J. Parallel Distrib. Comput. 72 (2012) 1412–1422

Contents lists available at SciVerse ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Efficient implementation of globally-aware network flow control

Lizhong Chen *, Ruisheng Wang, Timothy M. Pinkston

Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA

ARTICLE INFO

Article history:
Received 16 July 2011
Received in revised form
30 January 2012
Accepted 2 February 2012
Available online 10 February 2012

Keywords:
Interconnection network
Bubble Flow Control
Adaptive routing
Protocol-induced or message-dependent deadlock avoidance
Globally-aware network flow control

ABSTRACT

Network flow control mechanisms that are aware of global conditions potentially can achieve higher performance than flow control mechanisms that are only locally aware. Owing to high implementation overhead, globally-aware flow control mechanisms in their purest form are seldom adopted in practice, leading to less efficient simplified implementations. In this paper, we propose an efficient implementation of a globally-aware flow control mechanism, called *Critical Bubble Scheme,* for *k*-ary *n*-cube networks. This scheme achieves near-optimal performance with the same minimal buffer requirements of globally-aware flow control and can be further generalized to implement the general class of buffer occupancy-based network flow control. We prove deadlock freedom of the proposed scheme and exploit its use in handling protocol-induced deadlocks in on-chip environments. We evaluate the proposed scheme using both synthetic traffic and real application loads. Simulation results show that the proposed scheme can reduce the buffer access component of packet latency by as much as 62% over locally-aware flow control, and improve average packet latency by 18.8% and overall execution time by 7.2% in full system simulation.

1. Introduction

With parallel processing pervading the entire computing landscape – from server clouds built from chip multi-processors to embedded compute nodes built from systems-on-chip – the interconnection network plays an increasingly important role by providing efficient communication support among various system components. Along with routing, network flow control aims to maximize resource utilization while preventing oversubscription and deadlock in resource usage. It is, thus, critical in achieving high network and overall system performance.

Flow control mechanisms can be classified as being either locally-aware or globally-aware. Locally-aware network flow control mechanisms allocate network resources (e.g., channel buffers) to packets based solely on information local to router nodes, e.g., channel buffer occupancy of neighboring nodes. In contrast, globally-aware network flow control mechanisms make resource allocation decisions based on global network conditions that include local status information, e.g., channel buffer occupancy of neighboring router nodes as well as remote nodes. Recent work has shown that globally-aware flow control mechanisms can reap benefits in reducing network congestion and improve performance in terms of both latency and throughput [9,14,21]. However, such mechanisms typically rely on an omniscient global controller capable of collecting and distributing remote information nearly instantaneously everywhere at all times, which incurs substantial cost overhead or relies on simplified implementations which, oftentimes, are much less efficient.

A representative globally-aware flow control mechanism is Bubble Flow Control (BFC) which, in theory, requires a global controller to fully realize global awareness [18]. As implementing such a controller is complex, simplified versions of BFC that provide only local awareness have been implemented, e.g., in IBM Blue Gene/L [1], losing the potential benefits of true global awareness. BFC has been applied to off-chip environments to combat routing-induced deadlocks but can also be applied to the on-chip domain to avoid protocol-induced (i.e., message-dependent [19]) deadlocks caused by multiple message classes in cache coherence protocols used for shared memory chip multiprocessors. However, current simplified implementations of BFC would require many buffer resources to be devoted to safeguard against protocol-induced deadlocks which may occur rarely [17,19]. Hence, both off-chip and on-chip networks stand to gain from a more efficient implementation of BFC to achieve the benefits of globally-aware flow control while avoiding the drawbacks from added complexity of a global controller.

In this paper, we address the looming issue of realizing globally-aware flow control mechanisms by proposing the *Critical Bubble Scheme*, an efficient implementation of BFC applied to *k*-ary *n*-cube networks for the general class of buffer occupancy-based network flow control techniques. The basic idea behind the scheme is to mark and track as "critical" a certain number of free packet-sized buffers or bubbles (minimally, one per network dimension) and appropriately use only those critical bubbles to restrict packet injection for avoiding deadlock. This achieves

* Correspondence to: University of Southern California, 3740 McClintock Ave., EEB-224, Los Angeles, CA 90089, USA.
E-mail addresses: lizhongc@usc.edu (L. Chen), ruishenw@usc.edu (R. Wang), tpink@usc.edu (T.M. Pinkston).

0743-7315/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2012.02.004

nearly the same effect as would be provided by an omniscient global controller needed to implement theoretically-optimal BFC while not incurring the complexity of a global controller or the performance inefficiencies of the simplified implementation. We investigate the proposed Critical Bubble Scheme in both its minimized form (i.e., only one critical bubble per network dimension) and its generalized form (i.e., x critical bubbles per dimension).

The main contributions and organization of this paper are the following. The potential advantages and existing challenges of implementing globally-aware flow control mechanisms are discussed in Section 2 relative to locally-aware flow control. The Critical Bubble Scheme is proposed in Section 3 and shown as a way to implement near-theoretically-optimal BFC, which can reduce the minimum number of channel buffers needed to avoid deadlock by 50% compared to Localized BFC. This benefit of buffer reduction is also exploited by efficiently handling protocol-induced deadlock. Section 3.4 provides theoretical proof sketches of deadlock freedom of the proposed Critical Bubble Scheme for virtual cut-through switched k-ary n-cube networks. Modifications to router architecture for implementing the Critical Bubble Scheme are described in Section 4, and the proposed scheme is evaluated using both synthetic traffic and real application loads in Section 5. Simulation results show that the Critical Bubble Scheme can reduce the buffer access component of packet latency by as much as 62% over locally-aware flow control, and can achieve 18.8% reduction in average packet latency and 7.2% reduction in overall execution time under full system simulation. The scalability of the Critical Bubble Scheme and its generalization of using multiple critical bubbles to efficiently realize a large class of buffer occupancy-based globally-aware flow control mechanisms is discussed in Section 6. Finally, related work on globally-aware flow control and protocol-induced deadlock are summarized in Sections 7 and 8 concludes the paper.

2. Need for efficient implementation of globally-aware flow control

2.1. 1 pros and cons of globally-aware flow control

Fundamental metrics used to evaluate interconnection networks (such as throughput, latency, power and cost) are global measures as overall performance rarely is determined by the status of a given link or router but, rather, on communication paths comprised of multiple links and routers across the network. If resource allocation decisions are made locally at nodes, the effect should be optimized over the entire network. Given this, globally-aware flow control in which nodes take into consideration conditions from across the network in making decisions locally is preferred over locally-aware flow control where only local information is taken into account.

Globally-aware flow control has at least three advantages. First, as the status across the network typically is non-uniform, a locally-aware node may have quite different local status information than remote nodes. Hence, an allocation decision based solely on this information may not only be suboptimal but can even be counter-optimal. Second, as mentioned in [21], global awareness allows changes in network conditions to be detected earlier than the use of local information only as the latter suffers propagation delay of backpressure to the local node. Third, globally-aware flow control enables finer-granularity in optimally tuning network resources and restrictions on the use of those resources. For example, local-only flow control may unnecessarily place restrictions on the use of some set of local resources (e.g., channel buffers), thus requiring more resources as every node must enforce the same restrictions. With globally-aware flow control, local restrictions can be eased by enforcing restrictions amortized over a larger set of resources.

Along with these advantages, unfortunately, are some challenges in efficiently implementing globally-aware flow control. Key global information about network conditions must be gathered and acted upon by an omniscient global controller. This requires either prohibitively high overhead cost to implement or simplified implementations which can be quite inefficient. The following discusses a representative example.

2.2. Bubble Flow Control

Bubble Flow Control proposed in [18] is a well-known globally-aware flow control mechanism that reduces to a locally-aware flow control mechanism in simplified form. In what follows, we use the term *Theoretically-optimal Bubble Flow Control* (Theoretical BFC) to refer to the theoretically-optimal instantiation of Bubble Flow Control and use the term *Localized Bubble Flow Control* (Localized BFC) to refer to the simplified implementation in which only local information is used to control bubble flow. This is the scheme adopted in BFC implementations to-date [1]. Also in what follows, a *bubble* denotes a free packet-sized buffer.

(1) *Theoretical BFC*: Bubble Flow Control is applicable to k-ary n-cube networks (e.g., 2D tori). Fully globally-aware in its theoretically-optimal form, it applies virtual cut-through flow control in a way to avoid deadlock while requiring nominal buffer resources across the network. Dimension-order routing (DOR) in tori eliminates cyclic routing dependences that can occur across various network dimensions. It does not prevent, however, cyclic dependences and routing-induced deadlock that can occur within dimensions caused by torus wraparound links. A classic solution to this deadlock problem is to use a dateline technique in which two virtual channels (high and low) are associated with each physical channel [7]. When packets transported on low channels cross the dateline, they switch to high channels thus breaking cyclic dependency within network dimensions. A drawback of this approach, however, is the requirement of two virtual channels per link with their corresponding buffer resources.

Theoretical BFC reduces buffer requirements to only one virtual channel by imposing two simple rules on injection and forwarding of packets across dimensions. The idea is to prevent packets from using the potentially last free buffer of a dimensional ring [18]. The two rules are the following.

(i) Forwarding of a packet within a dimension is allowed if the receiving channel buffer has at least one packet-sized free buffer, i.e., a bubble.
(ii) Forwarding of a packet from one dimension to another (including injection of a new packet into a dimension) is allowed if the receiving channel buffer has a packet-sized free buffer and there is at least one additional free buffer located anywhere among the channel buffers of any router within that directional ring.

The first rule is the same as virtual cut-through flow control. In order to understand the second rule, Fig. 1 shows a simple illustration. Let us say this ring is a unidirectional ring of an arbitrary dimension in a k-ary n-cube network. Shaded rectangles indicate full buffers whereas non-shaded ones indicate empty buffers. Packet P wishes to enter into this dimension either from a different dimension or from an injection point (i.e., attached processor node). This access will be allowed only if the receiving channel buffer in Router I has a packet-sized free buffer, and there is an additional free buffer located anywhere (for example Router J) in the same direction. In this way, after accepting packet P, there is always at least one free buffer in the ring, which guarantees that at least one packet is able to make progress. This free buffer acts as

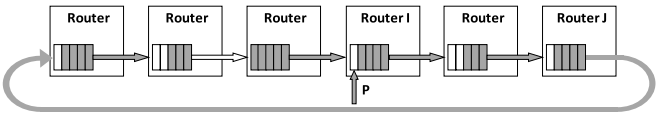


Fig. 1. Theoretical BFC requires global buffer information in the ring to avoid deadlock.

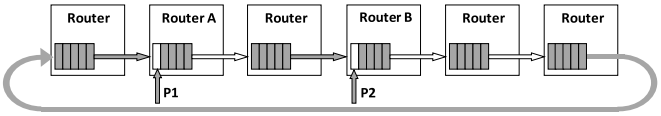


Fig. 2. Simultaneous injection in Theoretical BFC requires global coordination.

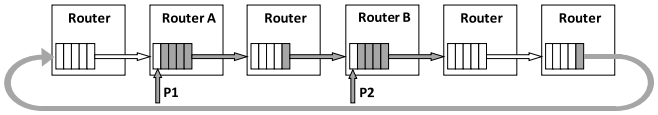


Fig. 3. Localized BFC decisions are not optimal across the network.

a *bubble* and ensures deadlock freedom. A detailed mathematical proof can be found in the original paper [18].

(2) *Difficulties with Theoretical BFC*: The allocation of buffer resources in Theoretical BFC requires buffer utilization information of other nodes in the directional rings of the network. A major difficulty in implementing Theoretical BFC is the need for a global controller. The global controller must gather and distribute global information about free buffer status within each network ring so that every node has sufficient global knowledge to decide on the allocation of buffers to requesting packets. Even with perfect global knowledge of free buffer status provided by the global controller, Theoretical BFC requires additional global control to handle multiple simultaneous injection requests in a deadlock-free manner. Consider the scenario in Fig. 2 in which, in the same cycle, two packets request to inject into the same directional ring containing only two free buffers. At most only one of the injections should be granted to avoid a potential deadlock configuration. However, according to the rules in [18] defining Theoretical BFC, each of the two routers will allow the injections based on global knowledge of the existence of two free buffers in the dimensional ring. Hence, besides enforcing the rules as defined in [18], a global controller must also determine which one of the multiple simultaneous injection requests should be granted to avoid deadlock and eliminate uncertainty. The above major difficulties hinder the adoption of Theoretical BFC in practice.

(3) *Localized BFC and its shortcomings*: To obviate the need for a global controller, a Localized BFC scheme was proposed and adopted to simplify BFC implementation [1,18]. Instead of checking for the existence of two free buffers anywhere along the directional ring as in Theoretical BFC, Localized BFC checks only that there are two free buffers in the channel buffer of the receiving router. For example, if the channel buffer of Router *I* in Fig. 1 has two free buffers, access will be granted to packet *P*. This simplification essentially reduces the globally-aware Theoretical BFC technique to a locally-aware technique as, now, all decisions are made based solely on local information as opposed to global information across the network's dimensional ring.

Localized BFC has three shortcomings that can degrade performance. First, by requiring two free buffers in the local channel buffer for packets to enter a new dimension, Localized BFC increases the *buffer access delay* component of packet latency. As shown in Fig. 3, both packets satisfy the Theoretical BFC rules and should be granted access given that overall buffer occupancy within the dimension is far from saturation. However, both accesses are denied with Localized BFC as only one free buffer exists in the corresponding local channels. Hence, the packets suffer a longer buffer access delay than incurred by Theoretical BFC.

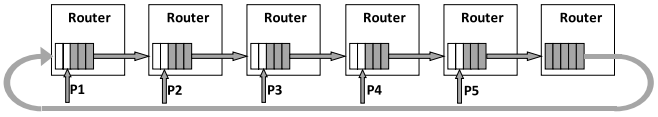


Fig. 4. Localized BFC requires more free buffers than that are needed by Theoretical BFC.

Second, Localized BFC does not use channel buffers to their theoretically optimal level as more than the minimum number of required buffers remain unused within each dimension. Fig. 4 depicts a representative scenario. Five packets wish to enter the directional ring simultaneously. According to Theoretical BFC, the minimum number of free buffers needed is six: one for each of the five packets plus one bubble to avoid deadlock. However, Localized BFC requires there to be at least ten free buffers: two free buffers for each packet. Thus, in this case, Localized BFC requires 66% more free buffers to guarantee deadlock freedom, which is highly inefficient.

Third, in this simplified implementation of BFC, the minimum number of buffers needed to avoid deadlock in each channel buffer of routers now becomes two instead of one as required by Theoretical BFC. This introduces another inefficiency, particularly when channels have shallow/minimum buffers. For example, given tight buffer budgets, some on-chip network or network-on-chip (NoC) designs may allow for only one packet-sized buffer per channel, which precludes the use of Localized BFC. Since NoC-based systems are gaining increasing importance in recent years, the following subsection discusses this issue in more detail.

The increased buffer access delay, lower buffer utilization, and increased minimum buffer size of Localized BFC leads to performance degradation and higher cost of the network, as shown experimentally later in Section 5.

2.3. Inefficiency of Localized BFC for handling protocol-induced deadlock

So far, the deadlock that we have discussed is *routing-induced deadlock*. However, there is also another type of deadlock—*protocol-induced* or *message-dependent deadlock* that can occur in computer systems such as shared-memory chip multiprocessors (CMPs). While its formal model and detailed description are provided in [19,22], essentially, message-dependent deadlock is caused by multiple dependent message classes in cache coherence protocols. For example, *reply* messages can only be generated by nodes receiving *request* messages. In other words, *reply* messages depend on *request* messages. This inter-message dependency creates additional arcs in the channel dependency graph, and if it happens to complete a cycle, then deadlock may occur.

To avoid message-dependent deadlock, separate logical networks (e.g., implemented as virtual channels) are often employed for separating different message classes [7]. Since the actual frequency of deadlock typically is very low [17,19], those logical networks acting as escape channels should be designed to use as few resources as possible so that the remaining buffer resources can be shared among all message classes as adaptive channels, thus maximizing resource utilization [12].

However, when applying Localized BFC to handle message-dependent deadlock in torus networks, resource utilization is far from optimal. For example, in the MOESI directory cache protocol [13], different message types can be classified into three dependent message classes which require at least three independent layers of virtual channels in Localized BFC to serve as

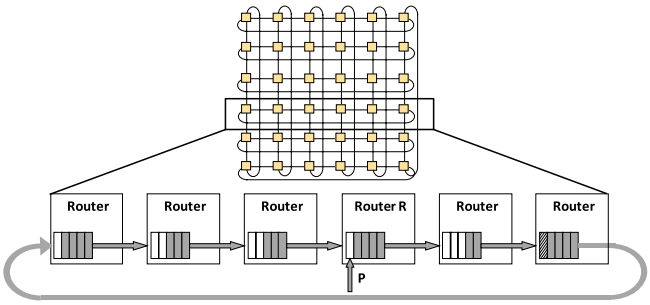


Fig. 5. The Critical Bubble Scheme avoids deadlock without requiring explicit global coordination.

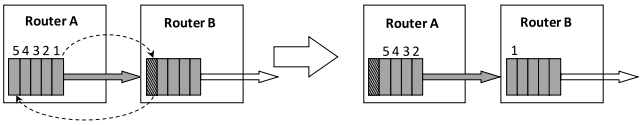


Fig. 6. Transfer critical bubble between routers.

escape resources. This would increase to six layers if the traditional dateline technique is used. Each layer of virtual channels for Localized BFC would require at least two buffers per VC. An additional layer of virtual channels with at least one buffer per VC is needed to implement minimal adaptive routing. This configuration devotes a large portion of available resources (i.e., 3 out of 4 virtual channel layers, or 6 out of 7 buffers) to avoid message-dependent deadlock in adaptive routing and safeguard against likely rare cases of potential deadlock.

3. Critical Bubble Scheme

3.1. The basic idea

Our proposed Critical Bubble Scheme is an efficient implementation of Theoretical BFC that retains global awareness properties for realizing near-maximal benefits, while avoiding costly global coordination. The principle behind the Critical Bubble Scheme is to mark and track as critical at least one bubble in each directional ring of a network and restrict the use of critical bubble(s) only to packets traveling within dimensions to prevent deadlock, as in Theoretical BFC. Critical bubble(s) flow within and are confined to directional rings of the network. Their movement is tracked using nominal control signals between neighboring routers. In essence, their presence (or non-presence) at router nodes conveys global information across each network dimension about what restrictions should be enforced locally to avoid deadlock.

As an illustration of how the scheme works, consider an arbitrary ring within a 2D torus as shown in Fig. 5. Initially, for each unidirectional ring of the network, a free buffer in any router along the ring is marked as the critical bubble for the entire ring (i.e., striped rectangle). This critical bubble is transferred backwards between routers and tracked along the ring as intra-dimensional packets displacing it move forward. Assume a packet P from another dimension wishes to enter this dimension through router R. If the one free buffer in R's local receiving channel is not the critical bubble (i.e., it is a non-critical bubble as shown in the figure), the packet is allowed to enter this dimension (i.e., it is allocated the free buffer) immediately. The packet need not wait for a second buffer in the local channel to become free as in Localized BFC nor does it have to check buffer occupancy of other nodes along the ring for explicit global awareness. The absence of the critical bubble at the local router indicates the existence of a free buffer elsewhere in the ring, thus providing implicit global awareness.

3.2. Detailed description of the Critical Bubble Scheme

(1) *Initialization*: Assume a k-ary n-cube in which every dimension is composed of two opposite unidirectional rings. For each unidirectional ring, a random free buffer from any router channel belonging to this direction can be marked as the critical bubble. The resulting network has one critical bubble in every dimension and direction. The other free buffers operate as normal buffers (i.e., non-critical bubbles).

(2) *Formal rules*: In implementing Theoretical BFC, the Critical Bubble Scheme imposes the following two rules on the forwarding of packets to avoid deadlock.

(i) Forwarding of a packet within a dimension is allowed if the receiving channel buffer has one packet-sized free buffer, no matter whether it is a normal free buffer or a critical bubble.
(ii) Forwarding of a packet from one dimension to another (including injection of a new packet into a dimension) is allowed only if the receiving channel buffer has at least one packet-sized normal free buffer. It is not allowed if the only packet-sized free buffer is a critical bubble.

(3) *Transfer of critical bubbles*: A key requirement of the Critical Bubble Scheme is always to maintain a free buffer (critical bubble) in each unidirectional ring of the network. According to the types of packet forwarding, there are two cases to be considered.

(a) If a packet is allowed to change dimension or inject into a new dimension, then according to the second rule, there is a normal free buffer ready to accept this packet. Hence, the critical bubble remains untouched.
(b) If a packet is forwarded from router A to router B within a dimension, then if the receiving buffer in router B has a normal free buffer, the critical bubble remains unused. Otherwise, as depicted in Fig. 6, the arrival of packet 1 at router B displaces the critical bubble backward to router A. This is done by router B asserting a special control line to indicate to router A that it should mark the newly freed buffer in router A as the critical bubble for the ring. More implementation details are provided in Section 4.

3.3. Impact on implementation and performance

The Critical Bubble Scheme provides an elegant and efficient implementation solution to overcome the difficulties of implementing Theoretical BFC while still avoiding the deficiencies of Localized BFC. It is also particularly useful in supporting multiple message classes in buffer-resource-limited on-chip network environments.

(1) *Implementing global awareness of Theoretical BFC*: As discussed in Section 2, the most difficult problem in implementing Theoretical BFC is the need for a global controller. The main advantage of the Critical Bubble Scheme is its global awareness without the need for a complex global controller. No status information needs to be gathered or distributed remotely across the network, and no uncertainty arises as to whether local routers can allocate buffer resources to packets when multiple simultaneous requests are made for dimensional resources. This is illustrated in Fig. 7 which shows the same scenario shown in Fig. 2 but for the Critical Bubble Scheme. Again, we have two free buffers in the dimension but also have two packets wishing to turn into the dimension. At most one of these packets should be granted access. It is clear that router A will deny access as there is no normal free buffer left, and router B will grant access. Uncertainty is removed without needing explicit global coordination.

(2) *Avoiding the deficiencies of Localized BFC*: The Critical Bubble Scheme also avoids all three deficiencies of Localized BFC. First, there is no increased access delay. Fig. 8 depicts the scenario in

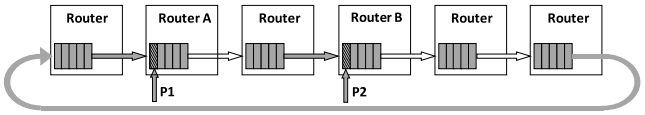


Fig. 7. Simultaneous injections have no uncertainty with the Critical Bubble Scheme.

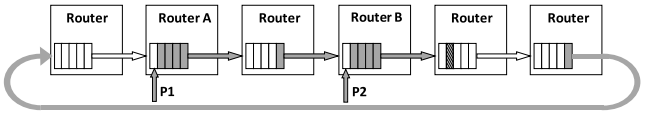


Fig. 8. Simultaneous injection of multiple packets with the Critical Bubble Scheme.

Fig. 3 but for the Critical Bubble Scheme. As can be been, even though there is only one free buffer at either of the receiving channels in routers *A* and *B*, both packets can access this dimension immediately without incurring extra buffer access delay.

Second, the scheme improves buffer utilization over Localized BFC. As shown in Fig. 4, no matter whether the critical bubble is in one of the five injecting router channels or in the rightmost router, only six free buffers are required to grant all accesses, achieving the minimum number of free buffers as in the theoretically-optimal case.

Third, as mentioned in Section 2.2, Localized BFC requires a minimum of two packet-sized buffers in every receiving channel while Theoretical BFC requires only one. The reason is that Localized BFC has to check that there are at least two free buffers in the channel buffer before allowing packets access. With the Critical Bubble Scheme, just one packet-sized buffer is required in every channel as this scheme checks only that there is at least one normal free buffer in the channel before granting access to packets. Therefore, the Critical Bubble Scheme effectively achieves the minimum number of buffers in every channel as dictated by Theoretical BFC.

(3) *Efficiently handling message-dependent deadlock*: As described earlier, in order to eliminate deadlock among the various message classes, routers require at least one escape virtual channel per port for each message class. With the Critical Bubble Scheme, it is possible to halve the buffer resources required in each channel and allocate the saved buffers to adaptive channels to speed up the common case. For the MOESI directory protocol example in Section 2.3, the Critical Bubble Scheme minimally requires three packet-sized buffers – one for each of the three message classes to implement escape paths – which allows the adaptive channel to use the four remaining buffers assuming the same total channel buffer budget. This greatly increases the potential buffer utilization and can improve system performance, as shown experimentally later in Section 5.

3.4. Deadlock freedom

This subsection provides formal proof sketches of the deadlock freedom of the proposed Critical Bubble Scheme for virtual cut-through switched torus networks for all the cases discussed above. To facilitate the proofs, we first introduce some basic definitions derived from [8].

Definition 1. Q represents the set of input queues associated with router nodes. Each queue $q_i \in Q$ has capacity of $cap(q_i)$ packets and the current number of packets occupying the queue is denoted as $size(q_i)$. Q_I is a subset of Q consisting of all injection queues. Q_y is a subset of Q consisting of all input queues belonging to some unidirectional ring y. Each queue, q_i, associates with a binary number b_i. It is set to 1 if the next free buffer of this queue is a critical bubble.

Definition 2. $F(q_i, q_j)$ is the flow control function for a packet that wishes to enter q_j from q_i. It can be either true or false, and the packet is allowed to take this move only if F is true.

Definition 3. If at any given cycle, there is at most one injection request to Q_y (or forwarding request from another dimension to dimension y), then the request pattern is called *sequential injection requests*. Otherwise, if there are multiple injection requests to Q_y (including forwarding requests from another dimension) in the same cycle, then the request pattern is called *simultaneous injection requests*.

Definition 4. A dependency between message classes M_i and M_j in which the generation of messages in M_j depends on the consumption of messages in M_i at nodes, defined by the communication protocol, is denoted by $M_i \leftarrow M_j$, in which $\leftarrow$ is a partial order relation indicating that M_i precedes M_j which is the terminus of the dependency.

Lemma 1. *The Critical Bubble Scheme is deadlock-free under sequential injection requests.*

Proof Sketch. Assume there is a packet that wishes to move from q_i in unidirectional ring x of a dimension to q_j in unidirectional ring y in another dimension. The rules of Theoretical BFC are the following:

$$\textit{Rule } 1.\ \textit{When } x = y, F(q_i, q_j) \textit{ is True if } : size(q_j) \leq cap(q_j) - 1 \quad (1)$$

$$\textit{Rule } 2.\ \textit{When } x \neq y \vee q_i \in Q_I, F(q_i, q_j) \textit{ is True if } : size(q_j) \leq cap(q_j) - 1 \wedge \quad (2)$$

$$\sum size(q_k) \leq \sum cap(q_k) - 2, \ \textit{over all } q_k \in Q_y. \quad (3)$$

The rules of Critical Bubble Scheme are the following:

$$\textit{Rule } 1^*.\ \textit{When } x = y, F(q_i, q_j) \textit{ is True if } : size(q_j) \leq cap(q_j) - 1 \quad (4)$$

$$\textit{Rule } 2^*.\ \textit{When } x \neq y \vee q_i \in Q_I, F(q_i, q_j) \textit{ is True if } : size(q_j) \leq cap(q_j) - 1 \wedge \quad (5)$$

$$b_j = 0. \quad (6)$$

We now prove by considering all cases that if a flow control function F_{CBS} satisfies *Rule* 1^* and 2^*, it must also satisfy *Rule* 1 and 2. First, comparing (1) with (4) and (2) with (5), they are exactly the same. Second, since F_{CBS} follows (6), it means the critical bubble is not in the next free buffer of input queue q_j but is somewhere else in Q_y. This indicates that there are at least two free buffers in unidirectional ring y: one is the free buffer found in (5), and the other is the critical bubble elsewhere in the ring. This is exactly the condition of (3). Therefore, by enforcing the two new rules, the Critical Bubble Scheme also satisfies the rules enforced by Theoretical BFC. As Theoretical BFC is proved to be deadlock-free with sequential injection request under its two rules in [1], the Critical Bubble Scheme is also deadlock-free under sequential injection request. □

Lemma 2. *The Critical Bubble Scheme is deadlock-free under simultaneous injection requests.*

Proof Sketch. First consider the case of packets being transported only within unidirectional rings of a network. As long as a critical bubble exists within each unidirectional ring, there is no intra-dimensional deadlock. This is because, in the worst case of all other buffers in the unidirectional rings being occupied, the critical bubble within each ring serves as the last free buffer, guaranteeing that at least one packet in each ring can make forward progress. This is the fundamental principle behind Bubble Flow Control.

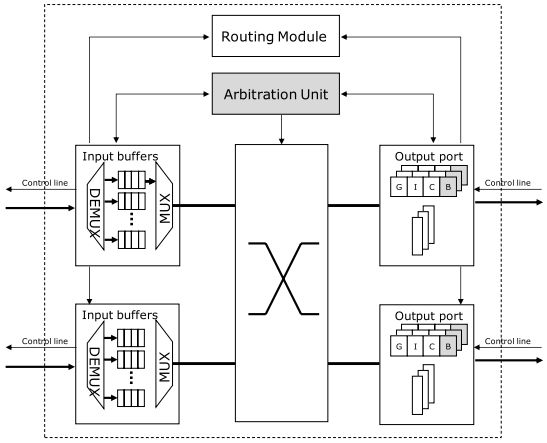


Fig. 9. Typical virtual cut-through router microarchitecture. The shaded areas are modified to implement the proposed Critical Bubble Scheme.

Now consider the situation in which there are k simultaneous injection requests to ring y in cycle T by packets from outside the ring and ring y has no intra-dimensional deadlock before T. Each injection request is either rejected or granted. If every request is rejected, then no more free buffers of ring y are consumed, and ring y remains deadlock-free. Otherwise at least one request is granted. Consider the worst case in which all requests are granted. Since each granted request must satisfy (5) and (6), there must be a normal free buffer (i.e., non-critical bubble) ready to accept the injected packet. Therefore, even if all packets are injected into ring y, a critical bubble continues to exist within the ring guaranteeing that at least one packet in the ring can make forward progress. Together with dimension-order routing, no inter-dimensional deadlock can occur either. Thus, the Critical Bubble Scheme is deadlock-free under simultaneous injection requests. □

Lemma 3. *The Critical Bubble Scheme applied to resources used for each message class of a communication protocol is deadlock-free under either sequential or simultaneous injection requests.*

Proof Sketch. Assume there are n message classes and the total virtual channel resources C are divided into two disjoint subsets C_1 and C_2. C_1 can be shared amongst all message classes (i.e., adaptive channels) while C_2 is further divided into n independent subsets, S_k ($k = 1, 2, \ldots, n$), one for each message class which is the minimum needed to separate message classes into distinct sets of escape resources, according to [19]. We prove the lemma by induction, considering all cases.

Without loss of generality, assume the n message classes have dependency $M_1 \leftarrow M_2 \cdots \leftarrow M_{n-1} \leftarrow M_n$ as defined by the protocol. As any set of S_k is independent of other sets and a partial ordering exists such that $S_1 \leftarrow S_2 \cdots \leftarrow S_{n-1} \leftarrow S_n$, there is no cyclic dependency among the n sets of escape virtual channels. When applying the Critical Bubble Scheme to each message class M_k separately, according to Lemmas 1 and 2, S_k is deadlock-free. However, as messages in M_n which is the terminus of the dependency can drain from S_n, messages in M_{n-1} can also drain from S_{n-1} and so on until messages in M_1 drain from S_1. Thus, the union of S_k (i.e., the C_2 channels) serve as escape resources for all channels C, including C_1 channels. According to [19], no deadlock can occur. □

4. Router architecture

This section discusses the modification of standard router architecture to support our Critical Bubble Scheme.

4.1. Typical virtual cut-through router architecture

Fig. 9 shows a block diagram of the architecture of a typical virtual cut-through router. Arriving packets first get stored in the input buffer and advance in FIFO manner. The routing module is responsible for computing the output port for packets. When a packet is at the head of the FIFO queue and ready to move, the arbitration unit will configure the switch to set up a path for the packet to the allocated virtual channel in the output. The packet then traverses the switch to the output port and moves to the next hop. In a cut-through router, virtual channel allocation and switch arbitration can be merged into a single module [7], denoted as the Arbitration Unit here. Each output virtual channel also contains several state fields to track its status, including a state I that records the input port and virtual channel that are forwarding packets to this output virtual channel, and a state C that counts the number of credits in the downstream input channel buffer.

4.2. Router architecture for Critical Bubble Scheme

The router architecture to enable the Critical Bubble Scheme is very similar to the typical virtual cut-through router architecture. We need three modifications to the router architecture as shown shaded in Fig. 9: a counter B at the output channel to count the number of critical bubbles in the input channel of the downstream router, a 1-bit control line to indicate the increase of B, and a slightly modified Arbitration Unit.

To illustrate the modification needed for the arbitration unit, we compare the original arbitration unit and the modified one. During arbitration, the original arbitration unit checks the availability of output virtual channels and whether there is packet entering or waiting in the input channel. Our modified arbitration unit adds the following check in parallel with the original checks:

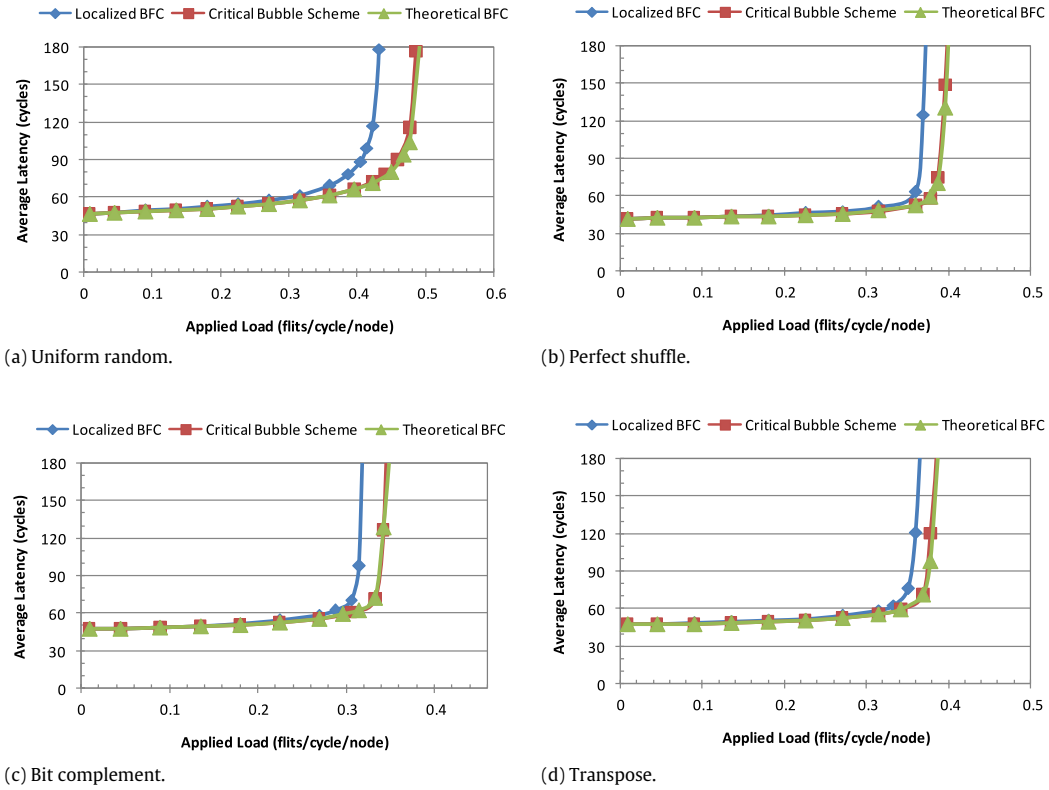
Input channel is in the same dimension as the output channel?	*No* $\Rightarrow$ *Bshould be less than C* *Yes* $\Rightarrow$ *If* $B = C$, *assert control line*

In the above condition, if the input is not in the same dimension as the output, then the added checking makes sure that there is at least one normal free buffer in the downstream input channel as the downstream channel has C free buffers but only B critical bubbles (e.g., $B = 1$). Since the output channel has a field I which records the input port and virtual channel, it needs only two comparators for the comparison of the input/output ports and B with C.

If the case of intra-dimensional packet forwarding, the arbitration unit checks whether B equals C before the packet is forwarded. If B equals C, the forwarded packet will occupy a critical bubble in the downstream input channel, causing the critical bubble to be displaced and transferred backward to the input channel of the router from which the packet came. This is done by decreasing B by 1 and decreasing the credit count. As the number of critical bubbles of the current router is stored in the upstream router, the 1-bit control line associated with the upstream router (denoted by *control line* in the figure) is asserted. When the upstream router detects an asserted control line signal, it will decrease its B in the next cycle and deassert the control line to complete the critical bubble transfer.

5. Evaluation

In this section, we present a detailed evaluation of our proposed Critical Bubble Scheme. We first stress this scheme under synthetic traffic to validate its correctness quantitatively across a wide



(a) Uniform random.

(b) Perfect shuffle.

(c) Bit complement.

(d) Transpose.

Fig. 10. Effects of the Critical Bubble Scheme under different synthetic traffic patterns.

range of load rates, examine its effects on buffer utilization, and compare its performance against other bubble-based flow control mechanisms. We then investigate the effectiveness of the Critical Bubble Scheme in handling message-dependent deadlock using full system simulation with the PARSEC benchmark suite.

5.1. Simulation methodology using synthetic loads

To evaluate the proposed scheme using synthetic loads, we implement it on a general-purpose cycle-accurate interconnection network simulator GARNET [2] written in C++. We modified the simulator to support virtual cut-through, adaptive routing and the Critical Bubble Scheme. Routers use a standard 4-stage pipeline with 1 cycle for link traversal. An 8-ary 2-cube torus network with bidirectional physical channels is simulated. Deadlock avoidance based on Bubble Flow Control is assumed using one escape and one adaptive virtual channel per physical channel and each virtual channel has two packet-sized buffers.

All simulations are run for 100,000 cycles with a warm-up period of 10,000 cycles. Packets are randomly generated to be either short packets with single-flit or long packets with nine flits. To stress the network, four synthetic traffic patterns are used in these evaluations: uniform random, perfect-shuffle, bit complement and transpose [7]. We compare the performance of Theoretical BFC, Localized BFC, and the Critical Bubble Scheme.

5.2. Effects of Critical Bubble Scheme under synthetic traffic

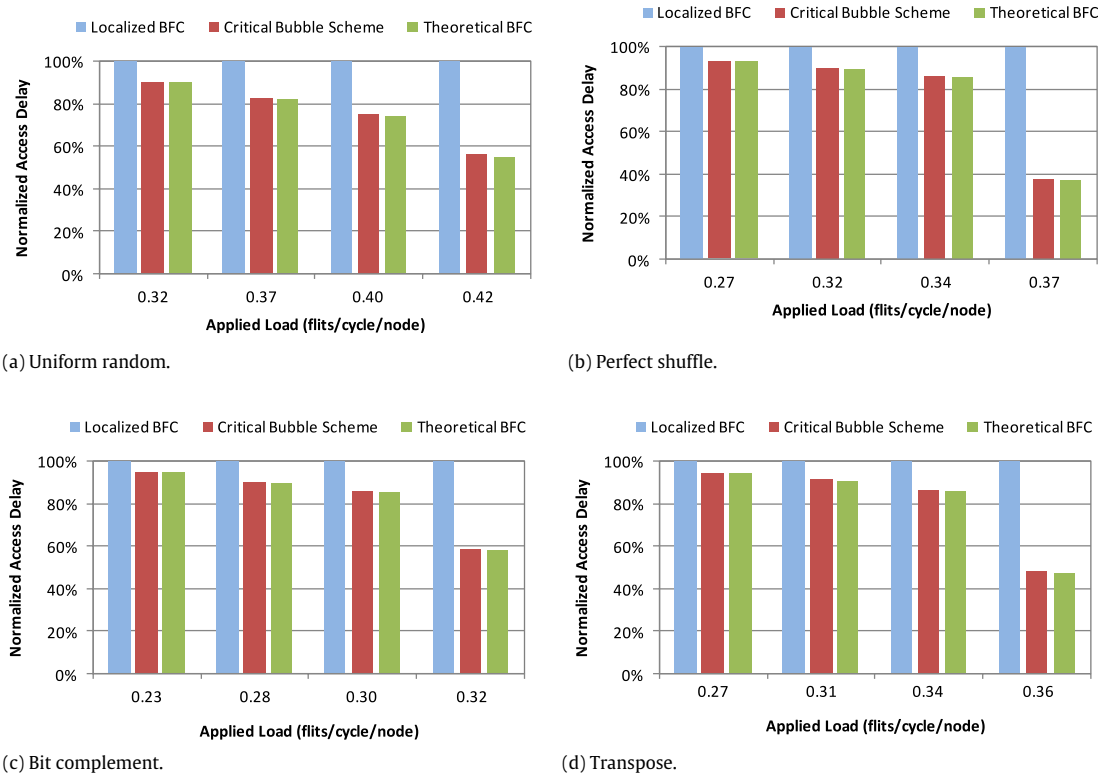
As the basis of this evaluation, we first validate the correctness of the Critical Bubble Scheme. Fig. 10 plots the performance of three versions of Bubble Flow Control: Theoretical BFC which is possible only in simulation and impractical in reality, Localized BFC which is a simplified implementation used in the original BFC paper [18], and our proposed Critical Bubble Scheme (CBS). As can be seen in the figure, the performance of CBS closely follows Theoretical BFC for all four traffic patterns, indicating that the proposed scheme is able to efficiently implement Theoretical BFC and, therefore, achieve similar potential maximum benefits.

When comparing the performance of CBS with Localized BFC under these four traffic patterns, CBS clearly has advantages. Specifically, CBS has much lower latency at medium and high load rates (relative to the load rate at the saturation point unless otherwise stated). When the applied load rate is low, buffer occupancy also is low in both schemes. Therefore Localized BFC, which requires at least two free buffers in the channel when injecting or changing dimensions of a packet, has almost the same effect as CBS which requires only one normal free buffer. However, as the applied load rate increases, the deficiencies of increased buffer access delay and low buffer utilization for Localized BFC gradually manifest while CBS, which has lower buffer requirements, enjoys a higher success rate for injection and changing dimensions by packets. The above lead to performance gains for the Critical Bubble Scheme under medium and high load rates.

To further demonstrate the effect of CBS on reducing the buffer access delay portion of packet latency, Fig. 11 plots the buffer access delay of CBS normalized to that of Localized BFC at medium and high load rates under the four traffic patterns. While the buffer access delay for injecting or changing dimensions of packets are very close between CBS and Theoretical BFC, the histogram shows significant reduction by up to 62% for CBS over Localized BFC. This indicates that the proposed Critical Bubble Scheme successfully overcomes the deficiencies of Localized BFC while achieving lower access delay and higher buffer utilization offered by Theoretical BFC.

5.3. Simulation methodology using full system simulation

We also evaluate the Critical Bubble Scheme under real application workloads using full system simulation. The simulation



(a) Uniform random.

(b) Perfect shuffle.

(c) Bit complement.

(d) Transpose.

Fig. 11. Effect of the Critical Bubble Scheme on reducing buffer access delay.

Table 1
Parameters used in full system simulation.

Component	Parameters
Network topology	4 × 4 Torus
Core	In-order, 3 GHz
Private L1 I& D cache	16 kB, 2-way, 1 cycle, LRU
Shared L2 cache	512 kB, 16-way, 6 cycles, LRU
Router	4-stage (RC-VA-SA-ST), 1 GHz
Link bandwidth	64-bit / cycle
Memory latency	128 cycles

infrastructure is based on SIMICS [11] enhanced with GEMS [13] for detailed timing of the memory and the modified GARNET [2]. We simulated a 16-node chip connected by a 4×4 2-D torus. Each node has an in-order core, a 16 kB private L1 cache and a 512 kB bank of shared L2 cache. The MOESI directory cache coherence protocol is used in the system with 2 memory controllers. Messages are divided into two lengths of packets. Short packets are 8-byte single-flit while long packets carrying 64-byte cache lines have nine flits. Additional parameters are listed in Table 1. We run PARSEC [4] benchmark suite compiled with pthread programming model.

To investigate the effectiveness of the Critical Bubble Scheme in handling message-dependent deadlock, we compare two configurations using the same buffer budget. Since the MOESI directory protocol has three dependent message classes, configuration 1 employs Localized BFC as the flow control for the three escape virtual channels, each having two packet-sized buffers. There is also a one packet-sized buffer for the adaptive virtual channel shared by all message classes. This is the minimal buffer resources required by Localized BFC using adaptive routing. Configuration 2 employs the Critical Bubble Scheme for each of the one packet-sized buffer for the escape virtual channel. The remaining four packet-sized buffers are used for adaptive virtual channels. Hence, both configurations have similar budget (i.e., 7 long packet-sized buffers) for implementing the 4 virtual channels per physical channel.

5.4. Effects of Critical Bubble Scheme in handling message-dependent deadlock

Fig. 12 compares the execution time of configuration 1 using Localized BFC and configuration 2 using the Critical Bubble Scheme across the benchmark suite. To present data from multiple applications more effectively, execution time is normalized to configuration 1. The results vary among applications as different applications may generate distinct network loads and have different sensitivity to network performance. On average, there is 7.2% overall execution time reduction with configuration 2. There are two major reasons for this improvement. The first is similar to the synthetic traffic case as the Critical Bubble Scheme reduces buffer access delay and increases buffer utilization. The second stems from the larger amount of buffer resources that can be used by adaptive virtual channels owing to the reduced resource requirements for escape virtual channels with our proposed scheme. This advantage of the Critical Bubble Scheme over Localized BFC increases with more complex protocols that have a greater number of dependent message classes.

We further analyze the impact of Critical Bubble Scheme on network performance in this environment of multiple message classes. Fig. 13 breaks down the average packet latency into zero-load latency (consists of serialization latency and hop latency) and contention latency [7]. Compared with configuration 1, configuration 2 using the Critical Bubble Scheme achieves an average of 18.8% reduction in average packet latency. This is mainly because the competition for scarce adaptive resources in configuration 1 incurs a large increase in contention latency. By using limited buffer resources more efficiently, configuration 2 is able to achieve better performance.

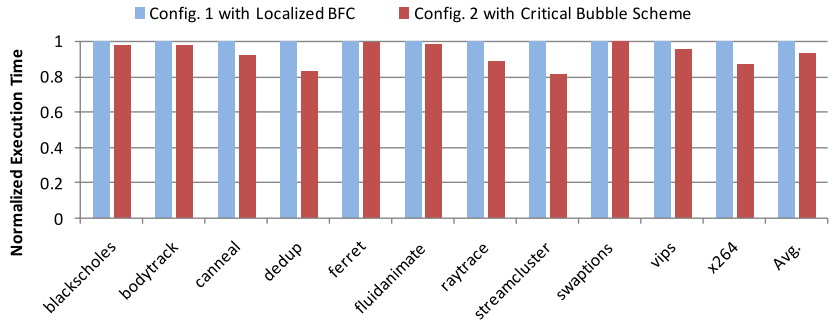


Fig. 12. Comparison of normalized execution time across different PARSEC applications.

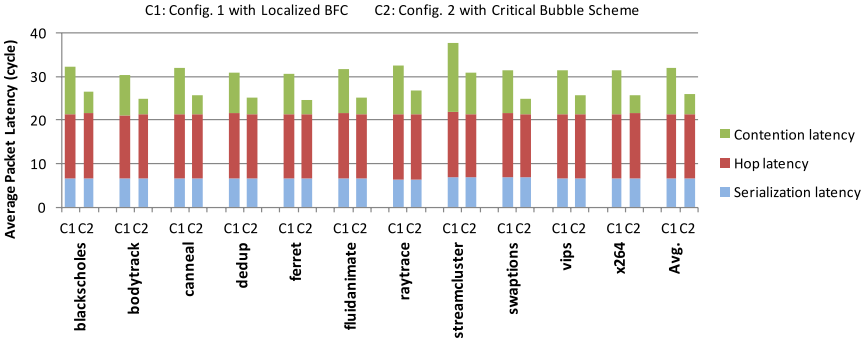


Fig. 13. Comparison of the breakdown of average packet latency across different applications.

6. Discussion

6.1. Scalability of Critical Bubble Scheme

We consider the scalability in terms of different network sizes and buffer sizes. With continuing technology scaling, the number of nodes to be connected in the network likely will increase. For larger networks, the difference between locally- and globally-aware flow control is more pronounced as local information can hardly reflect the global status of the network and the delay in propagating network state to local nodes will increase. Thus, the benefits of globally-aware CBS will be even greater for larger networks. Simulation results show that under uniform random traffic and high injection rate (i.e., 95% of the corresponding saturation load rate), average packet latency difference of CBS over Localized BFC for network sizes of 4×4 and 8×8 are 22.3% and 27.2%, respectively.

When buffers are large, there is little difference between Localized BFC and CBS as many free buffers are available anyway. However, in the more interesting and practical case (e.g., for NoCs) of shallow buffers, the relative fraction of free buffers needing to be reserved to avoid deadlock in Localized BFC is higher than with CBS, further reducing resource efficiency relative to CBS and causing increased performance degradation. The extreme case of one packet-sized buffer per channel precludes the use of Localized BFC. Simulation results show that when buffer size decreases from 4 to 3 to 2 buffers per channel under uniform random traffic and high injection rate, average latency difference of CBS over Localized BFC increases from 6.6% to 12.5% to 27.2%, respectively.

6.2. Implementing buffer occupancy-based global flow control using multiple critical bubbles

It is interesting to see that BFC is actually a special case of buffer occupancy-based flow control mechanisms which restrict the use of buffers based on their occupancy. A typical example of this is congestion control, which restricts packet injection into the network when the buffer occupancy is above a threshold to avoid drop in throughput from network saturation. Throttling can be locally-aware if it uses buffer occupancy of the local router or globally-aware if it uses aggregate buffer occupancy across a network dimension or throughout the network. The globally-aware version has many potential benefits as discussed, but efficient implementations remain a challenge.

The Critical Bubble Scheme can be generalized to implement this general class of globally-aware flow control. The idea is to allow multiple critical bubbles instead of just a single critical bubble as a quasi means of throttling packet injection to relieve network pressure. The only modification to the operation of the Critical Bubble Scheme described above is to mark multiple free buffers as critical bubbles during network initialization. The same two rules are enforced for forwarding packets, and deadlock freedom remains guaranteed. In this way, a broader class of globally-aware flow control mechanisms – such as global flow control with a preset buffer occupancy threshold T, self-tuned global congestion control with adaptive threshold T, and so on – can be implemented efficiently. Preliminary results show that global congestion control implemented using multiple critical bubbles successfully sustains throughput after network saturation giving 11% higher throughput than local congestion control. More discussion and preliminary results are presented in [6]. Further investigation of this topic is left for future work.

7. Related work

The notion of using bubbles (or ghost packets) in flow control to avoid deadlock in torus networks was first proposed in [5, 18], and adopted in the IBM Blue Gene/L [1]. However, only a localized version of Bubble Flow Control was actually implemented as opposed to a globally-aware version as proposed here.

Several works propose ways of improving buffer utilization and/or reducing buffer requirements. In [20], bubbles are drawn to heavily-congested spots within the network to relieve congestion and maintain deadlock freedom. Flit-reservation flow control was proposed in [16] to use buffers efficiently and reduce latency by scheduling buffer usage ahead of time. In [15], aggressive bufferless routing was proposed to eliminate the need for buffers. Our scheme differs from these proposals in that critical bubbles are used both to implement globally-aware flow control efficiently and to minimize buffer requirements for avoiding deadlock.

Message-dependent deadlock was discussed in detail in [19], and the formal model proposed in [22] can be used to analyze it. While those works provide foundation for handling message-dependent deadlock, our scheme can be employed to improve their efficiency.

Extensive evaluations of using global congestion control were presented in [9,10,14], which demonstrated the advantages of globally-aware flow control from a simulation perspective and showed that a buffer occupancy of around 50% results in high and sustained peek throughput. An ideal global controller is assumed in the evaluation, which is impractical to implement. Self-tuned congestion control using global knowledge is proposed in [21]. It uses a dimension-wise aggregation scheme to gather and propagate global information one hop after another, and the performance is shown to be better than the ALO scheme [3]. However, that scheme has to endure long delay to gather global information. Our proposed scheme can make grant decisions instantaneously.

8. Conclusions

Globally-aware flow control in interconnection networks has many potential advantages over locally-aware flow control but faces several serious implementation difficulties. The primary contribution of this paper is the development of the Critical Bubble Scheme (CBS) to provide a way to correctly and efficiently implement globally-aware flow control mechanisms. By marking a certain number of free buffers as critical bubbles (minimally one) and appropriately using them to restrict packet injection, this scheme achieves near-optimal performance offered by globally-aware flow control mechanisms while avoiding costly explicit global control and coordination. The proposed scheme can be readily applied to both off-chip and on-chip networks that employ Bubble Flow Control to ensure freedom from both routing-induced and protocol-induced deadlock.

Acknowledgments

The helpful comments and suggestions made by the reviewers are gratefully acknowledged. This research was supported, in part, by the National Science Foundation (NSF), grants CCF-0541417 and CCF-0946388.

References

[1] N.R. Adiga, M.A. Blumrich, D. Chen, P. Coteus, A. Gara, M.E. Giampapa, P. Heidelberger, S. Singh, B.D. Steinmacher-Burow, T. Takken, M. Tsao, P. Vranas, Blue Gene/L torus interconnection network, IBM Journal of Research and Development 49 (2005) 265–276.

[2] N. Agarwal, T. Krishna, L.-S. Peh, N.K. Jha, GARNET: a detailed on-chip network model inside a full-system simulator, in: International Symposium on Performance Analysis of Systems and Software, ISPASS, 2009, pp. 33–42.

[3] E. Baydal, P. Lopez, J. Duato, A simple and efficient mechanism to prevent saturation in wormhole networks, in: 14th International Parallel and Distributed Processing Symposium, IPDPS, 2000, pp. 617–622.

[4] C. Bienia, S. Kumar, J.P. Singh, K. Li, The PARSEC benchmark suite: characterization and architectural implications, in: 17th International Conference on Parallel Architectures and Compilation Techniques, PACT, 2008, pp. 72–81.

[5] C. Carrion, C. Izu, J.A. Gregorio, F. Vallejo, R. Beivide, Ghost packets: a deadlock-free solution for k-ary n-cube networks, in: Proceedings of the Sixth Euromicro Workshop on Parallel and Distributed Processing, 21–23 Jan. 1998, pp. 133–139.

[6] L. Chen, R. Wang, T.M. Pinkston, Critical bubble scheme: an efficient implementation of globally-aware network flow control, in: 25th IEEE International Parallel & Distributed Processing Symposium, IPDPS, 2011.

[7] W. Dally, B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers Inc., 2003.

[8] J. Duato, A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks, IEEE Transactions on Parallel and Distributed Systems (TPDS) 7 (1996) 841–854.

[9] C. Izu, Restrictive turning: deadlock freedom and congestion control for oblivious cut-through networks, in: 4th Australasian Computer Architecture Conference, ACAC, Auckland, 1999, pp. 251–262.

[10] C. Izu, C. Carrion, J.A. Gregorio, R. Beivide, Restricted injection flow control for k-ary n-cube networks, in: 10th International Conference on Parallel and Distributed Computing Systems, 1997, pp. 511–518.

[11] P.S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, B. Werner, Simics: a full system simulation platform, IEEE Computer 35 (2002) 50–58.

[12] M.M.K. Martin, D.J. Sorin, B.M. Beckmann, M.R. Marty, M. Xu, A.R. Alameldeen, K.E. Moore, M.D. Hill, D.A. Wood, Multifacet's general execution-driven multiprocessor simulator toolset, ACM SIGARCH—Computer Architecture News 33 (2005) 92–99.

[13] J.F. Martinez, J. Torrellas, J. Duato, Improving the performance of bristled CC-NUMA systems using virtual channels and adaptivity, in: Proceedings of the International Conference on Supercomputing, ICS, 1999, pp. 202–209.

[14] J. Miguel-Alonso, C. Izu, J.A. Gregorio, Improving the performance of large interconnection networks using congestion-control mechanisms, Performance Evaluation 65 (2008) 203–211.

[15] T. Moscibroda, O. Mutlu, A case for bufferless routing in on-chip networks, in: 36th Annual International Symposium on Computer Architecture, ISCA, 2009, pp. 196–207.

[16] L.-S. Peh, W.J. Dally, Flit-reservation flow control, in: 6th International Symposium on High-Performance Computer Architecture, HPCA, 2000, pp. 73–84.

[17] T.M. Pinkston, S. Warnakulasuriya, On deadlocks in interconnection networks, in: 24th Annual International Symposium on Computer Architecture, ISCA, 1997, pp. 38–49.

[18] V. Puente, C. Izu, R. Beivide, J.A. Gregorio, F. Vallejo, J.M. Prellezo, The adaptive bubble router, Journal of Parallel and Distributed Computing (JPDC) 61 (2001) 1180–1208.

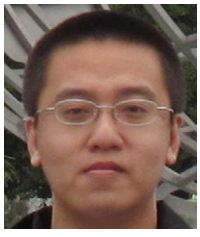
[19] Y.H. Song, T.M. Pinkston, A progressive approach to handling message-dependent deadlock in parallel computer systems, IEEE Transactions on Parallel and Distributed Systems (TPDS) 14 (2003) 259–275.

[20] Y.H. Song, T.M. Pinkston, Distributed resolution of network congestion and potential deadlock using reservation-based scheduling, IEEE Transactions on Parallel and Distributed Systems (TPDS) 16 (2005) 686–701.

[21] M. Thottethodi, A.R. Lebeck, S.S. Mukherjee, Exploiting global knowledge to achieve self-tuned congestion control for k-ary n-cube networks, IEEE Transactions on Parallel and Distributed Systems (TPDS) 15 (2004) 257–272.

[22] S. Warnakulasuriya, T.M. Pinkston, Formal model of message blocking and deadlock resolution in interconnection networks, IEEE Transactions on Parallel and Distributed Systems (TPDS) 11 (2000) 212–229.

Lizhong Chen is currently a Ph.D. student in Computer Engineering at University of Southern California. He received his B.S. degree in Electrical Engineering from Zhejiang University in 2009, and M.S. degree in Electrical Engineering from the University of Southern California in 2011. He was a research assistant in the Institute of VLSI Design at Zhejiang University in China from 2007 to 2009. His research interests focus on interconnection networks for parallel computing systems.



Ruisheng Wang is currently a Ph.D. student in Computer Engineering at University of Southern California. He received his B.S. degree in Computer Science from North China University of Technology, Beijing, China, in 2006 and his M.S. degree from Tsinghua University, Beijing, China in 2009. His research interests focus on computer architectures for multicore and multiprocessor system.

Timothy M. Pinkston received the BSEE degree from The Ohio State University in 1985 and the MSEE and Ph.D. degrees from Stanford University in 1986 and 1993, respectively. He is currently a professor in the Ming Hsieh Department of Electrical Engineering and Vice Dean of Faculty Affairs in the Viterbi School of Engineering at the University of Southern California. Recently, he served three years as the program director of the Computer and Information Science and Engineering Directorate of the US National Science Foundation (NSF) for the computer systems architecture area and the Expeditions in Computing Program. His research interests include interconnection networks and communication architectures for parallel processing systems, in particular multicore and multiprocessor computers. His professional service includes serving on the editorial board of IEEE Transactions on Parallel and Distributed Systems (TPDS) and on the IEEE TPDS Editor-in-Chief search and re-appointment committees. He has taken on leadership roles and membership in many conferences and workshops in the fields, including ISCA, HPCA, ICPP, IPDPS, NOCS and HiPC. He recently served as the Program Chair for ICPADS'06, the General Chair for IPDPS'07, and the Program Chair for HPCA'09. He is a fellow of the IEEE.