2011 IEEE International Parallel & Distributed Processing Symposium

Critical Bubble Scheme: An 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, California, USA
{lizhongc, ruishenw, tpink}@usc.edu

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,* and apply it successfully to *k*-ary *n*-cube networks for the general class of buffer occupancy-based network flow control techniques. Simulation results show that the proposed scheme can reduce the buffer access portion of packet latency by as much as 77%, leading to much lower average packet latency at medium and high network loads while sustaining 11% throughput improvement after network saturation.**

Keywords: interconnection network, bubble flow control, globally-aware network flow control

I. 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 interconnection 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 the 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 improving performance in terms of both latency and throughput [2-4]. 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 rely on simplified implementations which, oftentimes, are much less efficient.

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 Bubble Flow Control (BFC) [1] 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 preventing deadlock. This achieves nearly the same effect as would be provided by an omniscient global controller needed to implement theoretically-optimal BFC but does not incur the performance inefficiencies of the simplified implementation proposed in [1]. We investigate the effectiveness of the proposed Critical Bubble Scheme in 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 of this paper are the following.

- The potential advantages and existing challenges of implementing globally-aware flow control mechanisms are discussed relative to locally-aware flow control (Section II).
- The Critical Bubble Scheme is proposed as a way to implement near-theoretically-optimal BFC, which can reduce the minimum number of channel buffers needed to avoid deadlock by 50% as compared to Localized BFC (Section III). Buffer reduction is particularly useful for on-chip network environments.
- Critical Bubble Scheme is generalized for the case of using multiple critical bubbles to efficiently realize a large class of buffer occupancy-based globally-aware flow control mechanisms (Section III).
- Modifications to router microarchitecture are described for implementing the Critical Bubble Scheme (Section IV), and a cycle-accurate network simulator, SICOSYS [5], is used for evaluation (Section V). Simulation results show that Critical Bubble Scheme can reduce the buffer access component of packet latency by as much as 77%

over locally-aware flow control and can achieve much lower average message latency at medium and high network loads. With multiple critical bubbles, throughput improvement of 11% can be sustained.
- Related work on globally-aware flow control is summarized in Section VI. Several promising extensions of the proposed Critical Bubble Scheme are discussed in the conclusion (Section VII).

II. Globally-Aware Flow Control Mechanisms

We first provide background on why globally-aware flow control is preferred over locally-aware flow control and the challenges associated with it. We then present two cases to illustrate the inefficiencies of existing approaches for implementing globally-aware flow control mechanisms. This motivates the need for a better implementation scheme, which we propose in Section III.

A. *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 taken into consideration.

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 [3], 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 are some challenges in efficiently implementing globally-aware flow control. Key global information about network conditions need to be gathered and acted upon by an omniscient global controller. This requires prohibitively high overhead to implement or simplified implementations which can be inefficient. The following discusses two representative examples.

B. *Bubble Flow Control*

Bubble Flow Control proposed in [1] is a globally-aware flow control mechanism that reduces to a locally-aware flow control mechanism in a simplified implementation. 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 [7]. 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). 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 dependence that can occur across different dimensions. However, it does not prevent deadlock that can occur within dimensions caused by torus wraparound links. A classic solution to this problem is to use a dateline technique where two virtual channels (*high* and *low*) are associated with each physical channel [6]. When packets transported on *low* channels cross the dateline, they switch to *high* channels thus breaking cyclic dependence within network dimensions. A drawback of this approach, however, is the requirement for two virtual channels and the 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 [1]. 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 dimension (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's 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 a free buffer in the ring, which guarantees that at least one packet is able to make progress. This free buffer

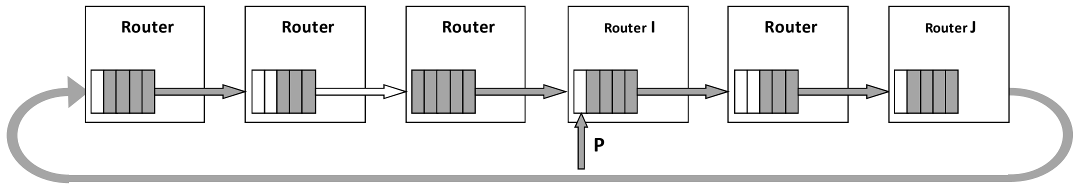


Figure 1. Theoretical BFC requires global coordination to avoid deadlock.

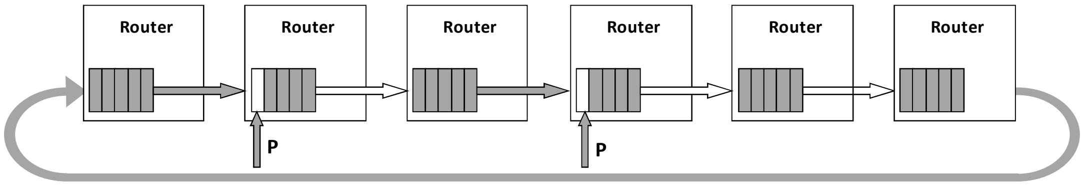


Figure 2. Simultaneous injection Theoretical BFC requires global coordination.

acts as a *bubble* and ensures deadlock freedom. A detailed mathematical proof can be found in the original paper [1].

2) Difficulty in Implementing Theoretical BFC: Theoretical BFC belongs to the class of globally-aware flow control techniques as the allocation of buffer resources requires buffer utilization information of other nodes in that directional ring. Therefore, the major difficulty in implementing this flow control technique is the need for a global controller. Specifically, the global controller must provide two services. First, it must gather and distribute global information about buffer utilization so that every node has sufficient global knowledge to make the optimal decision. Second, it must coordinate the allocation of resources globally. This is needed when multiple injections to the same directional ring are requested simultaneously, as illustrated in Fig. 2. Here, in the same cycle, two packets want 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. Without global coordination, both routers will allow injection based on their global knowledge of the existence of two buffers in the dimensional ring. Hence, the global controller must determine which injection should be granted to eliminate deadlock and avoid uncertainty. The above two important service requirements make realizing Theoretical BFC implementations challenging.

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, 7]. Instead of checking for the existence of two free buffers anywhere along the directional ring as in Theoretical BFC, the Localized BFC checks only that there are two free buffers in the receiving channel buffer. 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 turns the globally-aware Theoretical BFC technique into a locally-aware technique as, now, all decisions are made based solely on local information.

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.

Second, with Localized BFC, the minimum number of buffers needed in each router channel buffer 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 NoC designs may allow for only one packet-sized buffer per channel, which precludes the use of Localized BFC. This will be discussed further in Section III.

Third, this simplified implementation of 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.

The increased buffer access delay and low buffer utilization in Localized BFC leads to performance degradation of the network, as shown later in Section V.

C. Buffer Occupancy-Based Global Flow Control

We next consider buffer occupancy-based flow control, which can be thought of as a generalization of bubble flow control.

It is possible for some networks under certain traffic patterns to suffer from a throughput drop beyond the saturation point. In such situations, congestion control

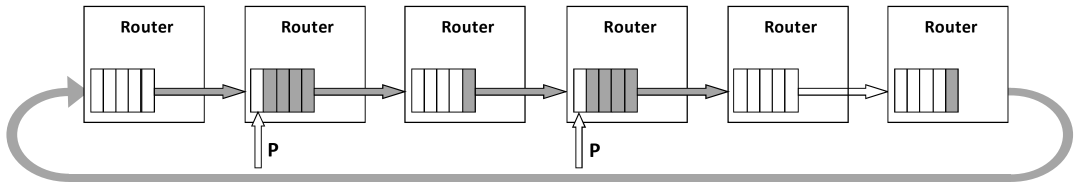


Figure 3. Localized BFC decisions are not optimal across the network.

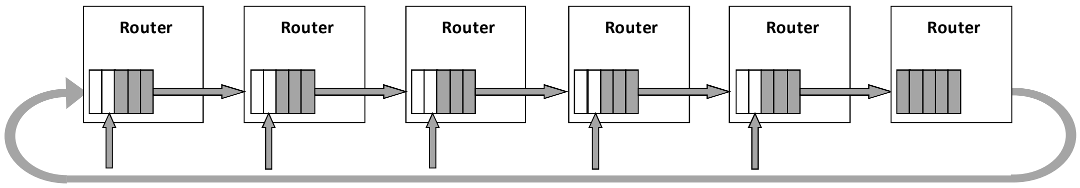


Figure 4. Localized BFC requires more free buffers than that are needed by Theoretical BFC.

mechanisms come into play. Many congestion control mechanisms are buffer occupancy-based and can be either locally or globally controlled. A typical example of local congestion control is when packets are allowed to be injected into a router channel only if the buffer occupancy of the channel is below a certain threshold. (Localized BFC can be considered as a special case of this. However, our simulation results show that Localized BFC suffers much higher latency because it is not designed for this purpose.) Alternatively, global congestion control grants the injection of packets only if the aggregate buffer occupancy of an entire dimension or region is below a certain threshold. The challenge remains realizing efficient implementations.

One implementation is described in [3]. It uses a dimension-wise aggregation scheme, where basically every node receives accumulated buffer occupancy information from its predecessor neighbor, updates this with its own buffer occupancy, and passes this information on to its successor neighbor. Besides the hardware overhead in this process, the aggregate delay could be considerable as every hop of buffer information propagation takes a number of cycles. By the time the global information arrives, there is a risk that it may already have become stale.

III. Critical Bubble Scheme

A. The Basic Idea

Our proposed Critical Bubble Scheme is *not* a new flow control technique; rather, it is an efficient implementation of Theoretical BFC that retains global awareness properties for realizing near-maximal benefits. 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. 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 the 2D torus shown in Fig. 5. Initially, for each unidirectional ring, 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. It 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 global awareness. The absence of the critical bubble at the local router indicates the existence of a free buffer elsewhere in the ring, providing implicit global awareness.

B. Detailed Decription 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.

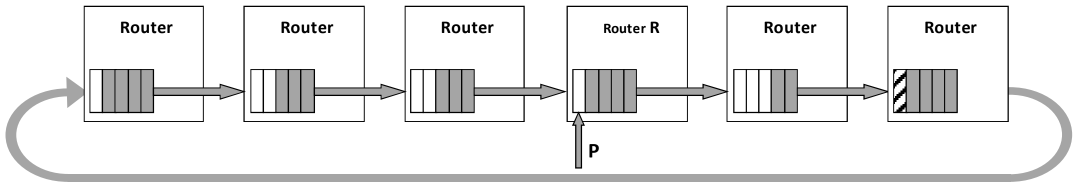


Figure 5. The Critical Bubble Scheme avoids deadlock without requiring explicit global coordination.

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

a) If a packet is allowed to change dimension or inject into a 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* is 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 for this is provided in Section IV.

4) Deadlock Freedom: Under the above two rules, no intra-dimensional deadlock will occur with Critical Bubble Scheme. Together with dimension-order routing, no inter-dimensional deadlock will occur either. In proving this, we first introduce some basic definitions derived from [8].

Definition: Below, Q represents the set of input queues associated to router nodes. Each queue $q_i \in Q$ has capacity of $cap(q_i)$ packets and the current number of occupied packets 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. $F(q_i, q_j)$ 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. 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.

Proof: 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.

$\boldsymbol{Rule\ 1}: When\ x = y, F(q_i, q_j)\ is\ True\ if$

$$size(q_j) \leq cap(q_j) - 1 \tag{1}$$

$\boldsymbol{Rule\ 2}: When\ x \neq y \vee q_i \in Q_I, F(q_i, q_j)\ is\ True\ if$

$$size(q_j) \leq cap(q_j) - 1 \wedge \tag{2}$$

$$\textstyle\sum_{\forall q_k \in Q_y} size(q_k) \leq \sum_{\forall q_k \in Q_y} cap(q_k) - 2 \tag{3}$$

The rules of Critical Bubble Scheme are the following.

$\boldsymbol{Rule\ 1}^*: When\ x = y, F(q_i, q_j)\ is\ True\ if$

$$size(q_j) \leq cap(q_j) - 1 \tag{4}$$

$\boldsymbol{Rule\ 2}^*: When\ x \neq y \vee q_i \in Q_I, F(q_i, q_j)\ is\ True\ if$

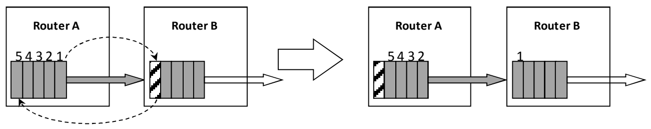


Figure 6. Transfer Critical Bubbles between routers.

$$size(q_j) \leq cap(q_j) - 1 \wedge \tag{5}$$

$$b_j = 0 \tag{6}$$

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), 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. 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 under its two rules in [1], the Critical Bubble Scheme is also deadlock-free. □

C. Impact on Implementation and Performance

The Critical Bubble Scheme provides an elegant and efficient implementation solution to overcome the difficulty of implementing Theoretical BFC while still avoiding the deficiencies of Localized BFC.

1) Implementing Global Awareness of Theoretical BFC: As discussed in Section II, the most difficult problem in implementing Theoretical BFC is the need for a global controller. The main advantage of Critical Bubble Scheme is its global awareness without the need for a 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 now router *A* will deny access as there is no normal free buffer left, and router *B* will grant access. Uncertainty is removed without explicit global coordination..

2) Avoiding the Deficiencies of Localized BFC: The Critical Bubble Scheme also avoids all the three deficiencies of Localized BFC. First, there is no increased access delay.

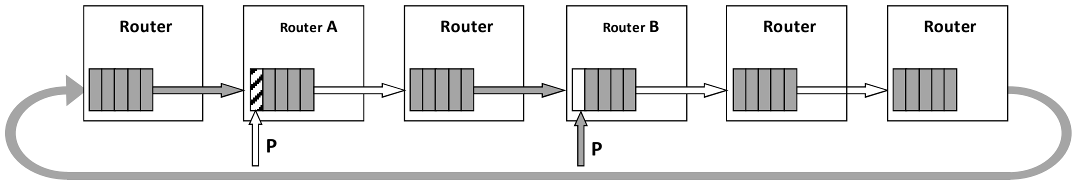


Figure 7. Simultaneous injections have no uncertainty with the Critical Bubble Scheme.

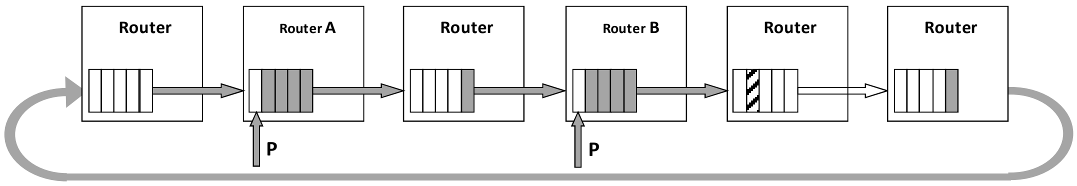


Figure 8. Simultaneous injection of multiple packets with the Critical Bubble Scheme.

Fig. 8 depicts the scenario in 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 having to incur extra buffer access delay.

Second, as mentioned in Section II-B, Localized BFC requires a minimum of two packet-sized buffers in every receiving channel while Theoretical BFC needs just one. The reason is that the Localized BFC has to check that there are at least two free buffers in the channel buffer before allowing packet 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 for the packet. Therefore, the Critical Bubble Scheme effectively achieves the minimum number of buffers in every channel as dictated by Theoretical BFC. This property is particularly useful when the interconnection network needs to support multiple message types (e.g., multiple message types in coherence protocols) or have limited buffer resources such as on-chip network. For example, in order to eliminate deadlock among all the message types, routers have to create one escape virtual channel for each message type. As a result, a large portion of the buffer resources is used to safeguard against deadlock. With the Critical Bubble Scheme, we could halve this amount and allocate the saved buffer resources to adaptive channels to speed up the common case.

Third, the Critical Bubble Scheme improves buffer utilization. As shown in Fig. 4, in the worst case, the critical bubble is in one of the five injecting router channels, requiring a second free buffer before access is granted. This leads to a total number of seven required free buffers, which is just one more than the theoretically-optimal case but three less than Localized BFC. In the best case, if the critical bubble is in the rightmost router, then the Critical Bubble Scheme achieves the minimum number of six required free buffers. In either case, the Critical Bubble Scheme greatly improves buffer utilization over Localized BFC.

D. Implementing Buffer Occupancy-Based Global Flow Control Using Multiple Critical Bubbles

So far we have used Critical Bubble Scheme to correctly and efficiently implement Theoretical BFC. Since buffer occupancy-based global flow control is a generalization of Theoretical BFC, we can generalize the Critical Bubble Scheme to realize this kind of global flow control.

The idea here 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 is that we mark multiple free buffers as critical bubbles during initialization. The same two rules are enforced for forwarding packets, and deadlock freedom will still be guaranteed. As an example, suppose we want to implement global flow control which restricts the injection of packets into dimensions only if the aggregate buffer occupancy of the entire dimension is below 80%. In this case, we mark 20% of the total buffers as critical bubbles in each unidirectional ring during the initialization phase. Then, according to the two rules, since every injected packet can consume only one normal free buffer in the router channels, we ensure that at least 20% of the total buffers in each dimension are not occupied when injection is allowed. Note that the existence of 20% critical bubbles does not influence the packets that are already in the dimension. These intra-dimensional packets are able to use all the buffers freely no matter whether they are critical bubbles or not. Thus, the net effect of multiple critical bubbles is to realize the desirable global flow control without losing performance.

With the Critical Bubble Scheme, we could easily implement various global flow control mechanisms that are buffer occupancy-based. The following are some examples.

a) Global flow control with a preset buffer occupancy threshold T: This class of global flow control can be implemented by the Critical Bubble Scheme as the previous example shows.

b) Self-tuned global congestion control: In this class of flow control, the threshold *T* can be adaptively adjusted at runtime. To implement it, a router can be designated as the bubble manager in each dimension. This router records the current number of critical bubbles in the dimension in a special register and calculates the most appropriate number of critical bubbles according to a certain algorithm. After comparing these two numbers, it can absorb a critical

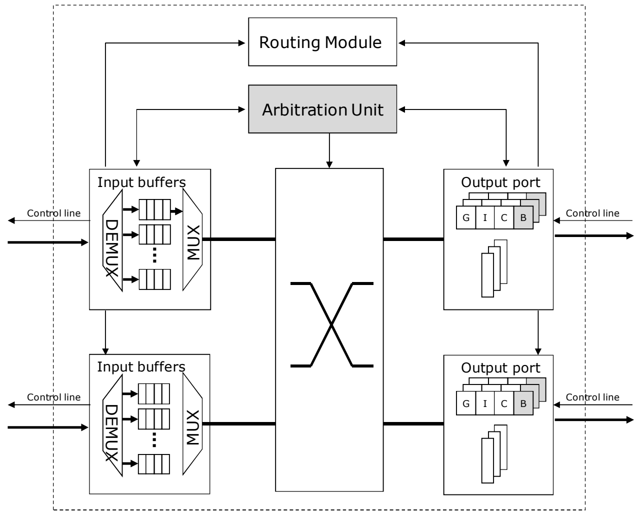


Figure 9. Typical virtual cut-throutgh router microarchitecture. The shaded areas are modified to implment the proposed Critical Bubble Scheme.

bubble by unmarking it or release a new critical bubble by marking the next available free buffer in the local channel.

c) Hybrid local and global flow control: In this class of global flow control, there is both a global buffer occupancy threshold, T_G, for the whole dimension and a local buffer occupancy threshold, T_L, for the local router channel to provide better fairness. To implement it, some critical bubbles are kept in local channels as determined by T_L, and the remaining critical bubbles are allowed to float in the dimension.

As these buffer occupancy-based global flow control mechanisms determine theoretically how much performance improvement they could achieve over local flow control, the objective of the Critical Bubble Scheme is to provide an efficient way to make these global flow control mechanisms practical and achieve maximum benefits.

IV. Router Architecture

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

A. Typical Virtual Cut-through Router Microarchitecture

Fig. 9 shows a block diagram of the microarchitecture 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 are normally merged into a single module [6], denoted as the Arbitration Unit here.

The output virtual channel also contains three states G, I and C to track the status of each virtual channel, as described in Table I.

TABLE I. Output Virtual Channel State Feilds

Field	Name	Description
G	Global state	Either idle (I), active (A), or waiting for credits (C)
I	Input VC	Input port and virtual channel that are forwarding packets to this output virtual channel
C	Credit count	Number of free buffers available to hold packets from this physical channel at the downstream node

B. Router Microarchitecture for Critical Bubble Scheme

The router microarchitecture to enable the Critical Bubble Scheme is very similar to the typical virtual cut-through router microarchitecture. While we could add a 1-bit field per packet-sized buffer to indicate whether it is a critical bubble, alternatively, we can use a counter for the entire input channel to maintain correct operation, which has lower hardware overhead. As a result, we need only three modifications to the router microarchitecture as shown shaded in Fig. 9: a counter *B* at the output channel to count the number of critical bubbles in the input channel buffer 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. The original arbitration unit checks the following condition before calling the arbitration algorithm:

1) G = A (i.e., the output virtual channel is active),

2) There is packet entering or waiting in the input channel, and

3) C should be greater than 0 (i.e., the downstream input channel has free buffer).

The modified arbitration unit checks the following:

1) G = A,

2) There is packet entering or waiting in the input channel,

3) C should be greater than 0, and

4) If the input channel is not in the same dimension as the output channel, then B < C.

The added checking makes sure that there is at least one normal free buffer as the downstream input channel has *C* free buffers but only *B* critical bubbles. Since the output channel has a field *I* which records the input port, it just needs only two comparators for the added checking.

Whenever intra-dimensional packet forwarding is granted, the arbitration unit checks whether *B* equals *C* before it is forwarded. If it does, the forwarded packet will occupy a critical bubble in the downstream input channel. Therefore, the critical bubble should be transferred backwards to the current input channel. This is done by decreasing *B* by 1 and decreasing the credit count. Since the number of critical bubbles of the current router is stored in the upstream router, we assert the 1-bit control line associated with the upstream router (denoted by *control line* in the figure). 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.

V. Evaluation

In this section, we present a detailed evaluation of our proposed Critical Bubble Scheme. Using simulation, we quantitatively validate the correctness of the Critical Bubble Scheme, examine its effects on buffer utilization, and compare its performance against other bubble-based flow control mechanisms.

A. *Simulation Methodology*

To evaluate the Critical Bubble Scheme, we implement it on a general-purpose cycle-accurate interconnection network simulator SICOSYS (SImulator of Communication SYStems) [5] written in C++. The assumed router microarchitecture is a 4-stage pipelined bubble router [1] with link latency of one cycle. We simulate an 8-ary 2-cube torus network with bidirectional physical channels. Each router node has one injection channel and four input buffer channels, one for each direction. Deadlock avoidance based on bubble flow control is assumed using one escape virtual channel associated with each physical channel and oblivious dimension-order routing over the torus.

All simulations are run for 10,000 cycles with a warm-up period of 2,000 cycles. In order to observe the effect of flow control after the network saturation point, injected load rate ranges from 0 to 1 flits/cycle/node. To stress the network, four synthetic traffic patterns are used in these evaluations: uniform random, perfect-shuffle, transpose, and tornado—which is an adversarial traffic pattern on torus topologies [6]. We compare the performance of Theoretical BFC, Localized BFC, Best local flow control (Best Local FC), and Critical Bubble Scheme using single and multiple critical bubbles. Best Local FC is the best local flow control that can be obtained by varying the minimum number of free buffers required in the local channel before injecting a packet. For example, Localized BFC corresponds to the case of two free buffers. By choosing the optimal number, Best Local FC provides a performance upper bound of what local flow control can achieve. In order to provide finer granularity for locally-aware congestion control, each buffer channel has eight 8-flit-sized buffers. Thus, the simulation environment is a biased in favor of locally-aware flow control compared to globally-aware flow control using Critical Bubble Scheme.

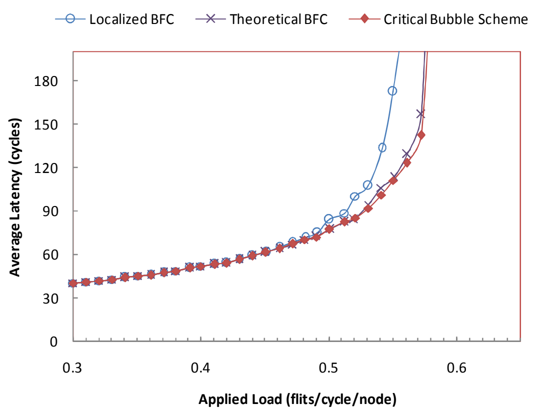
B. *Effects of Using a Single Critical Bubble*

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 [1], and our proposed Critical Bubble Scheme (CBS) using a single critical bubble. 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. In fact, there could be a small improvement of CBS over Theoretical BFC. This is because when the buffer occupancy is very high and there is only one free bubble in the next input queue, Theoretical BFC can still inject a packet into the input queue while CBS allows the injection only if it is a non-critical bubble. Therefore, CBS has a slight implicit throttling effect to keep the network from overloading. However, this difference between CBS and Theoretical BFC is minimal.

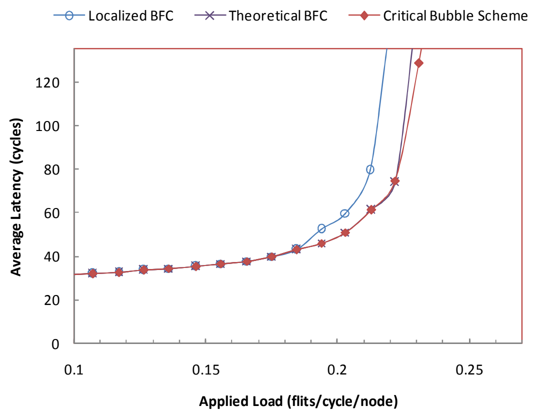
When comparing the performance of CBS with Localized BFC under these four traffic patterns, CBS clearly has advantages. Specifically, under uniform random, perfect shuffle and tornado traffic patterns, CBS has much lower latency at medium and high load rate (relative to the load rate at saturation point unless otherwise stated). This is because when the applied load rate is low, buffer occupancy is also 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 in doing so. However, as the applied load rate increases, the deficiencies of increased buffer access delay and low buffer utilization in Localized BFC gradually manifest while the CBS, which has lower buffer requirement, enjoys a higher success rate for injection and changing dimension. The above lead to performance gains for the Critical Bubble Scheme under medium and high load rates.

In the case of transpose traffic pattern, as Grot, et al., analyzed in [9], matrix transpose permutation directs all nodes in a given row to send packets to a given column. This makes some turn nodes extremely congested. While CBS is able to inject more packets into these busy routers because other routers are free, the increased number of intra-dimensional packets may result in a longer latency due to resource competition. However, this phenomenon happens only when certain routers are extremely congested (as in the transpose traffic pattern) and the rise in resource competition latency is very little compared with Localized BFC. In order to confirm the analysis of transpose traffic pattern, Fig. 11 plots a decomposition of latency for Localized BFC and CBS at load rates of 0.18 – 0.21 flits/cycle/node. As expected, CBS reduces buffer access delay by 30% – 70%, but increases the other components of latency. The net effect is that CBS with a single critical bubble has only slight performance improvement over Localized BFC for the transpose traffic pattern. As one might expect, CBS with multiple critical bubbles can solve this problem, as discussed below. Note that although the performance of CBS and Localized BFC are similar for this traffic pattern, CBS requires only half of the number of buffers in the router channel to ensure deadlock freedom.

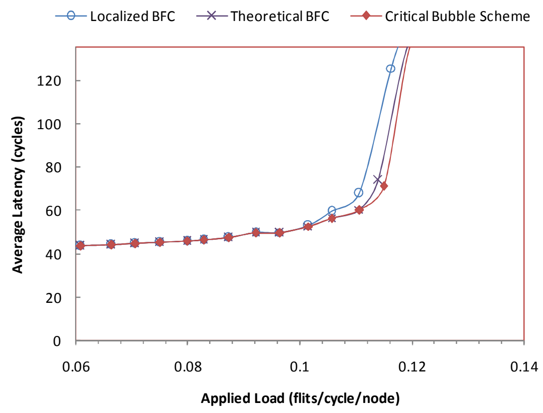
To further demonstrate the effect of CBS on reducing buffer access delay, Fig. 12 plots access delay of CBS normalized to that of Localized BFC at medium and high load rates under uniform random, perfect-shuffle, tornado, and transpose traffic patterns. The histogram shows significant reduction in buffer access delay for injecting or changing dimensions of packets by up to 77% for the Critical Bubble Scheme. This indicates that our proposed CBS



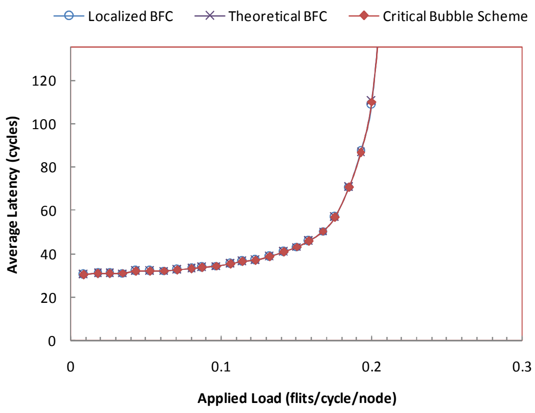
(a) Uniform Random



(b) Perfect Shuffle



(c) Tornado



(d) Transpose

Figure 10. Effects of using a single critical bubble for CBS under different traffic patterns.

successfully overcomes the deficiencies of Localized BFC and achieves lower access delay and higher buffer utilization.

C. *Effects of Using Multiple Critical Bubbles*

As described in Section II, sometimes networks under certain traffic patterns may suffer from a throughput drop beyond the saturation point. In such situations, it is useful to have some throttling flow control mechanisms in place, either local or global, to deal with it. In investigating how CBS can help with this issue, we compare three flow control mechanisms as plotted in Fig. 13. The Best Local FC is the best local flow control possible for the simulated network as explained earlier. CBS-Single is our proposed Critical Bubble Scheme using only one critical bubble, and CBS-Multiple is the implementation of a buffer occupancy-based global flow control mechanism that uses the optimum number of critical bubbles.

Fig. 13(a) shows a typical case where the throughput-drop problem exists under uniform random traffic pattern for CBS-Single. This is because by maintaining only one bubble in each unidirectional ring, it is susceptible to unstable throughput. In the worst case, when all channel buffers are full in the unidirectional ring except for the bubble, only one packet can make movement per cycle, resulting in a decreased throughput. The Best Local FC avoids this problem by restricting packet injection to the local channels to keep the network below saturation. The best scheme in comparison is CBS-Multiple. As shown in the figure, it has the best characteristics in both latency and throughput. This result indicates two things. First, global flow control does have potential benefits and perform better than local flow

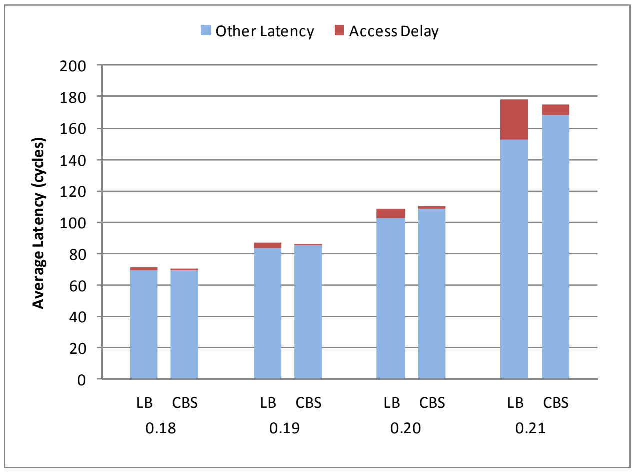
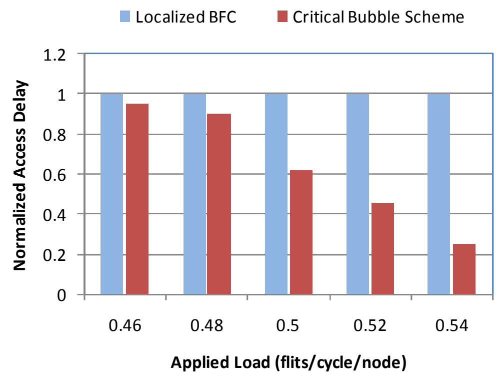
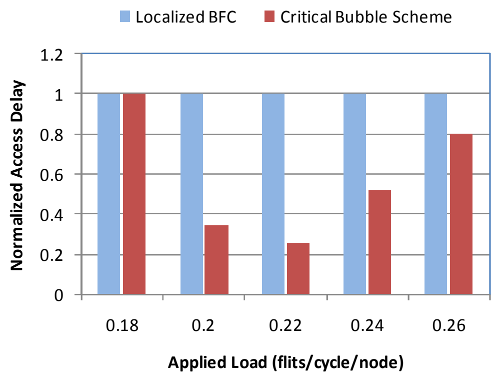


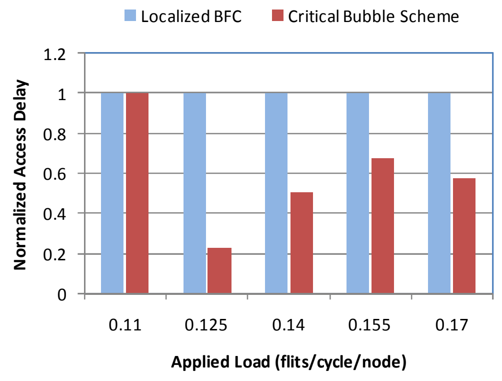
Figure 11. Latency decomposition comparison of Localized BFC and CBS for transpose at applied load of 0.18, 0.19, 0.20 and 0.21 flits/cycle/node.



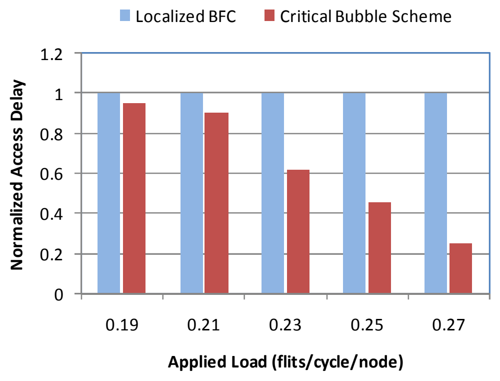
(a) Uniform Random



(b) Perfect Shuffle



(c) Tornado



(d) Transpose

Figure 12. Effect of the Critical Bubble Scheme on reducing buffer access delay.

control because of its earlier detection of network condition change, finer-granularity in tuning buffer occupancy threshold, and best decision making on behalf of the entire network. Second, with multiple critical bubbles, CBS is able to correctly and efficiently implement buffer occupancy-based global flow control and achieve its potential benefits. Fig. 13(c) shows a similar result for tornado traffic pattern.

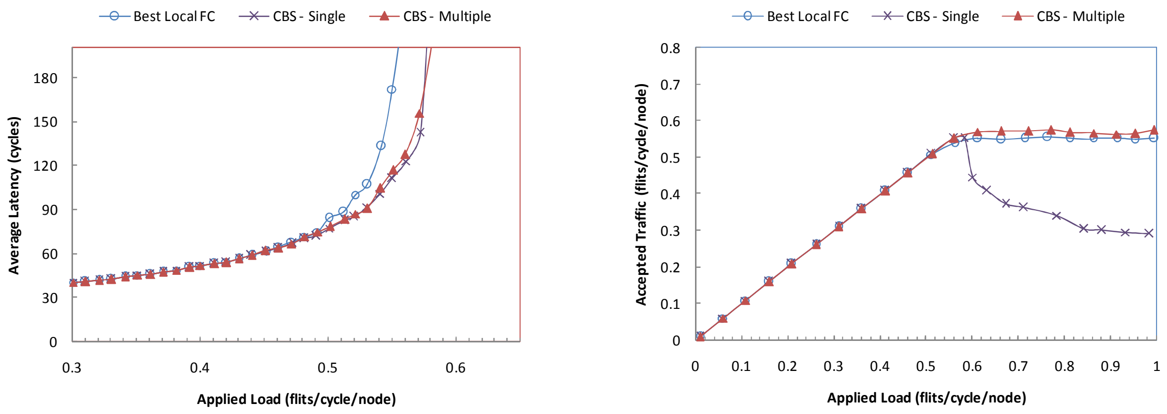
Fig. 13(b) and (d) present another scenario in which none of the three flow control mechanisms suffer from unsustainable throughput under these traffic patterns. In perfect-shuffle plotted in Fig. 13(b), CBS-Single and CBS-Multiple have very close performance, and both are better than the Best Local FC. In transpose plotted in Fig. 13(d), CBS-Single and Best Local FC have similar latency results, but CBS-Single has better throughput. Note that here, CBS-Multiple ameliorates the situation of injecting a lot of packets into the busy (congested) router as in CBS-Single, thus delivers the best latency among the three and 11% higher throughput than Best Local FC.

In summary, CBS-Single is a better flow control candidate than local flow control when there is no throughput-drop phenomenon, but Best Local FC can be useful for throttling to help sustained throughput. However, in all cases, CBS-Multiple correctly and efficiently implements buffer occupancy-based global flow control, and performs the best over both CBS-Single and Best Local FC.

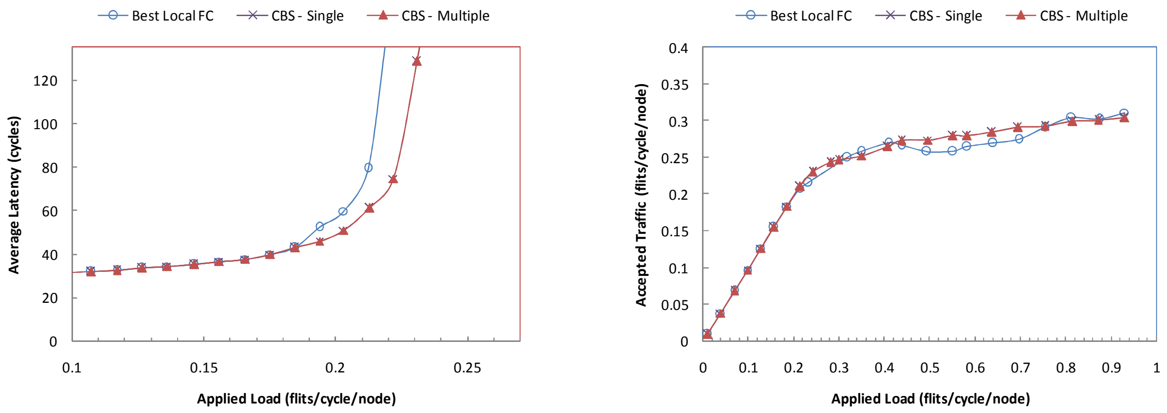
D. Effects of Network Scalability

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 network. Simulation results show that under uniform random traffic and high injection rate (i.e., 95% of the corresponding saturation load rate), average packet latency improvement of CBS-single over Localized BFC for network sizes of 4x4, 8x8 and 16x16 are 12.8%, 15.2% and 19.8%, 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 8 to 6 to 4 buffers per channel under uniform random traffic and high injection rate, average latency improvement of CBS



(a) Uniform Random



(b) Perfect Shuffle

Figure 13. Effects of using mulitple critical bubbles for CBS under different traffic patterns.

over Localized BFC increases from 15.2% to 21.2% to 31.6%, respectively.

VI. Related Work

The notion of using bubbles in flow control to avoid deadlock in torus networks was first proposed in [1] and adopted in IBM Blue Gene/L [7]. However, only a localized version of bubble flow control was actually implemented in those networks as opposed to a globally-aware version as is proposed here.

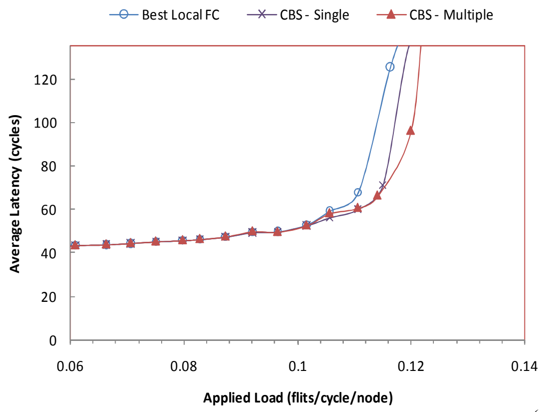
Several works propose ways of improving buffer utilization and/or reducing buffer requirements. In [10], bubbles are drawn to heavily-congested spots in network to relieve congestion and maintain deadlock freedom. Flit-reservation flow control was proposed in [12] to use buffers efficiently and reduce latency by scheduling buffer usage ahead of time. In [11], aggressive bufferless routing is 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.

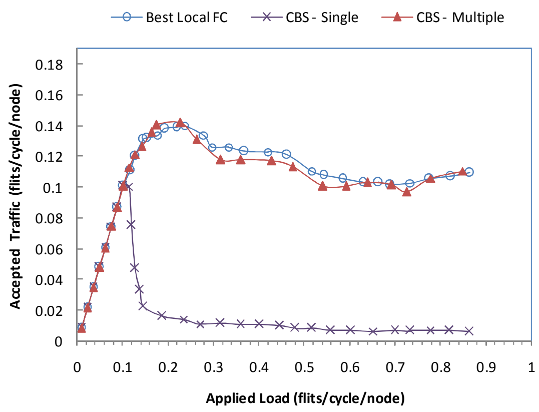
Extensive evaluations of using global congestion control were presented in [2] and [4], which demonstrated the advantages of globally-aware flow control from a simulation perspective. An ideal global controller is assumed in the evaluation, which was impractical. Self-tuned congestion control using global knowledge is proposed in [3]. It used a dimension-wise aggregation scheme to gather and propagate global information one hop after another, and the performance was better than the ALO scheme [13]. However, it has to endure a long delay to gather global information while our scheme can make grant decisions instantaneously.

VII. Conclusions and Future Work

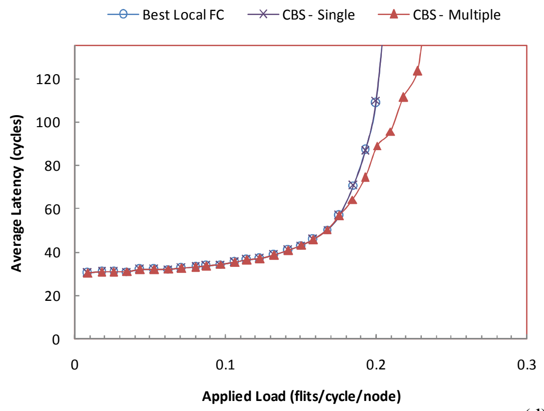
Globally-aware flow control in interconnection networks has many potential advantages over locally-aware flow control but faces several serious implementation difficulties. The primary contribution of this paper is the development of Critical Bubble Scheme (CBS) to provide a way to correctly and efficiently implement globally-aware flow control mechanisms. By marking a certain number of free buffers as critical bubbles and appropriately using them to restrict packet injection, this scheme achieves the same effect as if there were an omniscient global controller while ensuring deadlock freedom.

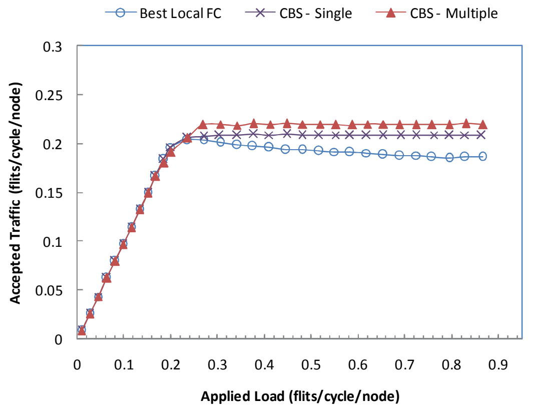
Several promising extensions based on the CBS can be explored in future work. The effectiveness of CBS on multiple message types can be investigated, for instance for chip multiprocessors using MOESI cache protocol.





(c) Tornado





(d) Transpose

Figure 13. Effects of Using Mulitple Critical Bubbles under different traffic patterns.

Implementation of other global flow control mechanisms (self-tune, hybrid local & global, etc.) can also be explored.

VIII. ACKNOWLEDGEMENTS

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

REFERENCES

[1] V. Puente, C. Izu, R. Beivide, J. A. Gregorio, F. Vallejo and J. M. Prellezo, "The adaptive bubble router," *Journal of Parallel and Distributed Computing,* vol. 61, pp. 1180-208, 2001.
[2] J. Miguel-Alonso, C. Izu, and J. A. Gregorio, "Improving the performance of large interconnection networks using congestion-control mechanisms," *Performance Evaluation,* vol. 65, pp. 203-211, 2008.
[3] M. Thottethodi, A. R. Lebeck, and 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,* vol. 15, pp. 257-272, 2004.
[4] C. Izu, C. Carrion, J. A. Gregorio, and R. Beivide, "Restricted injection flow control for k-ary n-cube networks," in *Proceedings of 10th International Conference on Parallel and Distributed Computing Systems,*Raleigh, NC, USA, 1997, pp. 511-18.
[5] V. Puente, J. A. Gregorio, and R. Beivide, "SICOSYS: an integrated framework for studying interconnection network performance in multiprocessor systems," in *Proceedings 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, 9-11 Jan. 2002*, Los Alamitos, CA, USA, 2002, pp. 15-22.
[6] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*: Morgan Kaufmann Publishers Inc., 2003.
[7] N. R. Adiga, M. A. Blumrich, D. Chen, et al., "Blue Gene/L torus interconnection network," *IBM Journal of Research and Development,* vol. 49, pp. 265-276, 2005.
[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,* vol. 7, pp. 841-54, 1996.
[9] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Express cube topologies for on-chip interconnects," in *IEEE 15th International Symposium on High Performance Computer Architecture, 14-18 Feb. 2009*, Piscataway, NJ, USA, 2009, pp. 163-74.
[10] Y. H. Song and T. M. Pinkston, "Distributed resolution of network congestion and potential deadlock using reservation-based scheduling," *IEEE Transactions on Parallel and Distributed Systems,* vol. 16, pp. 686-701, 2005.
[11] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *ISCA 2009 - 36th Annual International Symposium on Computer Architecture, June 20, 2009 - June 24, 2009*, Austin, TX, United states, 2009, pp. 196-207.
[12] L.-S. Peh and W. J. Dally, "Flit-reservation flow control," in *Sixth International Symposium on High-Performance Computer Architecture, January 2000*, Toulouse, France, 2000, pp. 73-84.
[13] E. Baydal, P. Lopez, and J. Duato, "A simple and efficient mechanism to prevent saturation in wormhole networks," in Proceedings 14th International Parallel and Distributed Processing Symposium. IPDPS 2000, 1-5 May 2000, Los Alamitos, CA, USA, 2000, pp. 617-22.