Ghost Packets: A Deadlock-free Solution for \( k \)-ary \( n \)-cube Networks

C. Carrión, C. Izu\textsuperscript{\dagger}, J.A. Gregorio, F. Vallejo, R. Beivide\textsuperscript{*}

Dpt. Electrónica y Computadores
Universidad de Cantabria
39005 Santander, Spain

Dpt. Computer Science\textsuperscript{\dagger}
University of Adelaide
Adelaide, SA 5005, Australia

Abstract

Improving interconnection subsystems is crucial for the overall performance of a multicomputer system. Hence, a theoretical presentation of a new deadlock-free message flow model for \( k \)-ary \( n \)-cube networks is developed in this paper. The key idea of this flow control mechanism is to preserve enough free resources for each possible routing dependency cycle, so that packet progress will be guaranteed. Based on this algorithm, we have proposed a simple router structure for a 2-ary \( n \)-cube topology with dimensional order routing. Edge or shared buffering can be used, requiring a minimum capacity of one packet per channel. Virtual channels are eliminated, reducing router complexity and, consequently, decreasing network latency at low loads. In fact, the performance evaluation for the 2-ary \( n \)-cube with different loads shows an improvement in the latency parameter of about 20\% with respect to a deterministic routing with two virtual channels.

1 Introduction

Routers are essential building blocks for present and future computer and communication systems. Thus, routers for direct interconnection networks have to make efficient use of all resources providing powerful communication services at all load conditions. One of the most popular interconnection networks is the \( k \)-ary \( n \)-cube. Basic node parameters such as the routing and the flow control have to be carefully designed [8] [12] to avoid anomalies such as deadlock, starvation, etc.

Many different flow control algorithms have been proposed in the literature [1] [4]. A survey of wormhole routing algorithms can be found in [13]. Other routing algorithms are specifically designed for cut-through flow control [11]. They require larger buffer capacity which increases the node delay. Notwithstanding, as technology advances channel propagation delay is becoming the bottleneck [2] so cut-through routers may be the choice for the future.

All these algorithms provide a special mechanism in order to avoid deadlock. The technique employed is based on the elimination of cyclic dependencies between the resources, [9] [7] by means of virtual channels. There are many router implementations in the literature such as, the Torus Routing Chip [8] and the CrayT3E [14] that use this approach. Virtual channels introduce non-uniformities by the addition of underutilized buffers [3]. Some adaptive techniques deal with non-uniformities by tuning the assignment of virtual channels to packets so that the network load makes a more balanced use of the buffer resources [6].

Nevertheless, the use of both virtual channels and adaptive algorithms increases hardware costs [5]. In fact, most research papers in the last years have focused on obtaining deadlock-free routing functions with minimal number of virtual channels [7]. Thus, our goal of studying new techniques to avoid deadlock without the necessity of virtual channels.

The aim of this paper is the development of a new deadlock-free message flow control model for \( k \)-ary \( n \)-cube networks. The key to avoid cyclic dependencies is based on the use of a special tag, the ghost packet, that guarantees the existence of a free buffer in each cycle. The hardware cost of the router is reduced, eluding the necessity of virtual channels. Due to the router simplicity, the sustained throughput at saturation point and the latency of the messages improved when compared to that of the router proposed by Dally\&Seitz [9].

The outline of this paper is as follows. Section 2 includes a set of assumptions and basic definitions that clarify the rest of the presentation. In section 3 a formal description of the new flow control is presented. The performance evaluation of our proposal for a 2D torus, and the comparison with the results obtained using virtual channels are shown in section 4. Finally, in section 5, results are assessed and conclusions drawn.

\textsuperscript{*} This work has been done with support of the Spain CICYT under grant TIC95-0378.
2 Preliminaries

First of all, it is necessary to introduce some assumptions and definitions helping us to understand the new proposal.

2.1 Assumptions.

The message flow fulfills some initial assumptions on their advance through the network. We assume the following:

- **Injection**: A node can generate messages of arbitrary length and be destined for any other network node. The messages can be produced at any rate, depending on the available source queue space. Messages must be split into fixed length packets.

- **Consumption**: A packet arriving at its destination node, \( n_d \), is eventually consumed.

- **Storage Capacity**: The buffers can be allocated to either the input or output channels with a minimum capacity of one packet. We have chosen input queueing as a trade-off between performance and implementation [10].

- The **Routing Function** depends on the current and destination nodes. The header flit of every message carries the information required to route the message from a queue, \( q_i \), to \( q_j \). The suffix \( i \) is a 2-tuple \( i = (c, d) \) which identifies a single queue by its coordinate, \( c \), and its dimension \( d \). In order to prevent starvation, the forwarding selection among messages routed to the same queue follows a round-robin policy.

2.2 Definitions

Although most of the following definitions are given in [7], [9], [13] we also introduce them here for completeness.

**Definition 1** The topology of a direct interconnection network is modeled as a strongly connected directed graph, \( I = G(N, Q) \), where the vertices of the graph, \( N \), represent the processing nodes and the edges of the graph, \( Q \), represent the communication channel queues.

The topology analyzed in this paper is the bidirectional \( k \)-ary \( n \)-cube. It consists on \( N = k^n \) nodes, each of them labeled by an \( n \)-tuple \( W(w_0, w_1, ..., w_{n-1}) \), \( 0 \leq w_i < k \) \( \forall i \in \{0, 1, 2, ..., (n - 1)\} \) with values corresponding to the node’s offset in each dimension with respect to node \((0, 0, 0, ..., 0)\). The connectivity in dimension \( i \) for this node is:

\[
W \rightarrow \{ \begin{align*}
(w_0, w_1, ..., (w_i + 1) \mod n, ..., w_{n-2}, w_{n-1}) \\
(w_0, w_1, ..., (w_i - 1) \mod n, ..., w_{n-2}, w_{n-1})
\end{align*}
\]

Each channel has associated a queue \( q_i \in Q \) of a capacity \( \text{cap}(q_i) \) packets and a current occupancy of \( \text{occu}(q_i) \leq \text{cap}(q_i) \) packets.

**Definition 2** A Routing Function \( R: Q \times N \rightarrow P(Q) \) supplies a set of alternative queues, \( P(Q) \), to send a packet from the current queue, \( Q \), towards the destination node, \( n_d \).

The deterministic dimensional order routing function, \( R_{DOR} \), returns a set \( P(Q) \) which has only one element. For a \( k \)-ary \( n \)-cube interconnection network \( R_{DOR} \) is formally defined as

\[
R_{DOR}(q_i, n_d) = q_j
\]

\[
\begin{align*}
& j = (c_i, d_i + 1) \quad \text{if } q_i \in n_d \\
& j = ((c_i \pm 1) \mod k, d_i) \quad \text{if } c_d \in \text{dim } d_i \neq c_i \\
& j = ((c_i \pm 1) \mod k, d_i - 1) \quad \text{if } c_d \in \text{dim } d_i = c_i
\end{align*}
\]

(1)

**Definition 3** A Flow Control Function \( F: Q \times R \times S \rightarrow \{\text{True}, \text{False}\} \) enables or disables the advance of a packet from the input queue, \( Q \), to the next queue selected by the routing function, \( R \), depending on the state, \( S \), of the destination queue. The state of a queue \( q \) denoted by \( S(q) \), has two possible values \( S(q) = \{\text{Busy}, \text{Free}\} \) which reflect the occupation of the queue.

With cut-through flow control, a packet located at queue \( q_i \) will be allowed to move to queue \( q_j \in R(q_i, n_d) \) if the function \( F \) verifies:

\[
F_{CT}(q_i, q_j, S(q_i)) = \text{True if } S(q_j) = \text{Free} \quad (2)
\]

The queue, \( q_j \), is busy if there is no room for another packet, \( \text{occu}(q) = \text{cap}(q) \), and is free otherwise. Thus, we can rewrite the previous equation as:

\[
F_{CT}(q_i, q_j, S(q_j)) = \text{True if } \text{occu}(q_j) < \text{cap}(q_j) \quad (3)
\]

**Definition 4** A Deadlocked configuration for a given interconnection network, \( I \), and routing function, \( R \), is a nonempty valid assignment of packets to each queue, verifying the following condition:

\[
\exists D_C \subset Q \mid \forall q_i \in D_C \text{ with } \text{cap}(q_i) > 0
\]
\[
\begin{align*}
R(q_i, n_d) & \neq q_d \text{ with } q_d \in n_d \\
cap(q_j) & = \text{occu}(q_j) \forall q_j \in R(q_i, n_d)
\end{align*}
\]

**Definition 5** A queue dependency graph, \( D \), for a given interconnection network, \( I \), and routing function, \( R \), is a directed graph, \( D = G(Q, E) \) where the queues, \( Q \), of the network are the vertices and the pairs of queues \((q_i, q_j)\), such that \( R(q_i, n_d) = q_j \) are the arcs, \( E \).

Many deadlock free algorithms for \( k \)-ary \( n \)-cube networks have been proposed based on Dally’s theorem [9] and Duato’s formal extension for adaptive routing [7]. Dally’s theorem asserts that a routing function \( R \) is deadlock free for an interconnection network iff there are no cycles in the queue dependency graph.

Hence, these deadlock algorithms [8] [14] break the cyclic resource dependencies by splitting the physical channel into several virtual channels which then share the physical channel, ie, packets are multiplexed over the link. Chien shows in [5] that using virtual channels increases both node delay and router cost. Thus, avoiding deadlock by using this technique has a negative influence on the final performance of the interconnection network.

Based on this knowledge we can assert that every acyclic dependency graph is deadlock free. Therefore, we will focus our attention on avoiding deadlock when cyclic dependencies occur as this is the case in a torus network under dimensional order routing [8]. So on the whole, we propose the use of ghost packets instead of virtual channels as a way to avoid deadlock in \( k \)-ary \( n \)-cube networks.

### 3 Ghost Packets and Ghost-CT flow control

This section presents a formal description of our deadlock-free message flow proposal for \( k \)-ary \( n \)-cube networks and the proof of its deadlock freedom.

Let us denote by \( D_C \) the set of queues involved in a cyclic dependency. Then, considering the deadlock definition given in section 2, it is easy to understand that the network will be deadlock free if the condition \( \sum_i \text{occu}(q_i) < \sum_i \text{cap}(q_i) \forall q_i \in D_C \) is always fulfilled for all sets of cyclic dependencies. This condition can be rewritten as \( \exists q_i \in D_C \) \( \text{occu}(q_i) < \text{cap}(q_i) \). Thus, deadlock freedom can be guaranteed if there is at least one buffer for every cyclic which cannot be exhausted.

We have introduced a new queue state called \( G_{\text{busy}} \) with peculiar properties which help us to achieve the above condition for the deterministic \( k \)-ary \( n \)-cube network. This state is characterized by a special occupancy function that depends on the relative dimension from which we are looking at.

Then, a queue can be in three states \( S(q_j) = \{ \text{Free, Busy, G_{busy}} \} \) and the occupation of the queue \( q_j \), for a given packet \( p \) located at \( q_i \), is:
\[
\text{occu}^*(q_j, q_i) =
\begin{cases}
\text{occu}(q_j) + 1 & \text{if } S(q_j) = G_{\text{busy}} \text{ and } d_i \neq d_j \\
\text{occu}(q_j) & \text{otherwise}
\end{cases}
\]

(4)

where \( i = (c_i, d_i) \), \( j = (c_j, d_j) \) denote the coordinate and dimension of the queues \( q_i \) and \( q_j \).

The new state behaves as a ghost packet which is visible from outside the dimensional ring, but has no blocking effect for packets within that dimensional ring. So, as long as one queue is in state \( G_{\text{busy}} \) there is no deadlock. From now on, we will use the term ghost packet to refer the buffer in \( G_{\text{busy}} \) state reserved for transit packets as this concept is more intuitive than the buffer state.

The flow control function is similar to the \( FCT \), equation 3, but using the new occupancy function, \( \text{occu}^* \).
\[
F_{G_{\text{CT}}}(q_i, q_j, S(q_j)) = \begin{cases}
\text{True} & \text{if } \text{occu}^*(q_j, q_i) < \text{cap}(q_j) \\
\text{False} & \text{otherwise}
\end{cases}
\]

(5)

If the flow control function is asserted, the packet is forwarded from \( q_i \) to \( q_j \). Consequently, the occupancy and the state of the two queues are modified. A formal description of the next states for each queue after a packet is forwarded is as follows:
\[
S_{\text{next}}(q_j) =
\begin{cases}
\text{Busy} & \text{if } \text{occu}^*(q_j, q_i) = \text{cap}(q_j) - 1 \\
S(q_j) & \text{otherwise}
\end{cases}
\]

(6)

\[
S_{\text{next}}(q_i) =
\begin{cases}
G_{\text{busy}} & \text{if } d_i = d_j \text{ and } S(q_j) = G_{\text{busy}} \text{ and } \\
\text{occu}^*(q_j, q_i) = \text{cap}(q_j) - 1 \\
\text{Free} & \text{otherwise}
\end{cases}
\]

(7)

This implies that if the packet moving in the same dimension reaches a queue \( q_j \), which only free buffer has a ghost packet, the forwarded packet is exchanged with the ghost packet.

### 3.1 Ghost Packet’s Lack of Mobility

Ghost packets break any unidimensional cyclic dependency queue but they may introduce a new blocking situation. They only move backwards when there is a exchange, so if transit packets do not use the
queue occupied by a ghost packet, that ghost packet will not move. This may prevent other queues at that router node from injecting a new packet into that dimensional ring indefinitely. This situation is illustrated in Figure 1. The input of new packets into the Y rings is locked by the position of the ghost packets even though all the resources of the Y rings are Free.

A simple solution is to force (if possible) the ghost packet to move backwards when there is an external request to access its buffer. In the above example, this will mean transferring the ghost packet to the previous free queue in its own dimension.

Let’s denote $q_b$ the previous queue in the same dimension of $q_j$, $q_b = q_{((c_j)\pm 1)\text{mod}k,d_j}$. Then, if $S(q_b) = \text{Free}$ it is possible for queues $q_j$ and $q_b$ to exchange status. The signal that triggers this state transition is defined as

$$
\text{Swap}_S(q_j,q_b) = (F_{G,L,T}(q_j,q_j,S(q_j)) = \text{False}) \quad \text{and} \quad (S(q_j) = G_{\text{busy}} \text{ and } S(q_b) = \text{Free})
$$

(8)

Then, if $\text{Swap}_S(q_j,q_b) = \text{True}$ and there are no other requests for $q_b$, the state transition will occur, $S_{\text{next}}(q_j) = \text{Free}$ and $S_{\text{next}}(q_b) = G_{\text{busy}}$. After that transition, $F_{G,L,T}(q_j,q_j,S(q_j))$ will become true and the packet will be accepted into the ring.

This new state transition resolves the blockage caused by a still ghost packet. In other words, when a ghost packet is unnecessarily preventing packets from entering into its ring, $(F_{G,L,T}() = \text{False})$, a request to move back that ghost packet will be made and eventually, the buffer will be free and new packets will enter the ring.

3.2 Deadlock Freedom

**Theorem 1** A k-ary n-cube network with a routing function, $R_{DOR}(q_i,n_d)$, a flow control function, $F_{G,L,T}(q_i,q_j,S(q_j))$, and a state function $S(q_j)$ as described above is deadlock free.

**Proof**

Firstly, we will consider the k-ary n-cube network for $n = 1$. Then, the n-dimensional network.

**Case 1.** $n = 1$

For this network and using $R_{DOR}$, there are only 2 cyclic dependencies, one in each direction, and each of them involved $k$ queues. The deadlock anomaly can appear in one of these cyclic dependencies, $D_C = \{q_{(c_1,1)},q_{(c+1,1)},\ldots,q_{((c+k-1)\text{mod}k,1)}\}$ if

$$
\sum_{i=0}^{k-1} \text{occu}(q_{((c+i)\text{mod}k,1)}) = \sum_{i=0}^{k-1} \text{cap}(q_{((c+i)\text{mod}k,1)})
$$

This implies that the last packet, $p_{j}$, injected from $q_{\text{inj}}$ in the queue $q_j \in D_C$ got the condition $F_{G,L,T}(q_{\text{inj}},q_j,S(q_j)) = \text{True}$.

But the state before all the resources were filled was:

$$
S(q_{(c,d)}),q_{(c+1,d)},q_{(c+1,d)},q_{(c+1,d)},\ldots,q_{((c+k-1,d))} = \text{Busy},\ldots,\text{Busy},G_{\text{busy}},\text{Busy},\ldots,\text{Busy}
$$

and

$$
F_{G,L,T}(q_{\text{inj}},q_j,S(q_j)) = \text{False} \quad \forall q_j \in D_C
$$

Hence, the last packet could not be accepted and the buffers could not be exhausted.

Besides,

$$
\exists q_a,q_b \in D_C \text{ with } R(q_a,n_d) = q_b \text{ and } S(q_b) = G_{\text{busy}} \rightarrow F_{G,L,T}(q_a,q_b,S(q_b)) = \text{True}
$$

so at least one packet inside the network can advance.

So, the network is deadlock free.

**Case 2.** $n > 1$

To extend the previous proof to a k-ary n-cube network is simple because under the $R_{DOR}$ function the
dependencies between queues are always from high dimension to low dimension.

Thus, provided there is a ghost packet at each unidimensional ring, all packets can advance along its current dimension. If all the packets of the low dimension, \( d = 1 \), advance and are consumed in a finite period of time, packets from dimension \( d = 2 \) will enter \( d = 1 \) in a finite period of time and be delivered too. That allow us to conclude that the \( k \)-ary \( n \)-cube network that fulfils the algorithm described above is deadlock free.

4 Network Performance

A low level simulation was used to evaluate the performance of a 2D-torus DOR network using the new flow control in comparison to its cut-through counterpart. Network performance is measured in terms of latency, throughput and hardware cost.

4.1 Router Model

The main components of every router are the crossbar, the routing module and the buffer. The input buffers, called BufferIn in figure 2, performs both the flow control and the storage of the packets. The header flit of the packet is updated by the Routing module which selects the path to follow. The Crossbar module arbitrates between input ports and provides the basic switching function from inputs to outputs. A virtual channel arbiter, \( V.C. \), is a component that regulates the access to the physical channel, so it is only required when two or more virtual channels are used.

Figure 2-a and figure 2-b outline the internal structures of Dally&Seitz's router with two virtual channels and \textit{Ghost-CT} router respectively. Notice the hardware cost has decreased in the second model because the \( V.C. \) modules are not needed and the crossbar complexity is reduced. This hardware reduction is reflected in the latency of the router.

4.2 Simulator

To evaluate the torus network performance under the \textit{ghost-CT} flow control we have built a simulator using Verilog, a hardware description language. This simulator allow us to analyze the latency and throughput performance of an \( 8 \times 8 \) bi-dimensional torus under different loads. Traffic patterns with uniform distribution have been chosen both to generate messages and to select their destinations.

Notice this simulator takes into account the router hardware complexity. The delay of the router components has been estimated according to Chien's delay model obtained for a 0.8\( \mu \) CMOS gate array technology [5]. This model calculates the components delay as a function of the input and output signals, \( P \), the grade of freedom, \( F \), and the number of virtual channels, \( V \). The different delays assigned to each component are shown in table 1. This fact has been omitted in many performance studies of parallel communication networks, although the different clock cycle for each router node determines the final performance [1].

The \textit{Ghost BufferIN} is the module that controls the movement of \textit{ghost packets} with an assigned delay of 2.20\( \mu \)s.

4.3 Simulation Results

The simulation results for the \textit{ghost-CT} flow control algorithm proposed in this paper for different loads and buffer capacities have been compared with the ones obtained using the Dally&Seitz algorithm [9].

Figure 3 shows latency and throughput obtained for both algorithms in an \( 8 \times 8 \) torus network (packet length equal to 10 flits). Labels \( G.CT_xpacking \) denote the \textit{ghost-CT} flow control algorithm with an input buffer capacity of \( x \) packets while labels \( VCT_xpacking \) denote results obtained with cut-through flow control and two virtual channels, each of them of size \( x/2 \). So,
the buffer requirements of $G_{\text{CT}}zpack$ and $VCzpack$ are the same.

Firstly, we can see in figure 3-a the effect that buffer capacity has in network latency. Focusing our attention on ghost-CT flow control, a significant latency improvement is obtained when increasing buffer capacity from 2 to 4 packets, $G_{\text{CT}2pack}$ versus $G_{\text{CT}4pack}$. Further increment to 10 packets per input queue shows only minor gains at high loads. Hence, a good trade-off between cost and performance for $G_{\text{packet}}$ flow control is obtained when the size of the queues is of 4 packets, $G_{\text{CT}4pack}$. The same conclusions can be drawn for the $VCzpack$ router.

Secondly, figure 3-a, shows that $G_{\text{CT}}$ achieves a latency reduction around 20% in relation to $VC$ for low loads. This latency improvement remains at higher loads, with latency reductions up to 35%. This is the result of hardware simplification due to the absence of virtual channels.

Network throughput is improved as well when using $G_{\text{CT}}$ flow control, as shown in figure 3-b. For instance, $G_{\text{CT}2pack}$ has a maximum sustained throughput which is 15% higher than the one achieved by its counterpart $VCzpack$.

In short, we can state that the $G_{\text{CT}}$ flow control, $G_{\text{CT}zpack}$, for a 2D torus network with DOR, improves the latency and the sustained throughput of Dally&Seitz's algorithm with two virtual channels, $G_{\text{CT}zpack}$ around 15% – 30%.

5 Conclusions

A deadlock-free algorithm for $k$-ary $n$-cube networks with a minimum buffer capacity requirement of a packet, the ghost flow control algorithm, has been presented. In this algorithm, packets advance following the $RDOR$ function towards their destinations, and ghost packets, which represent a special queue state, travel under demand in the opposite direction.

A ghost packet prevents incoming traffic from exhausting the resources of any possible cyclic dependency so that network packets can make use of that reserved buffer space and advance towards their destinations. This flow control can be extended to other routing algorithms and topologies by identifying the set of cyclic dependencies, then defining how ghost packets must circulate and how they restrict packet movement accordingly.

The evaluation of the algorithm by simulation, in which Chien’s delay model has been used to estimate hardware cost, shows there are significant gains for eliminating the need for virtual channels. An improvement in the mean latency and throughput of the packets of around 20% is obtained with this algorithm.
comparing with the Dally & Seitz algorithm with two virtual channels per link.

In the light of these results for the $G_{CT}$ flow control algorithm, a router implementation is being undertaken. We are also studying how to extend this flow control mechanism to other routings schemes.

References