routing. Step 3 indicates how to extend the original routing function for escape resources so that it also accepts packets (deadlocked packets) from the adaptive network. Step 4 establishes the final routing function.

This methodology does not exploit all the routing flexibility provided by the theorems in Section 3.3. In particular, packets being routed on escape resources are not allowed to use adaptive resources again. However, this methodology is very general. It can be applied to both direct and indirect networks with regular and irregular topologies. It is also valid for wormhole as well as virtual cut-through switching. Moreover, this methodology is very powerful. It is intended to maximize adaptivity and resource utilization while keeping the requirements for deadlock handling at a minimum. Finally, despite the complexity of the formal methodology, its practical application is very simple. As mentioned above, packets are routed adaptively by using adaptive resources until delivered or until a deadlock is detected. In the case of deadlock, deadlocked packets are routed using escape resources until delivered. The following lemma shows that this methodology always supplies deadlock-free routing algorithms; that is, it is always possible to avoid or recover from deadlock.

Lemma 2. *Methodology 1 supplies deadlock-free routing algorithms.*

Proof. According to the definition for R,

$$R_1^*(q_c, q_d) = R(q_c, q_d) \cap Q_{ND1} \quad \forall q_c \in Q_{N1} \cup Q_{N2}, \quad \forall q_d \in Q_D.$$

Thus, according to (2), there exists a subset of queues Q_{ND1} that defines a routing subfunction R_1^* that is connected and has no cycles in its resource dependency graph. Once a packet begins routing on resources supplied by R_1^*, it remains on Q_{ND1} resources. Taking into account Theorem 3, we can conclude that R is deadlock-free. □

5 Design Examples

The application of the proposed design methodology is straightforward. Here, we present some examples of deadlock-free routing algorithms based on progressive recovery for both regular direct and irregular indirect networks which nonatomically use a mixed set of resources. The escape resources used to progressively recover from impending deadlock can be defined on edge queues alone (as in [40]), on both edge and central queues (as in [26]), or on central queues alone (as in [21], [41]). The examples presented below define the escape resources on central queues and allow all edge queues to be used for true fully adaptive routing.

5.1 Application to Regular Direct Networks

Let us first apply the design methodology proposed in Section 4 to k-ary n-cubes. For Step 1, we use acyclic path-based routing over routers, each of which contains two central queues $b_{x,h}$ and $b_{x,l}$ for router x. Let Q_{N1} be the set of central queues (i.e., all the b queues). Router nodes are labeled in sequence in Hamiltonian path order, allowing the central queues in the network to be divided into two

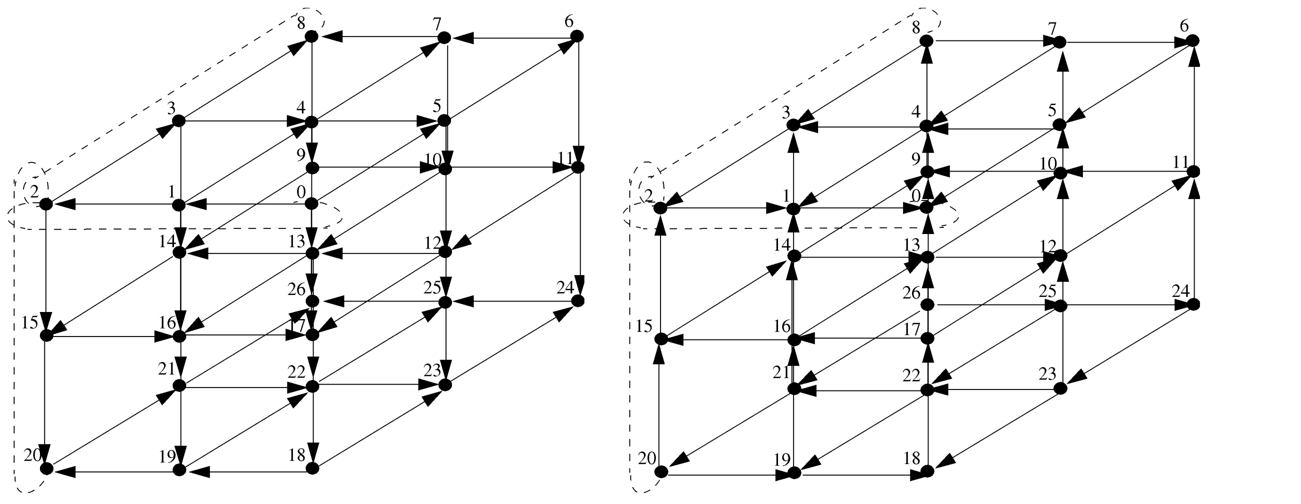


Fig. 6. Shortcuts allowed on the high-queue and low-queue subnetworks of an acyclic path-based routing algorithm on a 3-D torus network. To simplify the figure, wrap-around links for only one node are shown as dotted lines.

subnetworks (as in [42]) on which an adaptive routing function can be defined: a *high-queue subnetwork*, which consists of central queues and paths that can be used by packets having destinations with higher labels than the routers to which the queues belong, and a *low-queue subnetwork*, which consists of central queues and paths that can be used by packets having destinations with lower labels. This routing function is restricted to using non-wrap-around channels to ensure visitation of routers in Hamiltonian order; thus, once a packet routes on a given subnetwork, it remains on that subnetwork until delivered, possibly using nonminimal paths. Fig. 6 illustrates the high-queue and low-queue subnetworks for a 3-ary 3-cube torus network, showing the adaptive short-cuts allowed by the routing function. It is easy to see that this routing function (which will serve as the routing subfunction of the final routing algorithm) is connected, deadlock-free, and partially adaptive, but it is not fully adaptive.

For Step 2, consider that one or more edge queues are now added by splitting each physical channel c_i into v virtual channels, namely $a_{i,1}, a_{i,2}, \ldots, a_{i,v}$. Let Q_{N2} be the set of edge queues (i.e., all the a queues). A fully adaptive routing function can be defined on these queues as follows: Route over any useful dimension using any of the available queues in Q_{N2}, where a useful dimension is one that forwards a packet nearer to its destination.

For Step 3, consider expanding the domain of the routing subfunction while keeping the range fixed by allowing edge queues to inject into the central queues. The selection function determines when this injection should occur, i.e., only when deadlock is detected. For deadlocked packets currently in edge queues, central queues are supplied using the same acyclic path-based ordering rule. That is, assuming the router model presented in Section 2.2, packets in edge queues having destinations with higher labels than the routers to which the queues belong are supplied the high-queue at those routers; packets in edge queues having destinations with lower labels are supplied the low-queue at those routers.

The new routing algorithm obtained after applying all the steps can be stated as follows: Route using any of the a edge queues in true fully adaptive manner. If all of these queues are fully occupied and it is determined that deadlock is impending, route in acyclic path-based ordering using the corresponding b central queues. As indicated in Lemma 2, this routing algorithm is deadlock-free.

5.2 Application to Irregular Indirect Networks

Let us now apply our design methodology to irregular wormhole networks. Shown in Fig. 1 is an irregular indirect network composed of 16 routers and 22 processor nodes. For Step 1, it is possible to construct a breadth-first spanning tree graph for the network on which to base routing using table-lookup (as in [4]). Each router of the graph contains two central queues, $b_{x,up}$ and $b_{x,down}$, for router x. Queues are classified based on the direction of associated paths or links in the graph. In particular, *up* queues are defined to be those associated with paths (links) in a direction upwards toward the root of the spanning tree for routers in different tree levels and/or those queues associated with a link in a direction toward the router that has the lower label for routers in the same tree level. *Down*

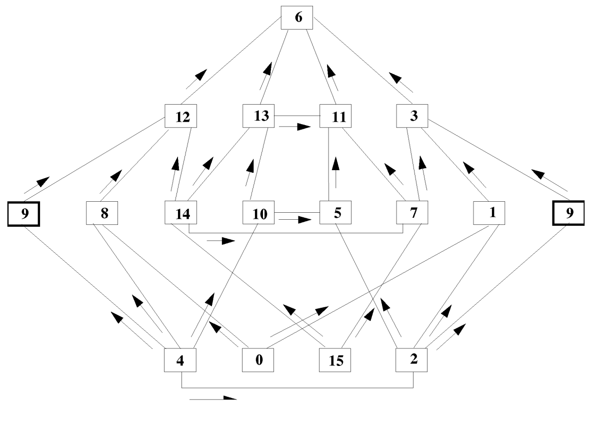


Fig. 7. “Up” link direction assignment for the irregular network.

queues are defined in the opposite manner. Fig. 7 shows the link direction assignment for the irregular network depicted in Fig. 1. Let Q_{N1} be the set of central queues (i.e., all the b queues). An adaptive up/down routing function can be defined on these queues as follows: Packets are routed progressively to their destinations over routes consisting of zero or more up queues followed by zero or more down queues. This routing function is connected and is deadlock-free since dependencies from up queues to down queues exist but not from down queues to up queues. However, this routing function is not fully adaptive and, in some cases, supplies only nonminimal paths.

For Step 2, consider that one or more edge queues are now added by splitting each physical channel c_i into v virtual channels as before. Let Q_{N2} be the set edge queues (i.e., all the a queues). A fully adaptive routing function can be defined on these queues as follows: Route over any useful link using any of the available queues in Q_{N2}, where a useful link is one that forwards a packet nearer to its destination.

For Step 3, consider expanding the domain of the routing subfunction while keeping the range fixed by allowing edge queues to inject into the central queues as determined by the selection function. For packets currently in edge queues, central queues are supplied using the up/down routing rule as originally prescribed. Note that once a packet uses one of the central queues, it must continue using only central queues at subsequent routers until it is delivered at its destination.

The new routing algorithm obtained after applying all the steps can be stated as follows: Route using any of the a edge queues in true fully adaptive manner. All minimal routes are allowed. If all of these queues are fully occupied and it is determined that deadlock is impending, route according to the up/down rule using the corresponding b central queues. As indicated in Lemma 2, this routing algorithm is deadlock-free.

5.3 Discussion

The generality and power of the proposed theory and methodology are illustrated by these examples. Routing algorithms which maximize both routing adaptivity and resource utilization are straightforwardly designed for direct, indirect, regular, and irregular networks. In addition, the resources required by these routing algorithms for handling infrequent deadlocks are kept to a minimum using a mixed set of resources. Below, a number of other interesting observations are made.

Routing subfunctions based on the Hamiltonian path method work for most regular networks and for many irregular networks. For the general case, routing subfunctions based on the up/down routing rule method work for all networks. The additional routing freedom provided by the fully adaptive (deadlock susceptible) network significantly increases routing options presented to packets, providing many more minimal paths not supported by the original (deadlock escape) network. For example, only one legal minimal path exists between Router 3 and Router 5 in Fig. 7 using up/down routing [4] (i.e., path 3-6-11-5), whereas four legal minimal paths exist using the new routing algorithm (i.e., paths 3-6-11-5, 3-7-11-5, 3-1-2-5, and 3-9-2-5).Moreover, for some node pairs, paths that are more minimal than those allowed by up/down routing are made legal by the new routing algorithm. For instance, in routing from Router 1 to Router 8, path 1-0-8 is made possible as opposed to path 1-3-6-12-8. Finally, it is noted that the selection function which defines which legal resource should be allocated reflects the way in which deadlocks are handled, either by avoidance or recovery. Although always supplied by the routing function, escape resources should be selected only in the rare cases when impending deadlock is detected for deadlock recovery routing.

6 Related Work

Most theories on deadlock avoidance focus on direct networks [7], [19], [21], [25], [37], [26], [43], [44], [45], [46]. Indirect networks have not been considered because the most popular indirect networks in the past (i.e., crossbars and unidirectional multistage interconnection networks) do not suffer from deadlock under unicast routing [30]. However, the use of bidirectional links in multistage networks and arbitrary (possibly irregular) topologies implemented in some networked-computing (e.g., NOW) systems may introduce deadlocks. Although these particular cases have been analyzed by several researchers [4], [14], [15], [47], [48], no general theory has been proposed for direct and indirect networks. The only exception is the unified view proposed in [30]. Although it is not a general theory, that work explicitly extends the theories for direct networks to indirect networks in an informal way. Similarly, most theories focus on regular topologies. Although most of them are also valid for irregular topologies, the particular issues arising in these networks have been analyzed separately [4], [14], [15], [48].

The particular requirements for each switching technique in order to avoid deadlock have been analyzed separately. Most research work deals with either packet switching [37], [49], [50], [43] or wormhole switching [19], [21], [25], [26], [44], [46], [51]. Results for packet switching are also valid for virtual cut-through switching but not for wormhole switching. Most theories for adaptive wormhole routing assume atomic queue allocation, preventing the sharing of queues by multiple packets [19], [25], [26], [51]. Results for other switching techniques exist, like pipelined circuit switching [52] and mad postman [53]. To the best of our knowledge, the only unified theories on deadlock avoidance for both packet (virtual cut-through) and wormhole switching are presented in [30], [45]. In addition, previous theories focus either on edge or central queues. To the best of our knowledge, the only exceptions are the theories presented in [37] and [26]. These theories support a mixed set of resources.

In general, deadlock avoidance strategies are valid for both minimal and nonminimal routing. However, regressive deadlock recovery routing strategies rely on minimal routing to detect deadlocks at source nodes to simplify the task of message retransmission [54], [55]. Moreover, deflective deadlock recovery routing strategies rely on nonminimal routing in order to deflect deadlocked packets into "holes" (packet-sized empty queue space) which propagate in the network to allow deadlocks to drain [22], [56], [57]. Finally, deadlock avoidance and recovery have been traditionally considered as completely different deadlock handling strategies. Only very recently has it been shown that deadlock avoidance and progressive deadlock recovery can be integrated into a single theoretical framework [26]. However, that work was applied only to meshes and tori.

7 Conclusion

This paper presents a theoretical framework for the design of deadlock-free fully adaptive routing algorithms for all classes of topologies and all types of switching techniques in a single, unified theory. A general formal theory is proposed that is applicable to direct and indirect networks as well as regular and irregular networks. It allows the design of deadlock avoidance-based as well as deadlock recovery-based wormhole and virtual cut-through adaptive routing algorithms that use a homogeneous or a heterogeneous (mixed) set of resources. Additionally, the theory relaxes the atomic queue allocation requirement of previous theories, allowing resources to be used more efficiently. To simplify the application of the theory, we also propose a general methodology applicable to arbitrary network topologies for the design of fully adaptive routing algorithms which maximize routing flexibility and resource utilization. The proposed theory and methodology allow the design of efficient network routers that require minimal resources for handling infrequent deadlocks.

Acknowledgments

The research described in this paper was supported in part by US National Science Foundation (NSF) Career Award grant ECS-9624251, and NSF grant CCR-9812137 and by the Spanish CICYT grant TIC97-0897-C04-01.

References

[1] C.B. Stunkel et al., "The SP2 High-Performance Switch," *IBM Systems J.*, vol. 34, no. 2, pp. 185-204, 1995.

[2] S.L. Scott and G.M. Thorson, "The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus," *Proc. Symp. Hot Interconnects IV*, pp. 147-156, Aug. 1996.

[3] J. Laudon and D. Lenoski, "The SGI Origin: A ccNUMA Highly Scalable Server," *Proc. 24th Int'l Symp. Computer Architecture*, pp. 241-251, June 1997.

[4] M.D. Schroeder et al., "Autonet: A High-Speed, Self-Configuring Local Area Network Using Point-to-Point Links," Technical Report SRC Research Report 59 (DEC), Apr. 1990.

[5] N. J. Boden, D. Cohen, R.E. Felderman, A.E. Dulawik, C.L. Seitz, J. Seizovic, and W. Su, "Myrinet—A Gigabit per Second Local Area Network," *IEEE Micro*, vol. 15, no. 1, pp. 29-36, Feb. 1995.

[6] D. Garcia and W. Watson, "ServerNet II," *Proc. Second Parallel Computer Routing Comm. Workshop*, pp. 119-135, June 1997.

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

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

[9] R. Horst, "ServerNet Deadlock Avoidance and Fractahedral Topologies," *Proc. Int'l Parallel Processing Symp*, pp. 275-280, Apr. 1996.

[10] J. Carbonaro, "Cavallino: The Teraflops Router and NIC," *Proc. Symp. Hot Interconnects IV*, pp. 157-160, Aug. 1996.

[11] M. Galles, "Spider: A High Speed Network Interconnect," *Proc. Symp. Hot Interconnects IV*, pp. 141-146, Aug. 1996.

[12] S.L. Scott and J. Goodman, "The Impact of Pipelined Channels on k-Ary n-Cube Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, no. 1, pp. 1-16, Jan. 1994.

[13] W. Dally, "Virtual Channel Flow Control," *IEEE Trans. Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194-205, Mar. 1992.

[14] F. Silla, M.P. Malumbres, A. Robles, P. Lopez, and J. Duato, "Efficient Adaptive Routing in Networks of Workstations with Irregular Topogy," *Proc. Workshop Comm., Architecture, and Applications for Network-Based Parallel Computing*, Feb. 1997.

[15] F. Silla and J. Duato, "On the Use of Virtual Channels in Networks of Workstations with Irregular Topology," *Proc. Second Parallel Computer Routing Comm. Workshop*, pp. 203-216, June 1997.

[16] S. Konstantinidou and L. Snyder, "Chaos Router: Architecture and Performance," *Proc. 18th Int'l Symp. Computer Architecture*, pp. 212-221, May 1991.

[17] D. Linder and J. Harden, "An Adaptive and Fault Tolerant Wormhole Routing Strategy for k-Ary n-Cubes," *IEEE Trans. Computers*, vol. 40, no. 1, pp. 2-12, Jan. 1991.

[18] W. Dally and H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks using Virtual Channels," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 4, pp. 466-475, Apr. 1993.

[19] J. Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 12, pp. 1320-1331, Dec. 1993.

[20] J. Kim, Z. Liu, and A. Chien, "Compressionless Routing: A Framework for Adaptive and Fault-Tolerant Routing," *Proc. 21st Int'l Symp. Computer Architecture*, pp. 289-300, Apr. 1994.

[21] Anjan K.V. and T.M. Pinkston, "An Efficient, Fully Adaptive Deadlock Recovery Scheme: DISHA," *Proc. 22nd Int'l Symp. Computer Architecture*, pp. 201-210, June 1995.

[22] P. Palazzari and M. Coli, "Virtual Cut-Through Implementation of the Hole-Based Packet Switching Routing Algorithm," *Proc. Sixth Euromicro Workshop Parallel and Distributed Processing*, pp. 416-421, Jan. 1998.

[23] W. Dally, L. Dennison, D. Harris, K. Kan, and T. Zanthopoulos, "Architecture and Implementation of the Reliable Router," *Proc. Hot Interconnects II Symp.*, Aug. 1994.

[24] T.M. Pinkston, Y. Choi, and M. Raksapatcharawong, "Architecture and Optoelectronic Implementation of the WARRP Router," *Proc. Fifth Symp. Hot Interconnects*, pp. 181-189, Aug. 1997.

[25] J. Duato, "A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 6, no. 10, pp. 1055-1067, Oct. 1995.

[26] Anjan K.V., T.M. Pinkston, and J. Duato, "Generalized Theory for Deadlock-Free Adaptive Wormhole Routing and Its Application to Disha Concurrent," *Proc. 10th Int'l Parallel Processing Symp*, pp. 815-821, Apr. 1996.

[27] T.M. Pinkston, "Flexible and Efficient Routing Based on Progressive Deadlock Recovery," *IEEE Trans. Computers*, vol. 48, no. 7, pp. 649-669, July 1999.

[28] S. Scott and G. Thorson, "Optimized Routing in the Cray T3D," *Proc. Workshop Parallel Computer Routing and Comm.*, pp. 281-294, May 1994.

[29] L.-S. Peh and W. Dally, "A Delay Model for Router Microarchitectures," *IEEE Micro*, vol. 21, no. 1, pp. 26-34, Jan./Feb. 2001.

[30] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach.* Los Alamitos, Calif.: IEEE CS Press, 1997.

[31] S. Warnakulasuriya and T.M. Pinkston, "Characterization of Deadlocks in Irregular Networks," *Proc. 1999 Int'l Conf. Parallel Processing*, pp. 75-84, Sept. 1999.

[32] S. Warnakulasuriya and T.M. Pinkston, "Characterization of Deadlocks in k-Ary n-Cube Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 10, no. 9, pp. 904-921, Sept. 1999.

[33] P. Lopez, J.M. Martinez, and J. Duato, "A Very Efficient Distributed Deadlock Detection Mechanism for Wormhole Networks," *Proc. High Performance Computer Architecture Symp.*, pp. 57-66, Feb. 1998.

[34] T.M. Pinkston and S. Warnakulasuriya, "On Deadlocks in Interconnection Networks," *Proc. 24th Int'l Symp. Computer Architecture*, pp. 38-49, June 1997.

[35] J.M. Martinez, P. Lopez, J. Duato, and T.M. Pinkston, "Software-Based Deadlock Recovery for True Fully Adaptive Routing in Wormhole Networks," *Proc. 1997 Int'l Conf. Parallel Processing*, pp. 182-189, Aug. 1997.

[36] Y. Tamir and G. Frazier, "Dynamically-Allocated Multi-Queue Buffers for VLSI Communication Switches," *IEEE Trans. Computers,* vol. 41, no. 6, pp. 725-734, June 1990.

[37] J. Duato, "A Necessary and Sufficient Condition for Deadlock-Free Routing in Cut-Through and Store-and-Forward Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 7, no. 8, pp. 841-854, Aug. 1996.

[38] S. Warnakulasuriya and T.M. Pinkston, "A Formal Model of Message Blocking and Deadlock Resolution in Interconnection Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 11, no. 3, pp. 212-229, Mar. 2000.

[39] J. Duato, "A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks," *Proc. 1994 Int'l Conf. Parallel Processing,* vol. 1, pp. 142-149, Aug. 1994.

[40] F. Petrini and M. Vanneschi, "Performance Analysis of Minimal Adaptive Wormhole Routing with Time-Dependent Deadlock Recovery," *Proc. 11th Int'l Parallel Processing Symp.,* pp. 589-595, Apr. 1997.

[41] Anjan K.V. and T.M. Pinkston, "DISHA: A Deadlock Recovery Scheme for Fully Adaptive Routing," *Proc. Ninth Int'l Parallel Processing Symp.,* pp. 537-543, Apr. 1995.

[42] X. Lin and L.M. Ni, "Multicast Communication in Multicomputer Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 4, no. 10, pp. 1104-1117, Oct. 1993.

[43] G.D. Pifarre et al., "Fully Adaptive Minimal Deadlock-Free Packet Routing in Hypercubes, Meshes, and Other Networks: Algorithms and Simulations," *IEEE Trans. Parallel and Distributed Systems,* vol. 5, no. 3, pp. 247-263, Mar. 1994.

[44] X. Lin, P.K. McKinley, and L.M. Ni, "The Message Flow Model for Routing in Wormhole-Routed Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 6, no. 7, pp. 755-760, July 1995.

[45] L. Schwiebert and D.N. Jayasimha, "A Universal Proof Technique for Deadlock-Free Routing in Interconnection Networks," *Proc. Symp. Parallel Algorithms and Architecture,* pp. 175-184, July 1995.

[46] L. Schwiebert and D.N. Jayasimha, "A Necessary and Sufficient Condition for Deadlock-Free Wormhole Routing," *J. Parallel and Distributed Computing,* vol. 32, no. 1, pp. 103-117, Jan. 1996.

[47] B. Abali, "A Deadlock Avoidance Method for Computer Networks," *Proc. Workshop Comm. and Architectural Support for Network-Based Parallel Computing,* pp. 61-72, Feb. 1997.

[48] W. Qiao and L.M. Ni, "Adaptive Routing in Irregular Networks Using Cut-Through Switches," *Proc. 1996 Int'l Conf. Parallel Processing,* pp. 52-60, Aug. 1996.

[49] P.M. Merlin and P.J. Schweitzer, "Deadlock Avoidance in Store-and-Forward Networks—I: Store-and-Forward Deadlock," *IEEE Trans. Comm.,* vol. 3, pp. 345-354, Mar. 1980.

[50] K.D. Gunther, "Prevention of Deadlocks in Packet-Switched Data Transport Systems," *IEEE Trans. Comm.,* vol. 4, pp. 512-524, Apr. 1981.

[51] E. Fleury and P. Fraigniaud, "A General Theory for Deadlock Avoidance in Wormhole-Routed Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 9, no. 7, pp. 626-638, July 1998.

[52] P.T. Gaughan and S. Yalamanchili, "A Family of Fault-Tolerant Routing Protocols for Direct Multiprocessor Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 6, no. 5, pp. 482-497, May 1995.

[53] C.R. Jesshope, P.R. Miller, and J.T. Yantchev, "High Performance Communications in Processor Networks," *Proc. 16th Int'l Symp. Computer Architecture,* pp. 150-157, May 1989.

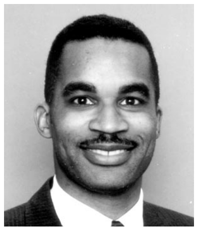
[54] D.S. Reeves, E.F. Gehringer, and A. Chandiramani, "Adaptive Routing and Deadlock Recovery: A Simulation Study," *Proc. Fourth Conf. Hypercube Concurrent Computers and Applications,* Mar. 1989.

[55] J. Kim, Z. Liu, and A. Chien, "Compressionless Routing: A Framework for Adaptive and Fault-Tolerant Routing," *IEEE Trans. Parallel and Distributed Systems,* vol. 8, no. 3, pp. 229-244, Mar. 1997.

[56] M. Coli and P. Palazzari, "An Adaptive Deadlock and Livelock Free Routing Algorithm," *Proc. Third Euromicro Workshop Parallel and Distributed Processing,* pp. 288-295, Jan. 1995.

[57] C. Carrion, R. Beivide, J.A. Gregorio, and F. Vallejo, "A Flow Control Mechanism to Prevent Message Deadlock in k-Ary n-Cube Networks," *Proc. 1997 Int'l Conf. High Performance Computing,* Dec. 1997.

José Duato received the MS and PhD degrees in electrical engineering from the Technical University of Valencia, Spain, in 1981 and 1985, respectively. He is currently a professor in the Department of Computer Engineering (DISCA), Technical University of Valencia (Universidad Politecnica de Valencia), Spain, and an adjunct professor in the Department of Computer and Information Science, The Ohio State University. He is currently researching multiprocessor systems, networks of workstations, interconnection networks, and multimedia systems. His theory on deadlock-free adaptive routing has been used in the design of the routing algorithms for the MIT Reliable Router, the Cray T3E router, and the router embedded in the new Alpha 21364 microprocessor. He coauthored *Interconnection Networks: An Engineering Approach* with S. Yalamanchili and L.M. Ni (IEEE CS Press). Dr. Duato served as an associate editor of the *IEEE Transactions on Parallel and Distributed Systems* from 1995 to 1997. He is currently serving as an associate editor of the *IEEE Transactions on Computers*. Also, he has been or is a member of the program committee for several major conferences (ICPADS, ICDCS, Europar, HPCA, ICPP, MPPOI, HiPC, PDCS, ISCA, IPPS/SPDP, ISPAN). He is the general cochair for International Conference on Parallel Processing 2001. He is a member of the IEEE and the IEEE Computer Society.



Timothy Mark Pinkston completed the BSEE degree from The Ohio State University in 1985 and the MS and PhD degrees in electrical engineering from Stanford University in 1986 and 1993, respectively. Prior to joining the University of Southern California (USC) in 1993, he was a member of the technical staff at Bell Laboratories, a Hughes Doctoral Fellow at Hughes Research Laboratory, and a visiting researcher at IBM T.J. Watson Research Laboratory. Presently, Dr. Pinkston is an associate professor in the Computer Engineering Division of the EE-Systems Department at USC and heads the SMART Interconnects Group. His current research interests include the development of deadlock-free adaptive routing techniques and optoelectronic network router architectures for achieving high-performance communication in parallel computer systems—massively parallel processor (MPP) and network-based (NOW) computing systems. Dr. Pinkston has authored more than 50 refereed technical papers and has received numerous awards, including the Zumberge Fellow Award, the National Science Foundation (NSF) Research Initiation Award, and the NSF Career Award. He has been a member of the program committee for several major conferences (ISCA, ICPP, IPPS/IPDPS, ICDCS, SC, CAC, PCRCW, OC, MPPOI, IEEE LEOS, WOCS), the program cochair for the International Conference on Massively Parallel Processing Using Optical Interconnections (MPPOI '97), and the finance chair for Cluster 2001. Currently, he serves is an associate editor for the *IEEE Transactions on Parallel and Distributed Systems*. Dr. Pinkston is a member of the ACM, a senior member of the IEEE, and a member of the IEEE Computer Society.